

Universidade do Minho
Escola de Engenharia

Rui Pedro Amaral Rodrigues

Robust and hardware-accelerated 3D point and line reconstruction from images

Tese de Doutoramento
Informática

Trabalho efectuado sob a orientação de:

Doutor António Ramires Fernandes
Doctor C. W. A. M. van Overveld

Novembro 2005

É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE, APENAS PARA EFEITOS DE INVESTIGAÇÃO,
MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

PARTIAL REPRODUCTION OF THIS THESIS IS AUTHORIZED FOR RESEARCH PURPOSES ONLY,
REQUIRING A WRITTEN DECLARATION OF THE INTERESTED PARTY.

Acknowledgments / Agradecimentos

Anyone who was ever related to a PhD work (directly or indirectly) knows that it requires a large amount of scientific and personal support. This work was no different in that sense, and therefore I would like to thank those who contributed to this work and its conclusion.

Undoubtedly, a major contribution for this thesis came from António (Tozé) Fernandes. His broader view of things and at the same time his sense of detail helped me many times to see things clearer, and to come up with new solutions to problems that had not occurred to me before. His practical sense, and the ability to go hands-on when needed was crucial in some points of this work. This also led, of course, to doing a couple of extra tests more often than we would like, but it was for a good cause. I thank him for relentlessly coping with me during this time, for the extensive reviewing work, and for his patience in the final lap. I hope to continue to enjoy our fruitful discussions and his friendship and his family's.

In the Netherlands, I had the pleasure of getting to know Kees van Overveld at Philips, when he supervised my traineeship. He seemed to me at the time as a remarkable scientist, able to come up very rapidly with interesting and novel ideas in the most varied fields. The brainstorming discussions we had many times, both during my traineeship and my PhD work, among other occasions, confirmed that idea. Kees' input in this thesis was invaluable, both in terms of generating and discussing ideas, of providing general guidance and foreseeing things, and in terms of proof-reading and correcting. I also admire his devotion to music and drawing, and I keep in my memory a first dinner together with him and his wife, at his place, where we spent the evening playing together. Even though such occasions were rare, the impression of them lasts. I thank him for supervising me and for giving me the privilege of knowing a bit of his artistic work.

Fabian Ernst was also a key person in my work at Philips. His insightful comments, help and availability were very important to me, sometimes maybe more than he would realize. Many times he provided me the link I was missing between theory and practice, and helped me structuring some confused ideas. He truly is to me the best example of how an applied mathematician should be.

Another very important person for me at Philips was Frans Peters. His kindness toward everyone, his eagerness to help and solve problems and his knowledge are also to me a striking role model. I am sure that all trainees that went from Minho to Frans' group will agree with me, as anyone else who knows him. I thank him for all the support, interest and help he gave me through these years. I also thank Elko, as the leader of IST group at Philips Research Eindhoven, for supporting my research there.

Continuing in the Netherlands: I have an eternal debt of gratitude to my Dutch Moms: Karen, Liesbeth and Ada. It was most comforting to arrive in cold, rainy Eindhoven and have a nice cup of hot tea and cookies, or a nice dinner, and a nice evening talking about the recent news in each country and families. They always made me feel at home, which was most important to me, and I look forward to be able to reciprocate that. I found a portuguese second-family in Utrecht, through my good friend João Saraiva, who was the main responsible for my Dutch Adventure. I will always remember the first time I met him and Joana, already there in Utrecht, and I keep great memories of long dinners, weekends, talks and parties with Cristina, Dora, Francisco and Cândida. João also gave me the privilege of knowing Luc and Annemieke, whom I admire in many ways, and I thank for the friendship and hosting when I visit Utrecht. I am sure I will keep meeting all of them both in Portugal and in the Netherlands.

In recent years, I also had the luck to get to know better a dear friend, Patric. I admire him for the openness, simplicity and kindness with which he deals with people and problems.

I thank him for all the nice dinners, parties and talks (serious and less serious), and general support. I still feel responsible by his more frequent trips to Portugal, but I think it is for a good cause. I cannot forget Bernardo and Natalia, who where always there for me, especially in the first hard days in the Netherlands. I also thank to all the people at Philips that helped in one way or another: Arno, Robert-Paul, Marcel, Bart, Jan, Koen, Orlando, Lut, Ronald and others.

I will now continue in Portuguese. É difícil enumerar todos aqueles que de uma forma ou de outra me ajudaram a chegar aqui, alguns deles pelo simples facto de estarem presentes (fisica ou mentalmente). As minhas primeiras palavras vão para o Miguel Costa, meu "companheiro de armas" na fase derradeira. A sua perseverança em situação semelhante à minha e quase em simultâneo foi imprescindível para chegar aqui (ele, tal como a Isabel e o Pablo, percebem bem como as coisas se passam), bem como o seu apoio e mão-de-obra na logística final de preparação do documento. Um enorme Obrigado. Espero que finalmente a nossa "Jamaica" se torne uma realidade, mesmo que alternativa.

Quero agradecer também à minha segunda-família do Coro Académico da Universidade do Minho (CAUM), por todos os momentos inesquecíveis que me deu e que decerto vai continuar a dar. Nada como um encontro de três ou quatro ou trinta elementos para levantar qualquer moral (para não falar das poderosas correntes de energia CAUM). Infelizmente, são demasiados para mencionar, além de que corro o risco de me esquecer de alguém. Por isso fica aqui um agradecimento colectivo, com uma atenção especial para o modelo de vida que é o Fernando Lapa, uma pessoa com um coração e génio musical enormes, que deixa uma impressão duradoura em quem o conhece. Obrigado por tudo. Gostaria também de agradecer às minhas outras segundas-famílias que estiveram presentes em tantos momentos. E são elas a minha família do Fontes – Paula, Orlando, Juliana, Mónica, Cláudia – a família dos Engenheiros de LESI – Luís, Paulo, Nuno, Ricardo Capote, Ricardo Viana, Múrias, João, Moisés, Sofia e outros – e a minha segunda família de LESI (que eu maltratei algumas vezes – *Dura Praxis, Sed Praxis* – mas ainda assim acho que não me expulsaram) – Margaça, Tróia, Marco, Pannuzzo, Bruno, Maik, Filipe, Sónia, Beatriz e outros.

Não existem palavras suficientes para agradecer às almas imensamente generosas que são o meu Pai António, a minha Mãe Guida, os meus Irmãos Andreia e Hugo, e a minha madrinha Rosa. Aos meus pais, pelo amor e presença sempre constante, pela vontade de ajudar, e pelo apoio incondicional a todos os níveis, mais do que por vezes seria merecido. São para mim o verdadeiro exemplo de amor paterno. À minha irmã Andreia, que cresceu tanto nestes últimos tempos em que eu não estive lá, e que me orgulha, entre outras coisas, pelo amor e confiança que sempre tem em mim e nos outros. Ao Hugo, pelo seu eterno à-vontade com (quase) tudo, e por manter os ânimos lá por casa. À Madrinha Rosa, por me mostrar o que é fazer anos sem envelhecer. A sua boa disposição constante é uma inspiração para qualquer um. A todos eles e à minha restante família, espero que me desculpem a ausência. Espero poder compensar-vos.

As honras finais estão reservadas à João, minha companheira em todos os momentos. Nunca conseguirei agradecer devidamente todo o apoio, a paciência, bondade e carinho com que sempre me acompanhou, principalmente nas alturas mais difíceis. A sua constante preocupação comigo e com os outros, e a sua capacidade de trabalho e de organização, entre muitas outras qualidades, são para mim um motivo de admiração e orgulho. Obrigado por existires comigo.

A todos, o meu muito obrigado.

Este trabalho foi apoiado pela Fundação para a Ciência e Tecnologia, bolsa PRAXIS XXI/BD/20322/99, e pelo Fundo Social Europeu - III Quadro Comunitário de Apoio.
This work had partial support of Philips Research Laboratories, Eindhoven.

Reconstrução robusta e acelerada por hardware de pontos e linhas 3D a partir de imagens

A área de Visão por Computador dedica-se a dotar os sistemas computacionais de capacidades de aquisição e processamento de informação visual, de forma a tornar possível uma série de tarefas – a detecção ou seguimento de objectos, o reconhecimento de objectos e símbolos – ou a resolver outros problemas mais complexos, como monitorização e vigilância, verificação de identidade ou controlo de qualidade.

Um problema importante que tem atraído a atenção dos investigadores desta área é o da *reconstrução 3D*, que além de ser uma tarefa complexa que pode envolver diferentes técnicas de visão, serve também como ferramenta para resolver problemas de maior abrangência.

De uma forma genérica, a reconstrução 3D pode ser definida como o processo de extrair informação 3D relativa a objectos do mundo real, com base na informação contida em imagens, ou produzida por sensores laser ou digitalizadores mecânicos. Quando baseado em imagens, o processo pode ser visto como o inverso de tirar fotografias. Tal processo permite a automatização de uma série de tarefas e aplicações, desde o cálculo de medições 3D para controle de qualidade, investigação forense ou reconstrução de acidentes, até à produção de conteúdos para sistemas de visualização 3D (p. ex. televisão 3D) a partir de conteúdos 2D já existentes.

Esta tese é dedicada ao ramo de reconstrução 3D baseado em imagens utilizando pontos e linhas, mais especificamente orientada aos casos em que as câmaras utilizadas não obedeçam a um posicionamento rígido ou pré-determinado.

Os sistemas envolvidos na aquisição e processamento de imagens e as sua limitações e especificidades levantam vários problemas ao processo de reconstrução. Existem três grandes fontes de problemas: a aquisição de imagens (distorção das lentes, limitações na resolução, quantização, ruído eléctrico, etc.), a calibração interna e externa das câmaras (erros de medida), e o estabelecimento de correspondências entre (partes de) imagens (causados não só pelos problemas de aquisição de imagem, mas também por oclusões, efeitos de perspectiva, reflexos e refracções ou ambiguidades nas imagens).

Estas questões, a variedade de configurações de câmaras e tipos de cenas representam problemas de peso para os métodos de reconstrução 3D. Surge assim a necessidade de métodos robustos que sejam capazes de reconstruir uma cena em caso de erros nos dados de entrada (calibração de câmaras, qualidade das imagens, etc.) e cenas mais complexas (com um grande número de oclusões ou reflexos, por exemplo).

Os métodos de reconstrução normalmente enfrentam uma série de compromissos entre a aplicabilidade a diferentes situações, a robustez e também o desempenho. Não existem métodos individuais definitivos que resolvam completamente o problema de reconstrução 3D em todos as situações. Quando se assumem fortes pressupostos e restrições, o problema torna-se mais tratável e simplificado, permitindo o uso de métodos de baixa complexidade. Porém, esses métodos são limitados a nível de aplicabilidade. Por outro lado, um método que se pretenda ser mais genérico e robusto a erros e variações nos dados de entrada, será naturalmente mais complexo e terá maior carga computacional. Isto pode comprometer a *usabilidade* em situações onde a capacidade de obter rapidamente dados de reconstrução é importante (por exemplo para determinar se são necessárias mais imagens ainda durante o processo de aquisição).

Esta tese pretende apresentar um melhor compromisso ao nível da robustez para uma gama

alargada de origens de erros, com um número de restrições reduzido, comparada com métodos existentes. Também é objectivo apresentar uma implementação de um método robusto que represente um bom compromisso com o desempenho, de forma a que a reconstrução se possa fazer a taxas que permitam interactividade. Um método com as características referidas pode também fornecer dados 3D robustos que sejam um ponto de partida para outros métodos de reconstrução que necessitem de uma boa aproximação inicial, complementando-os desta forma.

Para atingir estes objectivos, será proposto um novo método de reconstrução 3D de *pontos e linhas*, baseado em conjuntos de images (na ordem das dezenas), capturadas de pontos de vista arbitrários, e assumindo a existência de informação sobre a calibração das câmaras.

O método proposto explora uma combinação inovadora de conceitos, incluindo:

- ideias de diferentes métodos de reconstrução, tais como: o uso de pontos dos contornos, que reduzem as restrições na proximidade das câmaras entre si; o uso de múltiplas câmaras, que leva a um aumento de confiança nas estimativas; e o uso de um referencial absoluto, que facilita a integração de informação;
- técnicas de outros ramos de processamento de imagem, tal como o uso de transformadas de distâncias para criar uma métrica implícita para a avaliação de pontos;
- o uso de um mecanismo de votação, que contribui para a robustez pela acumulação de evidências;
- ferramentas das áreas de computação gráfica (p. ex. a projecção de texturas) e do hardware gráfico (o uso de API's gráficas standard) para aumentar o desempenho do método e torná-lo usável com hardware comummente disponível.

A versão básica do método dispõe de uma robustez implícita pela sua natureza de integração de informação, uma vez que todas as imagens contribuem simultaneamente para estimar cada ponto, promovendo assim a confiança nas estimativas. Dois passos adicionais são também propostos, que identificam e eliminam a maioria das estimativas incorrectas inicialmente. O resultado é um conjunto de pontos e linhas reconstruídos que cobre os objectos da cena de forma precisa.

Também é apresentada uma implementação acelerada por hardware que exlora as capacidades do hardware gráfico comummente disponível. Foi efectuada uma série de testes utilizando cenas sintéticas e reais, para analisar a qualidade, robustez e desempenho do método e da sua implementação. Os resultados apresentados mostram que é possível obter reconstruções utilizando pontos de vista arbitrários, simultaneamente de forma robusta e rápida, mesmo na presença de oclusões, reflexos e calibrações de câmara imperfeitas.

De uma forma geral, o método de reconstrução apresentado nesta tese é capaz de lidar com várias questões importantes que afectam outros métodos. Pode ser aplicado a configurações de câmaras afastadas e sem grandes restrições; é robusto a erros nas imagens e a variações de iluminação, problemas nos contornos e na calibração das câmaras; lida com oclusões e produz bons resultados sem necessitar de aplicar regularização; e tem um nível de desempenho que permite interactividade.

Robust and hardware-accelerated 3D point and line reconstruction from images

The research field of *Computer Vision* (CV) is devoted to provide to computer systems the ability to accept and process visual input in order to perform a series of tasks, such as object detection and tracking, object or symbol recognition, or for solving other complex problems, such as monitoring and surveillance, identity verification or quality control.

One important problem that draws the attention of computer vision researchers is *3D reconstruction*, which is in itself a complex task that can involve different computer vision techniques, and it can also be used as a tool for solving higher-level problems. Briefly stated, 3D reconstruction can be defined as the process of extracting 3D information regarding objects and scenes of the real world, based on inputs such as images, or data from laser or mechanical scanners. When based on images, it could be seen as the reverse of taking a picture. The availability of such a process enables the automation of a series of tasks and applications, from computing 3D measures for quality assessment, forensics or accident reconstruction, to producing content for 3D displays (which are gaining momentum) from existing 2D material.

This thesis focuses on the branch of 3D reconstruction methods using images captured from standard cameras as main inputs, without requiring or being bound to specific view points used for capturing the images or for representing the reconstruction, i.e. the focus is on view-independent methods.

There are multiple complications arising from the image acquisition and processing devices available and their limitations or specificities. Three major sources of possible issues are: image acquisition (lens distortion, resolution limitations, quantization, electrical noise, etc.), internal and external camera calibration (measurement errors), and correspondence establishment between (parts of) images (caused by image acquisition problems and others, namely occlusion, perspective effects, reflections and refractions or ambiguities in the images).

These issues and the variety of inputs that cause them to occur concurrently are major problems for 3D reconstruction methods. This calls for robust reconstruction methods that should be able to reconstruct a scene in settings with error sources in the inputs (e.g. camera calibration errors, noise in the images, etc.) and less favorable yet correct conditions, such as a high number of occlusions or highlights.

Reconstruction methods face a series of compromises, between the applicability to different settings, robustness, and also performance. There are no definitive individual methods that completely solve 3D reconstruction in all settings. When strong assumptions and constraints are applied, the problem becomes more conditioned and simplified, and less complex methods can be used. However, such methods are limited in terms of applicability. On the other hand, if a method is to cope with more general assumptions and to be robust to errors and input variations, it is in general bound to be more complex and thus computationally more demanding. This may compromise *usability* in many situations where the ability to quickly obtain reconstruction feedback, or in other words to *interact* with the reconstruction software, is important.

This thesis is aimed to present a better compromise in robustness for a wider range of error sources, with a lower number of restrictions, than those usually presented by other methods. It is also aimed to present an implementation of a robust method that has a good compromise with efficiency, such that interactive reconstruction rates can be achieved, without compromising robustness. Such a method is also expected to be useful for providing robust initialization

data for e.g. methods that require a good initial approximation of a scene's reconstruction i.e., complementing other reconstruction methods.

To accomplish those goals, a new *feature reconstruction* method, for points and lines, is proposed, based on sets of images (on the order of tens of images), captured from arbitrary viewpoints, and assuming known or *a priori* estimated camera calibration.

The method explores an innovative combination of concepts, including:

- ideas from different reconstruction methods, such as: the use of contour points, that reduces baseline restrictions; the use of multiple cameras, that leads to an increase in confidence of estimates; and the use of an absolute referential for estimation, facilitating integration of information;
- techniques from other fields of image processing, namely the use of distance transforms to provide an implicit correspondence metric for contour points;
- the use of a voting mechanism that yields robustness by accumulation of evidence;
- tools from the computer graphics and graphics hardware fields, such as projective texturing and graphics cards standard API's, that increase performance to the method and make it usable in current off-the-shelf hardware.

The core method provides basic inherent robustness due to its integrating nature, since all images contribute simultaneously to estimate each point, thus providing confidence in the estimation. Two additional steps for increasing robustness are presented, which identify and eliminate the majority of initial incorrect estimates. The result is a set of reconstructed points and lines that accurately covers the scene's objects.

An hardware-accelerated implementation of the method, exploiting standard *graphics hardware*, is also presented. Tests were performed on synthetic and real scenes, to assess the quality, robustness and performance of the proposed method and its implementation. The results are reported, and show that it is possible to achieve interactive and robust view-independent point reconstructions even in the presence of occlusions, highlights, and less-than-perfect camera calibrations.

Overall the reconstruction method presented in this thesis is able to deal with many important issues that hamper previous works. It can be applied to wide-baseline, weakly-constrained camera settings; it is robust to image noise and lighting variations, problems in contours and errors in camera calibration; it can cope with occlusion and produce good results without resorting to regularization; and it performs at interactive rates.

Contents

1 Introduction **1**
- 1.1 Computer vision and 3D reconstruction . . . 1
- 1.2 The geometric principles of binocular visual systems . . . 2
- 1.3 Issues of image-based 3D reconstruction . . . 3
  - 1.3.1 Image acquisition . . . 3
  - 1.3.2 Camera calibration . . . 4
  - 1.3.3 Correspondence . . . 4
- 1.4 Motivation . . . 7
  - 1.4.1 Robustness . . . 7
  - 1.4.2 Interactivity and usability . . . 8
  - 1.4.3 Trade-off between robustness and performance . . . 9
- 1.5 Thesis structure . . . 9

2 Background **11**
- 2.1 Geometric principles of image-based reconstruction . . . 11
  - 2.1.1 Cameras . . . 12
  - 2.1.2 Point projection and image generation . . . 14
  - 2.1.3 Reconstruction from multiple views . . . 16
  - 2.1.4 Correspondence . . . 17
- 2.2 Related work . . . 20
  - 2.2.1 Introduction . . . 20
  - 2.2.2 Fixed number of views . . . 23
  - 2.2.3 Multi-view methods . . . 28
- 2.3 Final remarks . . . 48

3 Multi-view 3D point reconstruction using contours and distance functions **51**
- 3.1 Introduction . . . 51
- 3.2 Contour points in the context of reconstruction . . . 53
  - 3.2.1 Classification of discontinuity sources . . . 54
  - 3.2.2 Contour extraction . . . 55
  - 3.2.3 Advantages and practical issues in the correspondence of contour points 58
- 3.3 Reconstruction of contour points . . . 58
  - 3.3.1 The main algorithm . . . 59
  - 3.3.2 Selection of the candidate set . . . 61
  - 3.3.3 Voting function . . . 63
  - 3.3.4 Vote analysis - multiple high maxima . . . 68
  - 3.3.5 Vote analysis - winner selection . . . 70
- 3.4 Robustness extensions . . . 73
  - 3.4.1 Cross checking . . . 73
  - 3.4.2 Merging of reconstructed points . . . 75

3.5 Reconstruction of feature points . . . 77
3.5.1 Classification of feature points . . . 78
3.5.2 Feature point extraction . . . 80
3.6 Comparison with other methods . . . 83
3.7 Summary and conclusions . . . 85

4 Accelerated reconstruction using commodity graphics hardware 89
4.1 Introduction . . . 89
4.2 Vote collection using projective texturing . . . 91
4.3 Vote accumulation for multiple cameras using additive blending . . . 95
4.3.1 Channel masking . . . 96
4.4 Definition of the candidates' rendering buffer . . . 97
4.4.1 Extrinsic parameters of the virtual camera . . . 98
4.4.2 Intrinsic parameters of the virtual camera . . . 99
4.4.3 From pixel coordinates to 3D candidates . . . 100
4.5 Final remarks . . . 100

5 Tests and results of point reconstruction 103
5.1 Introduction . . . 103
5.2 Quality assessment on synthetic scenes . . . 104
5.2.1 Error metric . . . 105
5.2.2 The reconstruction algorithm as a classifier . . . 107
5.2.3 Test setup . . . 110
5.3 Parameter testing with synthetic scenes . . . 111
5.3.1 Description of synthetic scenes and their contours . . . 112
5.3.2 Set of points . . . 114
5.3.3 Vote map range . . . 117
5.3.4 Cross-check threshold . . . 119
5.3.5 Similar camera angle . . . 123
5.3.6 Vote percent threshold . . . 126
5.3.7 Number of candidates . . . 130
5.3.8 Merging . . . 134
5.4 Effects of camera errors on synthetic scenes . . . 141
5.5 Tests with real scenes . . . 151
5.5.1 Audio speaker . . . 151
5.5.2 Boat . . . 156
5.5.3 Remarks on results with real scenes . . . 160
5.6 Performance tests . . . 161
5.6.1 Single-line buffer versus multi-line buffer . . . 161
5.6.2 Number of candidates . . . 161
5.6.3 Different hardware . . . 163
5.7 Summary and conclusions . . . 164

6 Line reconstruction 167
6.1 Introduction . . . 167
6.2 Variant A: Connecting merged points . . . 168
6.3 Variant B: Incremental line reconstruction . . . 173
6.4 Tests . . . 177
6.4.1 Quality assessment for line reconstruction . . . 177
6.4.2 Test setup . . . 178

6.4.3 Variant A . . . 179
6.4.4 Variant B . . . 188
6.4.5 Comparison . . . 194
6.5 Comparison with other line reconstruction methods . . . 199
6.6 Final remarks . . . 201

7 Conclusions and future work **203**
7.1 Conclusions . . . 203
7.2 Future work . . . 206

A Instructions for the accompanying CD **209**
A.1 Accessing the contents of the CD . . . 210
A.1.1 Using the menu . . . 210
A.1.2 Navigating through the directories and the files . . . 210
A.2 Interactive reconstruction viewer . . . 211
A.2.1 The control window . . . 212
A.2.2 The display window . . . 213

B Additional reconstruction images **215**

Bibliography **219**

Index **231**

Notation

World, cameras and images	
W	World space
$\mathbf{C}^i$	Camera i
I^i	Image taken from $\mathbf{C}^i$
$\mathbf{I}^i$	Internal parameters of $\mathbf{C}^i$
f^i	Focal length of $\mathbf{C}^i$ (world units)
w^i	Width of I^i (world units)
h^i	Height of I^i (world units)
$\mathbf{E}^i$	External parameters of $\mathbf{C}^i$
$\mathbf{c}^i$	Position of $\mathbf{C}^i$
$\mathbf{H}^i$	Horizontal ("right") vector of $\mathbf{C}^i$
$\mathbf{U}^i$	Vertical ("up") vector of $\mathbf{C}^i$
$\mathbf{L}^i$	Look vector of $\mathbf{C}^i$
π^i	Image plane for $\mathbf{C}^i$
$\widehat{\pi}^i$	Clipped image plane of $\mathbf{C}^i$

Points, view rays and epipolars	
$\mathbf{w}_x$	World point
V_x^i	View ray from $\mathbf{C}^i$ through point x
$\mathbf{p}_q^i$	Projected point q on I^i
d^i(x)	Depth of a point x relative to $\mathbf{C}^i$
$E_q^{i,j}$	Epipolar line of $\mathbf{p}_q^i$ in I^j
$\mathbf{v}_{q,x}^i$	Candidate point x selected from the view ray of $\mathbf{p}_q^i$
$\mathbf{p}_{q,x}^{i,j}$	Projection of candidate point $\mathbf{v}_{q,x}^i$ in I^j
$\mathbf{r}_x^i$	Reconstructed point (associated to reference $\mathbf{C}^i$)
$\mathbf{m}_x$	Merged point

Matching and voting	
$\sigma(\mathbf{p}_q^i, \mathbf{p}_{q,x}^{i,j})$	Match function between two points
$\upsilon_{q,x}^{i,j}$	Votes collected from I^j for the candidate $\mathbf{v}_{q,x}^i$
$\lambda_{q,x}^i$	Total accumulated votes for $\mathbf{v}_{q,x}^i$
Λ^i	Vote map for I^i

Projective texturing	
$\mathbf{J}^i$	Projector associated to $\mathbf{C}^i$
τ^i	Texture associated to $\mathbf{J}^i$
$\mathbf{T}^i$	Transformation matrix for τ^i

Error assessment with real data	
$\hat{\mathbf{w}}_p^i$	Real world point associated to a reconstructed point $\mathbf{r}_p$
$e_r\left(\mathbf{r}_p\right)$	Error of a reconstructed point $\mathbf{r}_p$
$e_m\left(\mathbf{m}_x\right)$	Error of a merged point $\mathbf{m}_x$

Line reconstruction	
L_m	Candidate line m
$l_{m,k}$	point k of candidate line L_m
$\upsilon_{m_k}^j$	Votes collected from I^j for $l_{m,k}$
λ_{m_k}	Total votes for $l_{m,k}$

List of Tables

2.1 Types of calibration and associated reconstructions 31
2.2 Comparison of reconstruction methods 50

3.1 Sources of image discontinuities 55

5.1 A confusion matrix 109
5.2 Parameters tested with synthetic scenes 111
5.3 Parameters values used on the speaker scene 153
5.4 Measurements of real and reconstructed segments: speaker scene 155
5.5 Measurements of real and reconstructed segments: boat scene 160
5.6 Performance results for varying numbers of candidates 162
5.7 Performance results for different hardware systems 163

6.1 Parameters tested in line reconstruction 179
6.2 Performance comparison between the two line reconstruction variants 199

A.1 List of file name labels 211

List of Figures

1.1 Parallax effect . . . 2
1.2 Distortions and artefacts . . . 5

2.1 A pinhole camera . . . 12
2.2 The pinhole camera model and associated geometry . . . 13
2.3 The frustum of a camera . . . 14
2.4 Point projection and occlusion . . . 16
2.5 Influence of the number of views on reconstruction . . . 17
2.6 Overlapping viewrays . . . 17
2.7 Epipolar geometry . . . 19
2.8 A classification tree of image-based reconstruction methods, based on number of input images . . . 20
2.9 Shape from shading . . . 21
2.10 Structured light . . . 21
2.11 Shape from texture . . . 21
2.12 Shape from focus . . . 22
2.13 Structure from motion . . . 22
2.14 Example of correlation-based correspondence estimation . . . 24
2.15 Image rectification . . . 25
2.16 Example of apparent contours . . . 26
2.17 Trifocal geometry . . . 27
2.18 Pollefeys' SfM technique [Poll 00] . . . 31
2.19 Zisserman's SfM technique [Fitz 98] . . . 32
2.20 Tsai's multi-baseline stereo [Tsai 83] . . . 34
2.21 Kanade et al's virtualized reality [Kana 97] . . . 34
2.22 Mellor's epipolar image approach [Mell 96] . . . 36
2.23 Szeliski and Weiss' apparent contour reconstruction [Szel 98] . . . 37
2.24 Example of a visual hull . . . 39
2.25 Silhouette cone intersection . . . 40
2.26 Voxel coloring . . . 40
2.27 Seitz and Dyer's voxel coloring [Seit 97] . . . 41
2.28 Kutulakos and Seitz's space carving [Kutu 00] . . . 41
2.29 Generalized voxel coloring [Culb 00] . . . 42
2.30 Collins' space sweep approach [Coll 96] . . . 44
2.31 Fua's particle-based reconstruction [Fua 95b] . . . 45
2.32 Zhang and Seitz's mesh refinement approach [Zhan 00] . . . 45
2.33 Example of a spatiotemporal volume and the effect of (apparent) motion . . . 47
2.34 Peng's spatiotemporal slices and strips [Peng 91] . . . 47

3.1 Different types of image discontinuity sources . . . 55
3.2 Example of edge detection based on gradient vs second derivative . . . 57

3.3 Problems of second derivative methods at T-shaped junctions . . . 57
3.4 Reconstruction using view ray candidates . . . 61
3.5 Binary-based vs distance-based voting function . . . 63
3.6 Example of a binary map and the corresponding distance map . . . 65
3.7 Distance map computation . . . 66
3.8 Out of frustum . . . 67
3.9 Occlusion of a candidate . . . 67
3.10 Example of a voting curve . . . 68
3.11 Cross-voting problem . . . 69
3.12 Plateaus . . . 70
3.13 Overlapped points . . . 71
3.14 Sparseness and the choice of merging thresholds . . . 77
3.15 Different types of contour points . . . 79
3.16 Contour following . . . 81
3.17 Contour lines as sequences of small line segments . . . 82
3.18 Line simplification . . . 83

4.1 Texturing . . . 92
4.2 Projective texturing . . . 92
4.3 Projective texturing for reconstruction . . . 93
4.4 Projective texturing with accumulation for reconstruction . . . 96
4.5 Orthographic camera for view ray segment rendering . . . 98

5.1 Problems using all the known w-points to compute reconstruction error . . . 106
5.2 Example of contour offsets . . . 106
5.3 Synthetic scenes . . . 113
5.4 Point set tests (reconstruction views) . . . 115
5.5 Point set tests (data chart) . . . 116
5.6 Vote map neighborhoods of one pixel using different vote map ranges . . . 117
5.7 Vote maps with different ranges in the case of close contours . . . 118
5.8 Vote range tests (reconstruction views) . . . 118
5.9 Vote range tests (data chart) . . . 119
5.10 Cross check threshold tests (reconstruction views) . . . 121
5.11 Cross check threshold tests (data chart) . . . 122
5.12 Similar camera angle tests (reconstruction views) . . . 124
5.13 Similar camera angle tests (data chart) . . . 125
5.14 Vote percent threshold example . . . 126
5.15 Vote percent threshold tests (reconstruction views) . . . 128
5.16 Vote percent threshold tests (data chart) . . . 129
5.17 Number of candidates tests (reconstruction views) . . . 131
5.18 Number of candidates tests (data chart) . . . 132
5.19 Error distributions for different error thresholds . . . 133
5.20 Distribution of point rejection . . . 133
5.21 Merging threshold 3D tests (reconstruction views) . . . 136
5.22 Merging threshold 3D tests (data chart) . . . 137
5.23 Merging threshold 2D tests (reconstruction views) . . . 139
5.24 Merging threshold 2D tests (data chart) . . . 140
5.25 Examples of the offsets introduced by the camera errors . . . 143
5.26 Reconstructions with camera errors . . . 144
5.27 Camera error tests (using default settings) . . . 145
5.28 Vote range tests (with camera errors) . . . 146
5.29 Cross-check threshold tests (with camera errors) . . . 147

5.30 Similar camera angle tests (with camera errors) . . . 148
5.31 Merging threshold 3D tests (with camera errors) . . . 149
5.32 Merging threshold 2D tests (with camera errors) . . . 150
5.33 Three views of the speaker . . . 152
5.34 Variations in appearance caused by lighting and reflection . . . 152
5.35 Camera placement for the speaker scene . . . 153
5.36 Point reconstruction of the speaker scene, before merging . . . 154
5.37 Point reconstruction of the speaker scene, after merging . . . 154
5.38 Measurements on the speaker scene . . . 155
5.39 Camera placement for the boat scene . . . 156
5.40 Details of the boat . . . 157
5.41 Examples of contour maps of the boat scene . . . 157
5.42 Views of the boat reconstruction taken from existing cameras . . . 158
5.43 Arbitrary views of the boat reconstruction . . . 159
5.44 Measurements on the boat scene . . . 159
5.45 Performance results for optimizing the number of lines rendered per buffer reading . . . 162

6.1 Flow chart of variant A: line reconstruction based on merged points . . . 169
6.2 Steps for connecting merged points . . . 170
6.3 Accumulation of votes on a candidate 3D line . . . 172
6.4 Flow chart of variant B: incremental line reconstruction . . . 174
6.5 Incremental line reconstruction . . . 175
6.6 Variant A: Minimum image count tests (reconstruction views) . . . 180
6.7 Variant A: Minimum image count tests (data charts) . . . 181
6.8 Variant A: Minimum average voting tests (reconstruction views) . . . 183
6.9 Variant A: Minimum average voting tests (data charts) . . . 184
6.10 Variant A: Maximum standard deviation tests (reconstruction views) . . . 186
6.11 Variant A: Maximum standard deviation tests (data charts) . . . 187
6.12 Variant B: Minimum average voting tests (reconstruction views) . . . 189
6.13 Variant B: Minimum average voting tests (data charts) . . . 190
6.14 Variant B: Maximum standard deviation tests (reconstruction views) . . . 192
6.15 Variant B: Maximum standard deviation tests (data charts) . . . 193
6.16 Line reconstruction: Arcade scene . . . 194
6.17 Line reconstruction: Bunny scene . . . 194
6.18 Line reconstruction: Carriage scene . . . 195
6.19 Line reconstruction: Chairs scene . . . 195
6.20 Line reconstruction: Solids scene . . . 196
6.21 Line reconstruction: Speaker scene . . . 196
6.22 Line reconstruction: Boat scene . . . 197
6.23 Lines crossing narrow surfaces . . . 198

A.1 CD menu . . . 210
A.2 Interactive reconstruction viewer . . . 212

B.1 Reconstructions of the speaker and boat scene . . . 215
B.2 Reconstructions of the synthetic scenes . . . 216
B.3 Additional scenes: clayjar and cube . . . 217
B.4 Additional scenes: dino, rose and knot . . . 218

List of Definitions

2.1 Internal camera parameters 13
2.2 External camera parameters 13
2.3 Image plane 14
2.4 Principal point of $\mathbf{C}^i$ 14
2.5 Clipped image plane corresponding to $\mathbf{C}^i$ 14
2.6 Depth of a w-point, relative to $\mathbf{C}^i$ 15
2.7 View ray from $\mathbf{C}^i$, going through $\mathbf{w}_q$ 15
2.8 Projected point (p-point) 15
2.9 World point defined by a p-point $\mathbf{p}_q^i$ and a depth d, relative to $\mathbf{C}^i$ 15
3.1 Distance to the closest point of interest of S_i 65
5.1 Real world point $\hat{\mathbf{w}}_q^i$ associated to a reconstructed point $\mathbf{r}_q^i$ 107
5.2 Reconstruction error of a point 107
5.3 Reconstruction error of a merged point 135

Chapter 1

Introduction

1.1 Computer vision and 3D reconstruction

The research field of *Computer Vision* (CV) is devoted to provide to computer systems the ability to accept and process visual input to perform a series of tasks, such as object detection and tracking, object or symbol recognition, or for solving other complex problems such as monitoring and surveillance, identity verification or quality control. Those complex, high-level tasks can be decomposed into less complex tasks such as image segmentation or motion estimation.

One important problem that draws the attention of computer vision researchers is *3D reconstruction*, which is in itself a complex task that can involve different computer vision techniques, and it can also be used as a tool for solving higher-level problems. Briefly stated, 3D reconstruction can be defined as the process of extracting 3D information regarding objects and scenes of the real world, based on inputs such as images, or data from laser or mechanical scanners. When based on images, it could be seen as the reverse of taking a picture. The availability of such a process should enable the automation of a series of tasks and applications, such as:

- Computing 3D measures for quality assessment, forensics or accident reconstruction, or determining distances or bounding boxes of objects, for e.g. collision avoidance;
- Providing 3D models such as virtual objects for:
  - educational or cultural purposes e.g. for virtual walk-through's on archaeological sites or exhibitions in virtual musea;
  - commercial purposes, such as publicizing products at an e-commerce website, or replication of objects (3D printing);
  - recreation purposes such as producing home-made 3D avatars representing a player in a game, or keeping "3D family pictures".
- Performing compression for low-bit-rate encoding and transmission of video recordings of a scene, by decomposing the scene or parts of it into textured

models, to improve the usage of low-bandwidth connections, or to empower 3D video conferencing.

- Producing content for 3D displays (which are gaining momentum) from existing 2D material.

This thesis focuses on the branch of 3D reconstruction methods using images captured from standard cameras as main inputs. In this introductory chapter, an overview of the principles and problems surrounding 3D reconstruction from images will be outlined, leading to the main motivations of this work.

1.2 The geometric principles of binocular visual systems

The basic geometric principle behind the 3D reconstruction methods focused here is *parallax*. Parallax is common to the low level human *binocular visual system*[1] and other biological visual systems. Consider two devices equipped with lenses, close to each other and aiming at similar directions (e.g. the eyes), which capture an image each. Due to the perspective effect introduced by each lens, there will be small differences – offsets – between the two images. These differences are the result of the *parallax* effect: images captured from two similar views present small differences related to the distance of the objects to the views, and the relative position and orientation of the cameras (Figure 1.1).

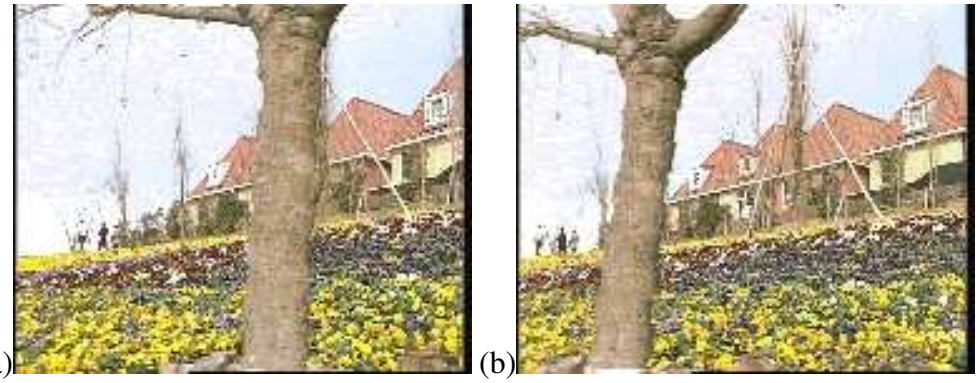


Figure 1.1: *Parallax effect.*
In this example, where the looking directions of the cameras are similar, distant objects (e.g. the houses) present smaller offsets than closer objects (e.g. the tree)

If a given point is identified in both images – point *correspondence* – and the relative position, orientation and lens properties of both imaging devices are known, it is

[1] The whole human visual system is much more complex, using much of the brain's acquired knowledge of the world. The discussion will however be focused on the assumption that only images and the eye's position and orientation information are available.

possible to apply a *triangulation* procedure to determine the position of that point in space, relative to the devices.

This is the principle explored by image-based 3D reconstruction methods. The devices used are cameras. Multiple cameras may be used (one for each position) or, alternatively, a single camera placed in multiple positions at different time instants.

Conversely, if sufficient and suitable correspondences are obtained between images, the information about the corresponding cameras' parameters can be determined from those correspondences. However, *the main problem in applying this principle in computer vision is the establishment of correspondences*.

In many cases, what a human intuitively identifies in two different images as corresponding to the same object may be represented digitally by very different 2D shapes due to perspective distortions, or by different color distributions, due to e.g. noise in the images or lighting effects.

The opposite may also happen: two parts of two distinct objects are represented by similar colors or shapes in two different images, introducing ambiguities.

The correspondence problem is detailed in the following section, together with related factors and issues that make the reconstruction task an interesting research problem.

1.3 Issues of image-based 3D reconstruction

The geometric visual process presented in the previous section is not trivial to implement in computer systems. There are multiple complications arising from the image acquisition and processing devices available and their limitations or specificities. Three major classes of possible issues are detailed in the following paragraphs: image acquisition, camera calibration, and correspondences between images.

1.3.1 Image acquisition

Consider a digital image capturing system that captures light from the world as digital images. Such a system may be composed of an analog camera (photo or video) that produces film which is later digitized into a computer system, or it may be an integrated digital camera (photo or video) that performs both the capturing and digitizing of the world images into digital images.

In any case, such a system will at least involve a *lens*, a set of *light sensors* that convert the light received through the lens to electronic signals, and an electronic circuit that processes the signals received by the sensors to produce a digital image.

In this path, a series of problems may occur:

- Camera lenses may introduce *distortions* in the light path that reaches the camera sensors or the film.

- The sensors (being in a digital camera or on a digital scanner) have limited *resolution* in terms of number of pixels and color range, meaning that only part of the information is used, and the remaining is lost.
- In the case of multiple overlapped sensors (e.g. red, green, blue sensors in digital cameras), there may be small misalignments that are reflected in the final image.
- The signal captured by those sensors is subject to *color interpolation*, *sampling* and *quantization* processes that introduce a number of artifacts (e.g. *aliasing*, color bleeding and others) in the final image.
- Magnetic or electrostatic interferences may also introduce *noise* in the final image (e.g. "grainy" images).

The influence of these problems in the final images is variable, leading to different distortions of an object in different images, and therefore introducing problems in object *correspondence* (described in Section 1.3.3).

1.3.2 Camera calibration

Knowing the cameras' position, orientation and internal lens parameters is an important part of the reconstruction process. This is known as *camera calibration*.

Camera calibration can be known or forced *a priori* e.g. by knowing detailed lens information from the manufacturer and by placing the camera in a fixed known position and orientation, or using a tracking device attached to the camera.

Camera calibration can also be estimated using images, being a research subject in itself. It has been a branch of research in the Computer Vision community for several years, and different methods exist nowadays to solve it, both automatically and semi-automatically [Hema 03][2]. Image-based camera calibration can rely on correspondences between images to establish a relative position between cameras, or placing calibration patterns to determine lens distortion. The use of calibration patterns can also facilitate the establishment of correspondences, providing an absolute referential in which the cameras are placed.

Camera calibration is still prone to errors, which are not trivial to model, and that affect the reconstruction e.g. when camera information is used to compute 3D positions.

Given the possibility of setting or obtaining camera parameters *a priori*, and the existence of automatic and semi-automatic camera calibration methods, it will be assumed throughout this work that the information about camera calibration is available.

1.3.3 Correspondence

The major reconstruction issues come from the need to identify the same object (or parts of an object) in different images – the *correspondence problem*.

[2]Which camera parameters are required may depend on the type of reconstruction or application.

In a computer-based reconstruction system, without *a priori* information of the objects of a scene, it is not generally possible to identify objects, let alone correspond objects on the images as a whole. Therefore, correspondence is usually established between lower-level image entities such as points, lines, or groups of pixels (blocks or segments).

Compound entities such as blocks or segments generally have the advantage of individually providing more disambiguating information (e.g. shape and color distribution) than simpler entities (e.g. points). On the other hand they are more sensitive to strong perspective effects. Larger entities may provide better coverage of the scene, but limit the resolution of the reconstruction (e.g. causing blocking artifacts), and may require the estimation of more parameters than 3D position (e.g. orientation) to be properly reconstructed.

Simpler entities such as points are much less sensitive (if at all) to strong perspective effects, and provide a finer grained element for the reconstruction. The number of parameters to estimate is smaller than for compound entities. Having less parameters to identify an entity also leads to a higher probability of two entities sharing the same properties, thus increasing ambiguity.

The choice of which entities are better suited to use in a reconstruction depends on the relations of these trade-offs with the application's goals and requirements. For instance, a given application may require a detailed reconstruction, which can be computed offline, while for another application an outline of the objects will suffice, but it has to be computed in real time.

Regardless of which entities are used, there is a series of factors that may change their appearance. This causes the same object (or parts of it) to be represented in two different images by different sets of pixels with different geometric arrangements, and different colors, with limited resolution (Figure 1.2).

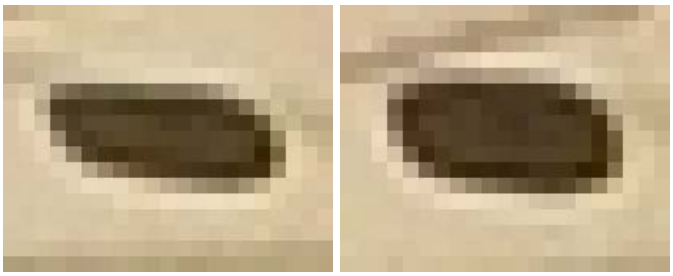


Figure 1.2: *Distortions and artefacts*.
Close-up of approximately the same region in two similar images. Notice the differences in shape and color distribution.

Besides lens distortions, noise, and other artifacts mentioned before, the main factors that contribute to those differences in appearance, and that consequently cause problems in correspondence, are the following:

Occlusion An object may be (partially) not visible in an image and visible (or with different visible parts) in another.

Perspective effects In general, if two cameras have significantly different positions and orientations, the projections of a given object will have different shapes in the corresponding images.

Reflections* and *refractions Reflective and refractive objects may have different appearance depending on the camera position. A common case of this are *highlights*, which are caused by the reflection of light on surfaces with high specularity. The position of a highlight relative to the surface varies depending on the camera's position, thus changing the objects' appearance.

Apart from these appearance-changing effects, there are other factors that affect the establishment of correspondences and the applicability of a triangulation procedure, such as:

Ambiguity It may happen that two (or more) parts of the scene are very similar and are represented as similar sets of pixels in two different images. A white wall is a good example: in this case, if there is no visible texture or discontinuity that may provide some parallax, even the human brain may lose depth perception. Another example is the case of *repetitive patterns*, such as checkerboards, tiles or bricks. This ambiguity may lead to incorrect correspondences and consequently to incorrect reconstructions.

Baseline The baseline is the line connecting two cameras. A baseline is considered small if the ratio between its length and the distance of the cameras to the objects is small. An important advantage of small baselines is that the differences from image to image tend to be very small, which facilitates correspondence. The influence of perspective effects between consecutive images is low, and the parts of objects that get visible or occluded are small (objects do not appear and disappear suddenly).

However, small baselines lead to *numerical instability* in triangulation, since even small errors in correspondence or camera calibration lead to large errors in the reconstruction. Thus, a large baseline provides better *conditioning* of the triangulation, but causes stronger perspective effects, making correspondence more difficult.

To take advantage of the best of these two worlds, it is possible to use *mixed-baseline* settings, where a large number of small baseline pairs can be found, but also large baseline pairs can be formed. A good example is a video sequence consisting of 50 frames taken with a moving camera, where the baseline between consecutive frames may be small, but the baseline between the first and the last frames is large. Correspondences could be propagated between consecutive frames from the first to the last frame, taking advantage of the small differences inherent to small baseline. Triangulation could then be performed using only the correspondence between the first and last frames, which have a larger baseline and are therefore more stable for triangulation.

The variety of inputs and the multiple sources of errors which may occur concurrently are the main problems for 3D reconstruction methods. The seemingly simple

solution of corresponding and triangulating is not directly applicable to real settings where those problems occur. This calls for robustness against the identified problems, or more specifically against their effects in the inputs (the images).

1.4 Motivation

The multiple problems that still limit image-based, view-independent 3D reconstruction, the inherent strive for robust methods and the interest in reaching interactive reconstruction rates to enable more powerful applications were the main motivations for the work presented in this thesis.

The goal was to create a method that would be robust to many of the problems that hamper current reconstruction techniques, while providing computational efficiency. This should then allow to perform 3D reconstruction robustly in interactive times.

1.4.1 Robustness

Robustness is defined in the context of a system as the degree to which the system can function correctly in the presence of invalid inputs or stressful environment conditions [IEEE 90].

In the context of this work, invalid inputs or stressful conditions can be seen respectively as error sources in the inputs (e.g. camera calibration errors, noise in the images, etc.) and less favorable yet correct conditions, such as a high number of occlusions or highlights.

A robust reconstruction method should still be able to reconstruct a scene in such settings, although some reduction in quality is to be expected. The level of robustness is inversely proportional to this quality loss that directly relates to the sensitivity to variations.

Quality can be judged at the level of the average error of the reconstruction, compared to the original scene. But for applications that include e.g. measuring or visualization of the reconstruction, it is also important to judge the number of incorrect estimates (correctness being defined by a threshold), since they are typically very noticeable in those applications.

In this sense, a robust method should, on one hand, provide estimates with an average low error in the presence of acceptable errors in the inputs. On the other hand, it should also be able to identify incorrect estimates – e.g. by determining a degree of confidence in the estimates. In case of more extreme variations, it should degrade gracefully i.e., as the quality of the inputs diminish, the average error and the number of outliers should not increase drastically.

Given the variety of error sources and the specificities of each reconstruction method, there is often a trade-off between robustness to different types of error sources. For instance, a reconstruction method based on the correspondence of rectangular blocks of an image will provide a degree of robustness to noise in the pixel colors

related to the size of the blocks (larger blocks provide more statistical stability); however, it will have less robustness to strong perspective effects in wide baseline settings, since the parts of an object that are covered by a large block will suffer significant perspective distortion, and will be more sensitive to occlusion.

Different methods present different robustness compromises. There are no definitive individual methods that completely solve 3D reconstruction in any settings. Rather, the methods are complementary to each other.

This thesis is aimed to present a better compromise in robustness for a wider range of error sources, with a lower number of restrictions, than those usually presented by other methods, by exploring the properties of features such as contour points and lines. Such a method would also be useful for providing robust initialization data for e.g. methods that require a good initial approximation of a scene's reconstruction.

1.4.2 Interactivity and usability

The input data fed to a 3D reconstruction process may be inappropriate or insufficient for an accurate or complete reconstruction. Furthermore, such flaws are not straightforward to identify directly from the input data, meaning that the user may only realize that more information is required after actually performing an initial reconstruction. In these cases, it is highly beneficial to be able to promptly get feedback from the reconstruction process.

For example, in the case of image-based 3D reconstruction, there may be parts of the scene that are not visible in a sufficient number of images to allow the computation of 3D coordinates for every element in the scene. If a user can quickly reconstruct a scene on-the-spot and finds out that more images are required for a better reconstruction, he can just capture some more images without leaving the spot or having to prepare the scene again. The reconstruction software might even indicate from which positions more images should be taken. If the reconstruction apparatus has physical control over the cameras or scene (e.g. through a robot arm holding a camera, or a turntable containing the objects) it could automatically adapt the system to acquire the missing images.

Another case where quick turnaround times are of importance is parameter tunning in user-assisted reconstruction. In many instances, it is possible to improve reconstruction results by incrementally tunning the reconstruction method's parameters.

In the above situations, the ability to quickly obtain reconstruction feedback, or in other words to *interact* with the reconstruction software, influences its *usability*.

In the context of this work, reconstruction methods are considered *interactive* if they provide results to the user in a short time period – for instance, getting previews in the order of a few seconds, at most.

1.4.3 Trade-off between robustness and performance

In reconstruction, there is a typical trade-off between robustness and performance. When strong assumptions and constraints are applied (e.g. the assumption of a regularly translating camera), the problem becomes more conditioned and simplified, and less complex methods can be used, thus being possible to achieve good performance. However, such methods will be limited in terms of applicability. On the other hand, if a method is to cope with more general assumptions and to be robust to errors and input variations, it is in general bound to be more complex and thus computationally more demanding.

An example of this duality are *view-dependent* and *view-independent* methods. A view-dependent reconstruction method is typically oriented toward visualization, with known and limited final viewpoints. This constrains and therefore simplifies the reconstruction process. In some cases, it is not necessary to actually reconstruct, but just to synthesize a new image, without explicitly extracting 3D information (e.g. the Lumigraph [Gort 96]). The process is even more simplified if the input viewpoints are close to the final ones. The reduction in the problem's complexity is reflected in a reduction of computational and algorithmic complexity. Some view-dependent reconstruction methods (e.g. [Yang 02]) can already achieve real-time performance using either dedicated hardware, parallel processing in multiple computers, reduced resolutions or combinations of those factors.

On the other hand, view-independent reconstruction presents a much more challenging problem: the final result of the reconstruction must be a 3D representation that can be visualized from an arbitrary viewpoint. This implies that the reconstruction has to be as complete as possible, and that there is no *a priori* information of the final viewpoints to constrain or simplify the problem. For this reason, typical view-independent methods are inherently more complex, and consequently, their performance levels for high quality, unconstrained reconstructions are still considerably lower than the ones of view-dependent methods.

This thesis aims to present an implementation of a robust method that has a good compromise with efficiency, such that interactive reconstruction rates can be achieved, without compromising robustness.

1.5 Thesis structure

This thesis is organized as follows. Chapter 2 presents a review of existing methods developed in the field of 3D reconstruction, and a classification in terms of their properties, as well as assumptions and requirements.

Based on the shortcomings identified in existing methods, a new point reconstruction method is proposed in Chapter 3. The core of the work consists of a view-independent *point reconstruction* method, based on sets of images (on the order of tens of images), captured from arbitrary viewpoints, and assuming known or *a priori* estimated camera calibration. The core method provides basic inherent robustness due to its integrating nature, since all images contribute simultaneously to estimate each

point, thus providing confidence in the estimation. Afterward, two additional steps for increasing robustness are presented, which identify and eliminate the majority of initial incorrect estimates. The result is a set of reconstructed points that accurately covers the scene's objects.

An implementation of the method exploiting standard *graphics hardware* is presented in Chapter 4. Tests were performed on synthetic and real scenes, to assess the quality, robustness and performance of the proposed method and its implementation. The results are reported on Chapter 5, and show that it is possible to achieve interactive and robust view-independent point reconstructions even in the presence of occlusions, highlights, and less-than-perfect camera calibrations.

In addition, a line reconstruction method with two variants was developed, based on the point reconstruction method, and it is presented in Chapter 6. The line reconstruction uses the reconstructed points as part of its inputs, serving as an example of integration of different types of reconstruction. On the other hand, it shares concepts, inputs and by-products of point reconstruction, thus sharing also some of the robustness benefits.

Finally, conclusions are drawn in Chapter 7, completed with possible lines of future work.

Reconstruction results of all the tests presented in Chapter 5 and Chapter 6 and additional scenes can be found in the CD accompanying this thesis. The details on the contents and usage instructions can be found in Appendix A. Some images of the additional reconstructions contained on the CD are presented in Appendix B.

Chapter 2

Background

This chapter is devoted to the basic concepts and existing research in the area of 3D reconstruction from images.

The first part introduces the main principles of reconstruction, including some notation used throughout the thesis. It is at first assumed that the scenes being reconstructed and the inputs are ideal i.e., there is no noise in the images, camera calibration is perfect, there are no reflections, refractions or highlights, neither occlusion nor movement. This provides a common ground for many reconstruction methods.

As camera errors, noise and other factors come into play in real world scenes, different ways of dealing with them give rise to different approaches. The second part of this chapter comprises an overview of existing work in the area, complemented with an analysis of the strong and weak points of the various methods. At the end of the chapter, this analysis will be used to point out possible directions of research, some of which lead to the methods proposed in this thesis.

2.1 Geometric principles of image-based reconstruction

As the name suggests, the purpose of image-based 3D reconstruction is to use a set of two-dimensional images that were created from a three-dimensional world, and rebuild that original three dimensional information, or at least part of it. To achieve this goal, it is important to analyze how images are created in the first place, how 3D information is lost, and which cues and hints may be explored from the remaining data.

This section introduces the principles of image generation using cameras, and it explains how the geometric properties behind these principles can be explored to recover 3D information. Both the scene objects and cameras are placed on an Euclidean three-dimensional *world space* $W = \mathbb{R}^3$. It is assumed that camera information, such as its position in W and other camera parameters (discussed in the next section), are either known *a priori* or estimated in the process[1]. Furthermore, it is assumed that

[1]Even methods that do camera calibration and 3D reconstruction simultaneously [Torr 00] estimate

the scene is static, i.e. there are no moving objects or lighting variations, and that the objects are opaque.

2.1.1 Cameras

Each view of a scene is recorded by a *camera*. Cameras can be geometrically modeled to describe the process of generating an image of a scene. Most cameras have lenses that introduce distortions, leading to more complex geometric models. A detailed overview of common camera models can be found in [Fors 03].

The simplest camera model – the *pinhole camera*, also known as *projective camera* – is lens-less. If the geometric model of a camera with lenses is known, it is possible to use a pinhole camera model that approximates the original camera, by transforming the images acquired by the original camera. For this reason, the pinhole camera model will be used throughout this work.

The original pinhole camera consists of a dark box (*camera obscura*) with a very small hole (the *pinhole*) in one of the faces. Light coming from outside the box traverses the hole and projects on the inner face of the box opposite to the hole. This creates an inverted image of the outside world (Figure 2.1). The scale of the image projected on the face opposite to the hole is directly proportional to the distance between the hole and the face where the image is created.

The pinhole is the *camera center* (also known as *focal point* or *optical center*), the face where the image is projected, a rectangular surface of limited width w and height h, is the *image plane* (or *projection plane*), and the distance from the hole to the face is the *focal length* (or *focal distance*) (see Figure 2.1).

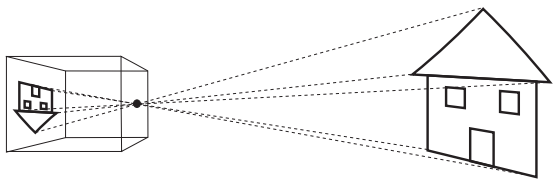


Figure 2.1: *A pinhole camera.*

To make the model more intuitive, and without loss of generality, it is commonly considered that the image plane is placed in front of the camera center, and not behind. Due to this shift of the image plane, the projected image is not inverted, as opposed to the original pinhole camera.

The set of *internal camera parameters* (or *camera intrinsics*) of a camera[2] $\mathbf{C}^i$ under this lens-less model will consist of focal length f^i, the width w^i and height h^i (in world units) of the projection surface[3] (see Figure 2.2 and Definition 2.1).

at least some camera parameters from correspondences.

[2]Since multiple cameras will be considered for a scene, cameras and camera-dependent entities are related by a superscript.

[3]In more complex camera models, other parameters are usually considered, such as principal point

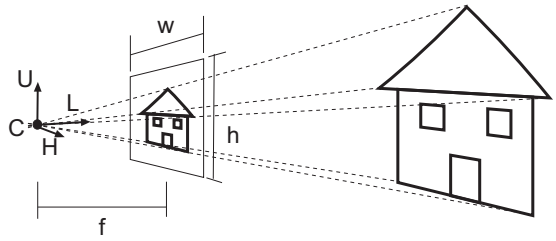


Figure 2.2: *The pinhole camera model and associated geometry*.

Definition 2.1 *Internal camera parameters*

$$\mathbf{I}^i = \langle f^i, w^i, h^i \rangle$$

□

The *external camera parameters* (or *camera extrinsics*) define the position of the camera center and the orientation of the camera in the world. The camera center's position $\mathbf{c}^i$ is commonly represented by its 3D coordinates in W. The orientation can be represented in different ways. A common representation is to represent orientation using a set of three *orthogonal unit vectors* defining the horizontal, vertical (or up) and look directions of the camera[4] ($\mathbf{H}^i$, $\mathbf{U}^i$, $\mathbf{L}^i$), expressed in world space (Definition 2.2). Another common option is to use the set of three angles representing axial rotations of the world referential to transform it to the camera referential (e.g. mapping $\mathbf{X}$ to $\mathbf{H}$, $\mathbf{Y}$ to $\mathbf{U}$ and $\mathbf{Z}$ to $\mathbf{L}$). The three-vector notation is more practical for the purposes of this work, and therefore it will be the one used to represent the camera orientation.

Definition 2.2 *External camera parameters*

$$\mathbf{E}^i = \langle \mathbf{c}^i, \mathbf{H}^i, \mathbf{U}^i, \mathbf{L}^i \rangle$$

□

The image plane π^i is the plane normal to $\mathbf{L}^i$, located at a distance f^i of the camera center $\mathbf{c}^i$ (Definition 2.3). The *principal point* $\mathbf{m}^i$ is the projection of $\mathbf{c}^i$ in π^i (Definition 2.4).

Only a rectangular region of dimensions w^i by h^i of the image plane is used to record information. This subset will be called the *clipped image plane* $\widehat{\pi}^i$. It will be assumed that the center of $\widehat{\pi}^i$ is coincident with the principal point $\mathbf{m}^i$, without loss of generality[5]. In this case, the clipped image plane is defined according to Definition 2.5.

offset, radial distortion, skew and ratio.

[4]Since the vectors are orthogonal, two vectors would be sufficient to define the camera, the third one being defined as their cross-product. In the context of e.g. computer graphics it is common to use the up and look vectors.

[5]If the principal point is not aligned, a 2D translation can be used to account for it.

Definition 2.3 *Image plane*

$$\pi^i = \left\{ \mathbf{c}^i + f^i \mathbf{L}^i + x\mathbf{H}^i + y\mathbf{U}^i \middle| \ -\infty < x < \infty \wedge \ -\infty < y < \infty \right\}$$

□

Definition 2.4 *Principal point of* $\mathbf{C}^i$

$$\mathbf{m}^i = \mathbf{c}^i + f^i \mathbf{L}^i$$

□

Definition 2.5 *Clipped image plane corresponding to* $\mathbf{C}^i$

$$\widehat{\pi}^i = \left\{ \mathbf{c}^i + f^i \mathbf{L}^i + x\mathbf{H}^i + y\mathbf{U}^i \middle| \ -\frac{w^i}{2} < x < \frac{w^i}{2} \wedge \ -\frac{h^i}{2} < y < \frac{h^i}{2} \right\}$$

□

The *frustum* of a camera is the volume of the world that projects in the clipped image plane i.e., and is limited by a far plane, parallel to $\widehat{\pi}^i$, as seen in Figure 2.3 (the far plane may be at infinity).

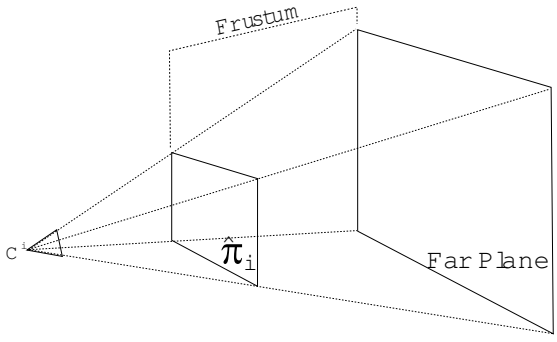


Figure 2.3: *The frustum of a camera*.The far plane may be at infinity

2.1.2 Point projection and image generation

Consider a *world point* (*w-point*) $\mathbf{w}_q$ with coordinates $\langle x, y, z \rangle$ in W, that is on the surface of an object of the scene. Consider that $\mathbf{w}_q$ is in front of $\mathbf{C}^i$ and projects in $\widehat{\pi}^i$.

The *depth* of $\mathbf{w}_q$ relative to $\mathbf{C}^i$ will be defined as the euclidean distance between $\mathbf{w}_q$ and the camera center $\mathbf{c}^i$ (Definition 2.6). Depth is sometimes defined in the literature as the distance *projected* in the direction of $\mathbf{L}^i$, which has advantages in some settings. That is not the case in this work, where Definition 2.6 is more convenient, and will therefore be preferred.

The half-line starting at $\mathbf{c}^i$ and going through $\mathbf{w}_q$ is the *view ray* $V^i_{\mathbf{w}_q}$ (Definition 2.7).

The view ray $V^i_{\mathbf{w}_q}$ intersects the camera's clipped image plane $\widehat{\pi}^i$ in a *projected point* (*p-point*) p-point $\mathbf{p}^i_q$ (Definition 2.8).

Definition 2.6 *Depth of a w-point, relative to* $\mathbf{C}^i$

$$d^i(\mathbf{w}_q) = \left\|\mathbf{w}_q - \mathbf{c}^i\right\|$$

□

Definition 2.7 *View ray from* $\mathbf{C}^i$*, going through* $\mathbf{w}_q$

$$V^i_{\mathbf{w}_q} = \left\{ \mathbf{c}^i + k\frac{\mathbf{w}_q - \mathbf{c}^i}{\|\mathbf{w}_q - \mathbf{c}^i\|} |\ k > 0 \right\}$$

□

Definition 2.8 *Projected point (p-point)*

$$\mathbf{p}^i_q = V^i_{\mathbf{w}_q} \cap \widehat{\pi}^i$$

□

The view ray $V^i_{\mathbf{p}^i_q}$ corresponding to $\mathbf{p}^i_q$ is coincident to $V^i_{\mathbf{w}_q}$. Therefore, a world point can be uniquely defined from a camera $\mathbf{C}^i$, a p-point $\mathbf{p}^i_q$ and a depth d, by determining the point on $V^i_{\mathbf{p}^i_q}$ that lies at a distance d of the center of $\mathbf{C}^i$, as shown in Definition 2.9.

Definition 2.9 *World point defined by a p-point* $\mathbf{p}^i_q$ *and a depth* d*, relative to* $\mathbf{C}^i$

$$\mathbf{w}_q = \mathbf{c}^i + d\frac{\mathbf{p}^i_q - \mathbf{c}^i}{\left\|\mathbf{p}^i_q - \mathbf{c}^i\right\|}$$

□

If m w-points $\mathbf{w}_{q_1}$, ..., $\mathbf{w}_{q_m}$ are aligned with a view ray V^i_q, the w-point closest to the image plane occludes the others[6].

An image consists of a representation of the p-points generated by the projection on the image plane of the non-occluded w-points of the scene. For each view ray associated with a p-point, only the w-point that is closest to the camera is represented in the image, *occluding* the other w-points on that same ray (Figure 2.4). Due to limitations on the systems that are used to capture, store and process images, these points are represented in a discrete frid, and each point is typically represented in the form of a color or luminance discrete value.

[6]Recall that objects are considered opaque.

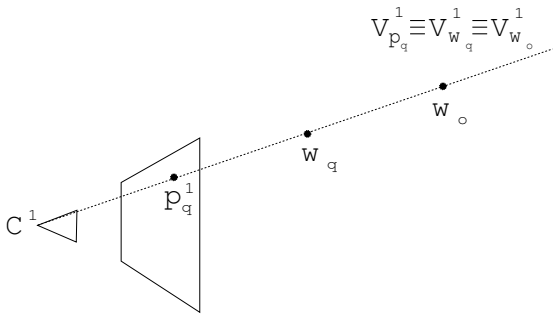


Figure 2.4: *Point projection and occlusion.*
$\mathbf{p}_q$ is the projection of $\mathbf{w}_q$, since $\mathbf{w}_q$ is in front of $\mathbf{w}_o$

2.1.3 Reconstruction from multiple views

The coordinates of an unknown world point $\mathbf{w}_q$ may be estimated given at least two images, the parameters of the cameras that recorded them and correspondences between the projections of that point in the images.

With one single view with a projection $\mathbf{p}_q^1$, one can only infer that $\mathbf{w}_o$ is located somewhere in the view ray of its projection (Figure 2.5(a)):

$$\mathbf{r}_q \in V_q^1. \tag{2.1}$$

If two projections of $\mathbf{w}_q$, namely $\mathbf{p}_q^1$ and $\mathbf{p}_q^2$, in I^1 and I^2 are known (and assuming the ideal case of no discretization, exact camera calibration and no errors in the acquisition process), the geometric position of $\mathbf{r}_q$ corresponds to the intersection of the two view rays, as in Equation 2.2 (Figure 2.5(b)).

$$\mathbf{r}_q \in V_q^1 \wedge \mathbf{r}_q \in V_q^2. \tag{2.2}$$

There are two particular cases worth noting, that correspond to situations where the view rays may have the same direction. One is when the w-point is at infinite depth from the cameras: in this case, the view rays do not intersect. In practice, such points are not considered relevant for reconstruction purposes. The other situation occurs when the camera centers and the point are collinear, such that the view rays overlap, resulting in multiple solutions for Equation 2.2 (Figure 2.6).

In an ideal situation, and assuming no discretization, all the view rays corresponding to projections of the same point in multiple images intersect at the correct world point (Figure 2.5(c)). However, in real situations, errors may cause that the view rays associated to the established correspondences do not intersect (Figure 2.5(d)), even if the correspondences are correct. For this reason, the geometric principles cannot be applied directly in real situations, and reconstruction methods strive to provide tolerance and robustness.

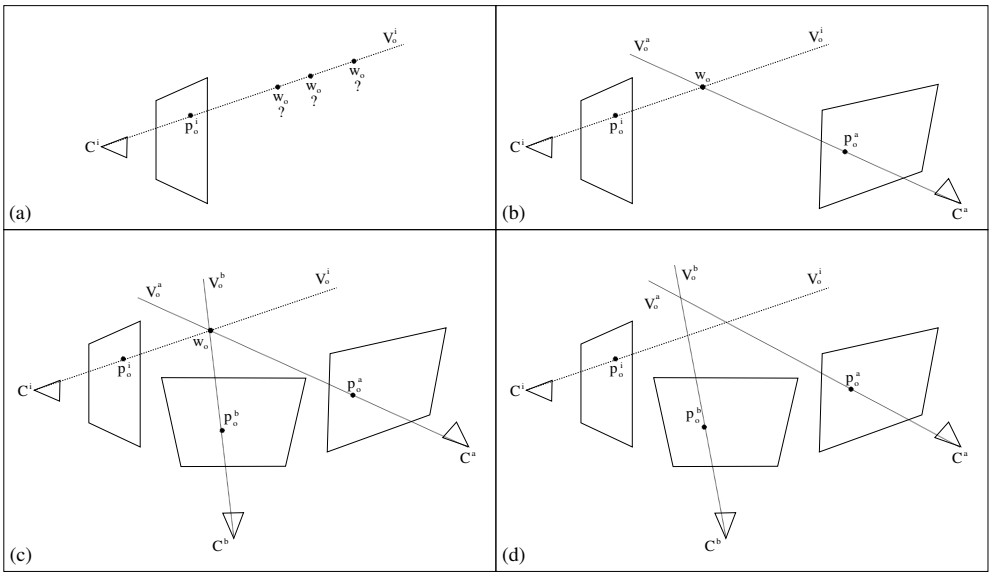


Figure 2.5: *Influence of the number of views on reconstruction*.

(a) One view: the position of the w-point is ambiguous

(b) Two views: the w-point coincides with the intersection of the two view rays (assuming no errors)

(c) Three (or more) views: the view rays should redundantly intersect in the same point (assuming no errors)

(d) In the presence of errors, the view rays may not intersect (this may happen with any non-singular set of views)

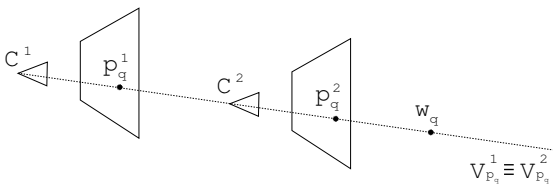


Figure 2.6: *Overlapping viewrays*.

View rays overlapping in the case of a w-point collinear with camera centers

2.1.4 Correspondence

So far, it was assumed that the correspondence between two p-points $\mathbf{p}_q^1$ and $\mathbf{p}_q^2$ representing the same (unknown) w-point $\mathbf{w}_q$ was known. However, finding correspondences between points, parts of images or objects is a major problem faced by 3D reconstruction and other branches of computer vision, as referred in Section 1.3.3.

Consider two images I^1 and I^2, and two p-points $\mathbf{p}_q^1$ and $\mathbf{p}_s^2$ representing the same (unknown) w-point $\mathbf{w}_q$ in each image (meaning that $\mathbf{w}_q$ is not occluded in any of the images). The basic information available on $\mathbf{p}_q^1$ and $\mathbf{p}_s^2$ are their attributes in the images, such as their color and their position. If these points represent the same $\mathbf{w}_q$, their attributes are expected to be similar or related (e.g. similar colors). However, it is

not safe to assume that a pair of p-points, one from each image, with related attributes represent the same w-point. Such similarity is only a possible indicator that such pair *may* correspond to the same w-point.

The measures of attribute similarity between two p-points may be found in the literature under different names, such as *correspondence measure*, *similarity measure* or *match function*. Their duals are also common: instead of a measure of similarity, a measure indicating how dissimilar two p-points are is used – a *correspondence error*, *match penalty*, or *error function*. Without loss of generality, the discussion will be based on match functions, keeping in mind that the dual is just as valid.

A match function provides a *matching value* for a pair of points. Two points that are similar will have a high matching value, and two points that are different will have a low matching value. There are several possible match functions. For instance, if the attributes quantify luminance, the match function could be the similarity of luminance values between the two p-points. Or, in the case of RGB color attributes, the inverse value of the color distance between pairs of p-points in a color cube could be used.

With a match function, one can, given two points from different images, quantify how good is their correspondence. To determine the best correspondence for a point $\mathbf{p}_q^1$ in image I^1, one could compare it with all points of I^2, by computing their match value. Finding the best correspondence would thus consist of *maximizing the match function* of $\mathbf{p}_q^1$ over all points of I^2.

Matching a point from an image with all the points in another image is not only computationally expensive, but also can lead to a high number of ambiguities, as it is possible that there are multiple points that provide high match values (e.g. many points with similar colors, or repeating patterns) – reflected by the existence of multiple high-valued local maxima in the matching function.

The availability of camera information provides geometric constraints that can be applied to limit the search space to a smaller region of the second image. Such constraints help eliminating a significant number of false matches, resulting on a *robustness increase*. They also reduce the *computational load* required to find the best correspondence.

One of the most important constraints is based on *epipolar geometry*, and it will now be described.

Epipolar geometry for two-image point reconstruction

Consider two images I^1 and I^2, with known cameras $\mathbf{C}^1$ and $\mathbf{C}^2$, and a point $\mathbf{p}_q^1$ of I^1. Let V_q^1 be the *view ray* of $\mathbf{p}_q^1$ associated to $\mathbf{C}^1$ (the half-line starting in the centre of $\mathbf{C}^1$ and going through $\mathbf{p}_q^1$). The *epipolar line* $E_q^{1,2}$ is the projection of V_q^1 in I^2. Points $\mathbf{v}_{q,x}^1$ belonging to V_q^1 are projected in $E_q^{1,2}$, and points $\mathbf{p}_{q,y}^{1,2}$ of $E_q^{1,2}$ can be projected back to V_q^1, using $\mathbf{C}^2$ (Figure 2.7).

In terms of correspondence search, this means that the correct correspondence for $\mathbf{p}_q^1$ in I^2 has to be in $E_q^{1,2}$ (assuming exact camera calibration). This is called the *epipolar constraint*.

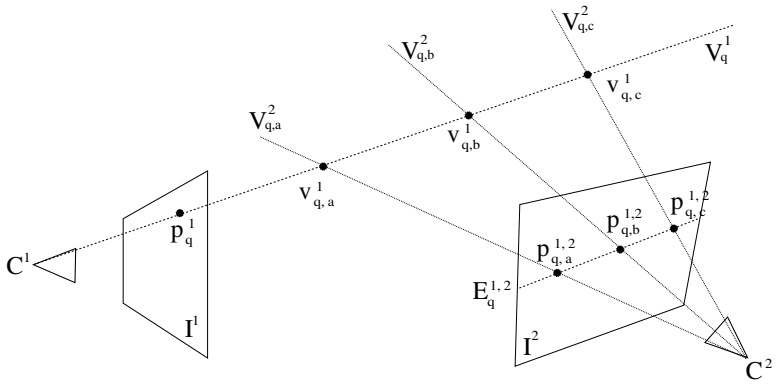


Figure 2.7: *Epipolar geometry*.

Assume there is a *match function* $\sigma(\mathbf{p}_q^1, \mathbf{p}_{q,x}^{1,2})$ that, when applied to two points $\mathbf{p}_q^1$ and $\mathbf{p}_{q,x}^{1,2}$, quantifies the similarity between them. In this case, determining the best correspondence of $\mathbf{p}_q^1$ amounts to maximize $\sigma(\mathbf{p}_q^1, \mathbf{p}_{q,x}^{1,2})$ over a set of candidates $\{\mathbf{p}_{q,a}^{1,2}, ..\mathbf{p}_{q,n}^{1,2}\}$. In the two-image setting, this could be done by evaluating the match function for the points belonging to the epipolar line $E_q^{1,2}$. The generic algorithm is presented in Algorithm 2.1.

- Let V_q^1 be the view ray of $\mathbf{p}_p^1$ associated to $\mathbf{C}^1$
- Project V_q^1 in I^2 as the epipolar line $E_q^{1,2}$
- Select a set of image points $\mathbf{p}_{q,a}^{1,2}$, $\mathbf{p}_{q,b}^{1,2}$, ... from $E_q^{1,2}$ as the candidates
- The correspondence $\mathbf{p}_{q,\overline{x}}^{1,2}$ for $\mathbf{p}_q^1$ is the candidate with higher match value $\sigma(\mathbf{p}_q^1, \mathbf{p}_{q,x}^{1,2})$
- The reconstructed point (r-point) $\mathbf{r}_q^1$ corresponding to $\mathbf{p}_q^1$ is computed as the intersection of the view rays V_q^1 and $V_{q,\overline{x}}^2$

Algorithm 2.1: Reconstruction using epipolar line candidates

The occurrence of impairing factors such as noise or occlusion may cause the right candidate for correspondence in the second image to be missed, resulting in an incorrect reconstruction of $\mathbf{p}_q^i$, or even no correspondence.

Camera calibration errors are likely to occur as well, resulting on incorrect epipolar constraints. For this reason, epipolar constraints should not be strictly imposed.

This uncertainty adds up to the ambiguity in the correspondence itself, and lead to the development of different reconstruction methods. Some of the most relevant methods are discussed in the following sections.

2.2 Related work

2.2.1 Introduction

Given the problem of correspondence, different reconstruction techniques have been developed to find innovative and better ways of solving the problems of reconstruction, namely robustness in correspondence.

Reconstruction techniques can be classified in different ways. Figure 2.8 depicts a possible classification tree for reconstruction methods based on the number of images (or views) used for the reconstruction. For completeness, methods that use one single image are included, but the focus will be set on the methods that use two or more views, more precisely generic multi-view methods.

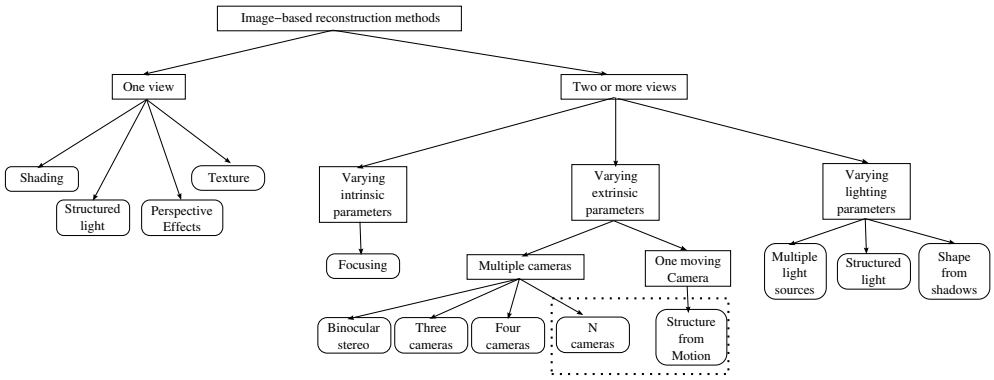


Figure 2.8: *A classification tree of image-based reconstruction methods, based on number of input images*.
The dotted box represents generic multi-view methods.

Methods using one single image typically exploit two kinds of *depth cues*: light-based or model-based. Examples of methods using light cues are *shape from shading* [Zhan 99] and *structured lighting* techniques [Scha 03] (see Figure 2.9 and Figure 2.10, respectively).

Model-based methods assume some *a priori* knowledge of a model of the scene or the objects, such as the existence of 3D horizontal lines, rectangular surfaces, or objects with a regular texture. That information is used as a cue for depth, e.g. on techniques based on *perspective effects* [Scha 00, Yang 05, Se 03] or based on *texture analysis* [Cler 02, Witk 81, Kana 89] (see Figure 2.11).

With more than one image as input, depth cues arise from variations on camera or lighting parameters. In this class, a distinction is made for the cases in which, between two images:

- the intrinsic camera parameters change - cue: focus (e.g. *shape from focus* [Asad 98], Figure 2.12);

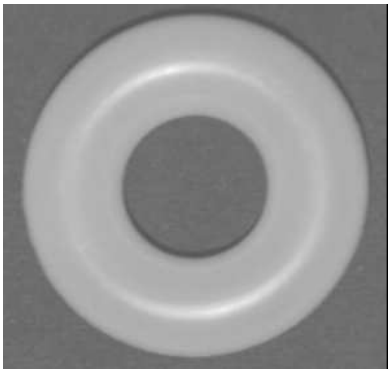
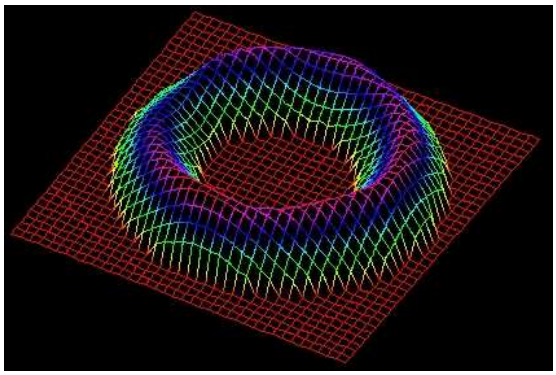
(a)  (b) 

Figure 2.9: *Shape from shading*.

(a) The original image

(b) The recovered 3D structure

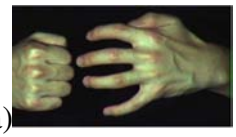
(a)  (b)  (c)  (d) 

Figure 2.10: *Structured light*.

(a) One view of a scene

(b) A known and easily identifiable light pattern

(c) The pattern projected on the scene

(d) The model extracted by analyzing the pattern deformation

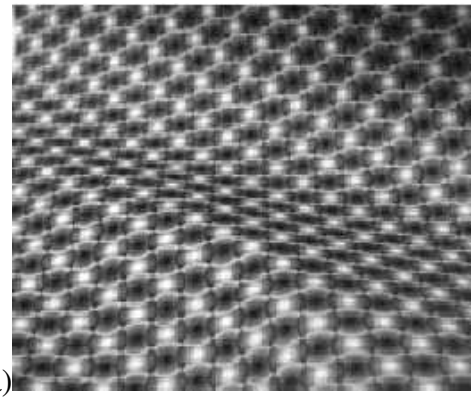
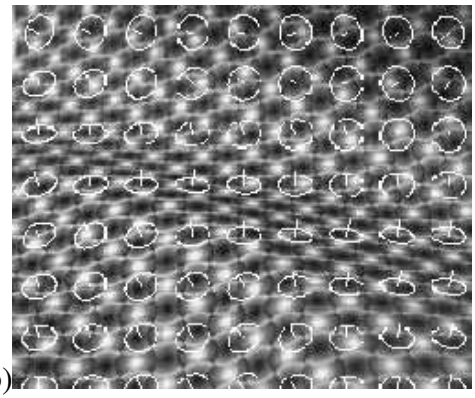
(a)  (b) 

Figure 2.11: *Shape from texture*.

The deformation of a regular texture provides a cue for the surface shape

(a) an example surface

(b) the extracted texture elements and surface orientation

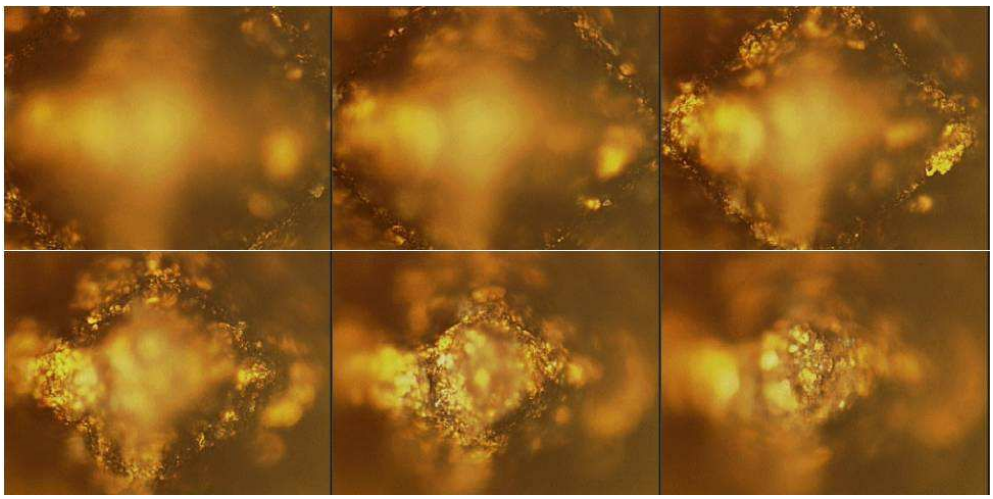


Figure 2.12: *Shape from focus*.
Six images taken with the same camera external parameters, with varying (known) focal length. Notice the difference in the sharp areas from image to image



Figure 2.13: *Structure from motion*.
Four non-consecutive images selected from a video sequence. The parallax effect can be noticed on the large apparent displacement of the tree from image to image, versus the small displacement of the houses, which are more distant of the camera

- the extrinsic camera parameters change - cue: parallax (e.g. *structure from motion*, Figure 2.13);

- the lighting parameters change - cue: light pattern distortion/position (e.g. *structured lighting*, *shape from shadows*).

From these cases, the parallax is the one that presents itself as more interesting: it is not limited to a single viewpoint position, as usually shape from focus techniques are, and it is not as intrusive as the case of structured lighting. This thesis focuses on the case of *techniques that explore the parallax originated by varying extrinsic camera parameters*.

Within this class, the specific cases of two images (binocular stereo), three images and four images will be distinguished from the general case of multiple-image reconstruction. There are specific geometric properties of each of those settings that can be exploited, namely *epipolar geometry*, the *trilinear tensor* (or *trifocal tensor*) and the *quadrifocal tensor*.

After an overview of these particular cases, the focus will turn to the general case of multi-view techniques.

2.2.2 Fixed number of views

Binocular stereo

Most of the initial research carried out in the field of 3D reconstruction has been based on the stereo case. Binocular stereo uses the smallest set of images that allows for parallax-based reconstruction, and it is related to known organic visual systems such as the human visual system.

A large number of stereo techniques exist, differing on the assumed set-ups (e.g. camera calibration known *a priori* or estimated), matching methods and entities, and the target applications (e.g. model generation, image synthesis, video-conferencing).

Stereo techniques are based on the geometric framework described in Section 2.1, and many of them make use of the epipolar constraint to simplify matching. This constraint can be represented in the form of a matrix – the *fundamental matrix* – that is used to describe the geometry between two views (more information can be found in [Hart 00]).

Classic approaches to correspondence estimation in image pairs include correlation-based algorithms [Scha 02] and feature-based algorithms [Faug 93]. The most important properties of these two classes of approaches will be presented now, since they are also widely used in multi-view techniques. For a more detailed review, the reader is referred to [Rede 99].

Correlation-based methods

Correlation-based methods (also known as *area-based methods* or *region-based methods*) perform matching by maximizing a similarity measure between a small *window* (or *block*) on a reference image and another window on a second image (Figure 2.14). A typical similarity measure is the sum of squared differences (SSD) of the intensities of the windows' pixels.

The window pair with the highest correlation is a match. If the camera parameters of both images are known, the depth corresponding to a match can be estimated by triangulation. Depth values can then be assigned equally to all pixels in the window, interpolated over the window, or assigned only to a representative pixel in the window (e.g. the center pixel).

A number of variants and extensions exist, starting with varying shapes and sizes of the windows, alternative ways of choosing the candidates (the use of the epipolar constraint is common), and extending to more refined enhancements, such as using adaptive windows [Kana 94]. Scharstein and Szeliski [Scha 02] proposed a taxonomy and evaluation of dense binocular stereo algorithms, with a comparison of several algorithms, and an extensive list of references in the area.

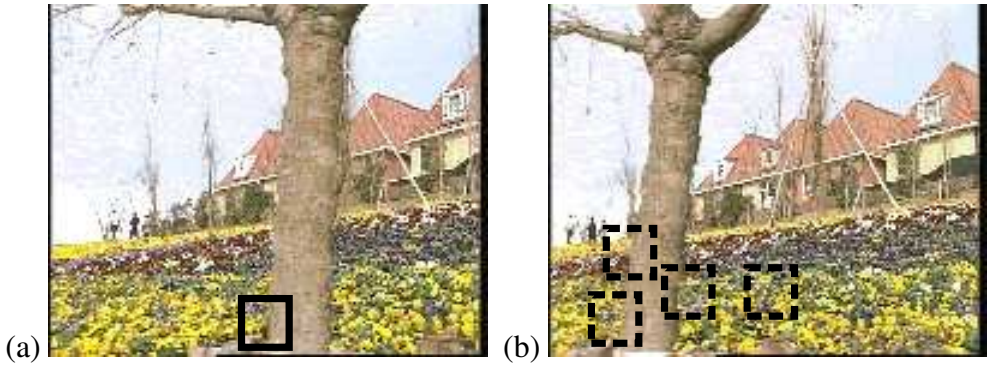


Figure 2.14: *Example of correlation-based correspondence estimation.*
The correspondence for the block marked in (a) is searched in (b)

Since correspondence in these techniques is based on the correlation of sets of pixels, textured areas contain a high quantity of information that helps to establish correspondences in the general case. However, problems occur when repetitive patterns are present in the scene (e.g. a checkerboard): the various occurrences of the pattern will have similar correlation, thus causing ambiguity in correspondence. The same problem occurs in situations where there is low texture, such as a clear sky or a white wall.

Correlation-based methods are able to provide a dense depth map, by attributing depth values to each pixel of an image. When the resulting depth value is either attributed to all the pixels in the block, or used to interpolate values for the pixels of the block, some disadvantages arise. On one hand, it gives rise to *blocking artifacts* in the resulting reconstruction – step-like surfaces or disconnected surfaces due to different depth values attributed to adjacent blocks of an image. It is common to apply some sort of regularization to reduce these artifacts, as well as to smooth out outliers caused by incorrect correspondence estimation. On the other hand, when rectangular or other fixed-shape types of blocks are used, the borders of those blocks will be very likely misaligned with real depth discontinuities of the scene. This may result in real discontinuities being missed or displaced in the reconstruction.

The dimensions of the windows used for correspondence estimation have a strong influence in these methods, as referred in [Scha 96]. A small window reduces the problem of blocking artifacts and misaligned discontinuities, but it increases the probability of wrong matches due to ambiguity (not enough texture) or noise in the image. A window too large not only increases the blocking and misalignment, but it is also more difficult to match if there are strong perspective distortions or occlusion, resulting in lost detail and blurring of object boundaries.

Methods that assign depth to a single pixel, while still estimating correspondences by comparing blocks, may achieve less blocking artifacts. Nevertheless the issues related to ambiguity in small blocks and perspective distortions and occlusions in larger blocks may still be significant. Furthermore, the computational load of such methods can be substantially higher than the one for methods that assign depth to blocks of

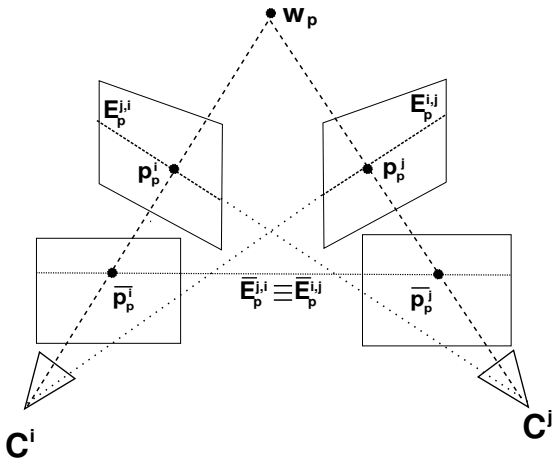


Figure 2.15: *Image rectification*.
A pair of images taken from known viewpoints $\mathbf{C}^i$ and $\mathbf{C}^j$, can be rectified by projecting them to a single plane, such that their epipolar lines become parallel and aligned (typically with their horizontal scan lines aligned)

pixels (in the proportion of the number of pixels per block). These methods are also referred in [Scha 02].

A strong disadvantage of correlation-based methods is their limitation in terms of sensitivity to camera baseline. As mentioned before, perspective effects can make it very difficult to match windows. Many correlation techniques employ rectification (see Figure 2.15), to reduce the distortion caused by perspective inside windows. However, this is only effective in cases where the differences in camera parameters is small, and does not help in case of occlusions. Therefore, most correlation-based techniques are limited to cameras with similar orientation and position, resulting in a limited coverage of a scene.

Feature-based methods

Feature-based methods attempt to find correspondences between image features, such as corner points or contour lines. These methods often quantify symbolic properties of features into a similarity metric that is then used for matching.

Features represent entities with specific properties that stand out (e.g. strong curvature in the case of corners, or strong gradients in the case of contours), and thus are inherently *sparse*. This sparseness represents a constraint in the search space – a feature should match only with another feature – being an advantage in searching for correspondences. But this benefit comes with a cost: a sparse set of features also means a sparse reconstruction. There is a trade-off between the reduction in search space and the coverage of the scene. For instance, the correspondence of corner points is more constrained than the correspondence of contour points, but the results are also sparser.

Many types of features present low sensitivity to wide baseline settings. For in-

stance, contour-based features found in occlusion borders are prone to be consistent over images taken from significantly different viewing angles, where correlation based methods often fail due to the strong perspective effects and changes in the occluded or revealed background area.

Due to the added constraints and lower sensitivity to baseline, features can be used for tracking and camera calibration. For a more detailed overview of properties of feature-based methods the reader is referred to [Torr 00].

Another advantage of features (contours in particular) is that they are in many cases associated to discontinuities in the images. Typically these originate either from texture or depth discontinuities. Therefore, features themselves can be considered as depth cues, as an indication of the possibility of a depth discontinuity.

Feature-based methods have become popular especially with *Structure from Motion*, where feature tracking can be used both for camera calibration and structure recovery [McLa 00, Tell 00]. *Maximum Likelihood Estimation (MLE)* may be used with features for simultaneously computing camera calibration and points' positions from point correspondences [Fitz 98, Zucc 02, McLa 00]. Features are also particularly suited for wide-baseline matching [Prit 98, Tell 00]. [Lu 04] provides good coverage on feature-based techniques.

Reconstruction based on contours suffers some problems with *apparent contours*. Apparent contours are the projection of the boundaries of smooth surfaces whose view rays are tangent to the surface. A simple example can be seen in Figure 2.16. In general, an apparent contour is viewpoint-dependent – it changes with the views, even if the geometric surface that it belongs to is not occluded. When a contour being matched is apparent, two situations may occur: either no match is found in another image, as the contour only appears in the first image; or the contour is incorrectly matched with another contour, leading to an incorrect reconstruction.

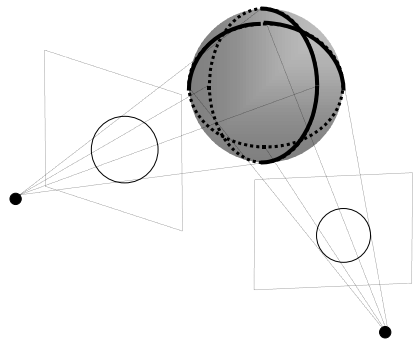


Figure 2.16: *Example of apparent contours*.
Contour points from one image are prone to be erroneously matched with points from the other image. Although the contours are similar, they correspond to different points in space

There has been research in this specific area of apparent contours (also referred to in the literature as *extremal boundaries*, *occluding contours*, *obscuring edges*) for

some time now. Vaillant and Faugeras [Vail 92] show that it is possible to reconstruct the *rims* – the actual 3D lines that correspond to apparent contours –, their normals and curvatures. Others researchers like Cipolla et al [Cipo 90, Cipo 92, Sato 99] and Szeliski and Weiss [Szel 98] explored this subject, and some current shape-from-contour techniques like [Mend 00] were partly inspired by these studies. However, to achieve this, small baseline settings are required, and the results obtained with real scenes are limited to scenes with single objects of simple geometry.

Three- and four-views methods

The use of the epipolar constraint for reducing the search space for correspondences of an entity to a line has been seen in Section 2.1.4. Suppose there are two possible candidates in the epipolar line. Adding a third view and the associated epipolar constraints, it is possible to disambiguate the candidate choice [Hart 00].

These constraints can be represented by a $3\times3\times3$ matrix, called the *trifocal tensor*. The trifocal tensor is the matrix form of the trifocal constraints, and it is equivalent to the fundamental matrix in the binocular case.

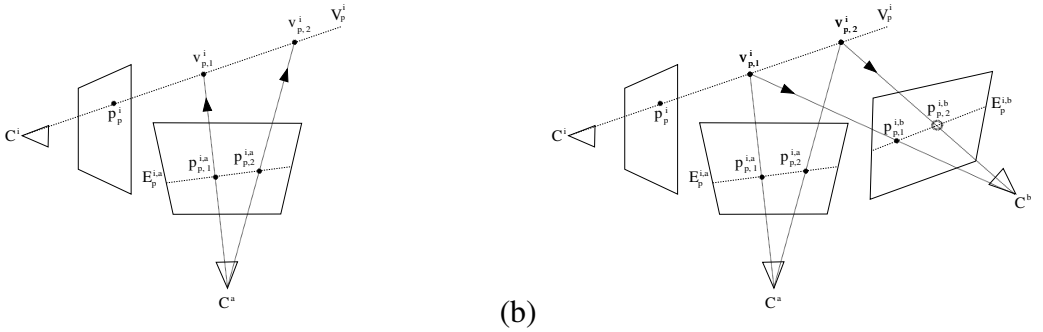


Figure 2.17: *Trifocal geometry*.

(a) Two possible candidates for $\mathbf{p}_p^i$ are found in $E_p^{i,a}$, and back-projected as world points in the view ray of $\mathbf{p}_p^i$

(b) These world points are projected into camera $\mathbf{C}^b$, which is used to disambiguate, leading to the rejection of $\mathbf{p}_{p,2}^{i,b}$.

The trifocal tensor provides constraints that can be used for camera calibration (to derive camera parameters from correspondences) or for disambiguating point or line correspondences [Shas 97, Ziss 96, Faug 95, Stei 00a, Fitz 98, Ziss 99, Mull 02, Mull 00]. Figure 2.17 shows an example of disambiguation: the two candidates chosen in (a) that seem equally good matches for the point being reconstructed are projected (b) on a third image. The incorrect candidate (in this case $\mathbf{p}_{p,2}^{i,b}$) has a very low probability of also being a good match in the third image.

Higher order tensors were derived, such as the *quadrifocal tensor* for the case of four views, but in terms of theoretical constraints do not add more information than the one provided by the trifocal tensor [Faug 95, Fors 03].

A more detailed overview of tensors is available in [Fors 03]. The uncalibrated case of the trifocal tensor is especially detailed in [Hart 00]. Given their specificity, they will not be explored further here.

2.2.3 Multi-view methods

Binocular techniques typically based in correlation require small baseline and are limited in the effective field of view [Prit 98] (three- and four-image techniques also, although to a lesser extent), in such a way that the resulting reconstructions typically resemble bas-relief representations of the scene. One can argue that two to four cameras can be distributed such that there is a significant spatial coverage (for instance, in three orthogonal faces of a cube, pointing to its center). However, reconstruction in such settings would be very difficult, if not impossible, due to occlusion and projection distortion. Furthermore, they are in general sensitive to issues such as repetitive patterns or texture-less regions.

Therefore, for reconstructions with a more complete spatial coverage, it is necessary to have more images taken from different viewpoints. Adding more images implicitly adds more stability and robustness. However, scaling binocular or three or four views methods to a larger number of images is not always trivial.

This leads to multi-view reconstruction methods. The following categories of such methods will be detailed here:

Binocular-based methods These techniques are inspired and based in the binocular and trinocular stereo techniques mentioned before, and perform reconstruction either by processing pairs or triplets of images individually, and merge the individual results incrementally [Poll 00, Corn 01, Poll 99, Fitz 98, Ziss 99], or they try to integrate simultaneously the information of multiple views [Tsai 83, Okut 93, Kana 97, Nara 98, Mell 96].

Volumetric methods These methods operate in *world space* (also referred to in some literature as *scene space*) by defining a bounded volume that will contain the reconstructed scene and discretizing it into 3D surfaces [Laur 94, Mart 83, Matu 00, Matu 02, Matu 01, Cheu 03, Mlay 03], *voxels* (**vo**lume **el**ements) [Seit 97, From 95, Kutu 00, Culb 00, Bhot 02, Stei 00b, Broa 00, Eise 99, Bone 99, Sait 99, Mlay 03], or 3D points [Coll 96, Kawa 01]. These entities are then projected into the images and a fitness or cost function is evaluated. If an entity is consistent with the images it is kept, if not it is either removed or adjusted.

Model-based methods Model-based techniques start from an initial model of the scene and iteratively adjust it to the objects in the scene by varying the model's parameters. Possible initial models are meshes [Fua 95a, Zhan 00], oriented particle systems [Fua 95b] or surfels. The initial model is then projected to the views of the scene and an error measure is computed. This error measure is then used to guide the readjustment of the model's parameters.

Spatiotemporal These methods are oriented to video sequences, where the baseline is very small and hence the differences between images are also very small. By stacking the images of the sequence in the time axis, a *spatiotemporal volume* is formed. Researchers try to extract scene information directly from this spatiotemporal volume [Adel 85, Peng 91, Capl 95, Duc 95, Otsu 97, Mosc 98, Rodr 02, Rodr 04c], by means of volume processing and analysis techniques.

There are other interesting approaches that fall outside the previous categories and will not not be explored here, as they are not sufficiently related to the work presented in this thesis. These include e.g. light fields [McMi 95], the use of genetic algorithms in [Kiri 98] or bipartite graphs [Chen 01].

Works that rely on manual assistance e.g. on the placement of primitives, or in the feeding of geometric constraints such as planarity or collinearity [Debe 96, Wilc 02, Gros 02] will also not be focused here.

Binocular-based methods

Structure from Motion

Structure from Motion (*SfM*) is a class of multi-view stereo techniques that was spawned from the video research community. SfM usually refers to "one moving camera"[7]. The use of the term "Structure from Motion" is not consensual. Some authors (e.g. [Torr 00]) distinguish between "Structure from Motion" – where camera motion is known and the scene structure is estimated from images – and "Structure and Motion" – where both scene structure and camera motion are estimated from the images. But in many cases the term "Structure from Motion" is used regardless of the camera parameters being known, estimated *a priori* or estimated simultaneously with the structure of the scene. Since camera calibration is not the focus of this work, the term "Structure from motion" will be applied generally here, regardless of the assumptions on known camera calibration. Structure from motion is associated to settings where there is a large number of images and a very small baseline, normally associated with a moving camera. Even though the large number of images would in theory allow an effective large or mixed baseline, in many cases the use of correlation-based methods leads to only an effective use of small-baseline pairs or triplets of images.

The video processing research community has been studying for years the field of *motion estimation* (also related to *optical flow* estimation) for video compression and analysis. One of the aims of motion estimation is to find correspondences between consecutive images (typically between blocks or segments) [Beau 95], which is also one of the key issues in depth and 3D reconstruction. Therefore, motion estimation and depth reconstruction have long been intertwined, and will likely continue to be, with the growing interest of providing 3D video services to users, and the need to generate 3D content from existing 2D video [Javi 02, Theo 04, Matu 04].

[7] As opposed to the "multiple static cameras" setting of other multi-baseline stereo referred later in this section.

The problem of camera calibration is usually not posed in the motion estimation community, as the focus is on getting the 2D motion of blocks in consecutive images, for which no 3D information is required. However, in the 3D reconstruction field, camera calibration represents a series of difficulties.

Camera calibration requires a set of reliable point correspondences to be established between images. Not all points are well suited for automatically finding a correspondence. Feature-matching methods are better suited for finding point correspondences for camera calibration, since they reduce the combinatorial complexity of the matching, and features are normally easy to differentiate and to match [Poll 00].

Consider the case of SfM techniques that assume no *a priori* camera calibration and that rely on feature-based methods for calibration.

These are divided in groups that use:

- feature-based methods for calibration and correlation-based methods for reconstruction
- feature-based methods for both calibration and reconstruction

While the first group provides a denser reconstruction, the second group integrates calibration and reconstruction in a single step, normally providing faster reconstruction at the cost of sparseness. The choice normally depends on the desired application and available resources. For instance, for object modeling a dense depth-map might be more appropriate, but for vehicle guidance a faster, feature-based sparse reconstruction could suffice [Smit 95].

The work by Pollefeys et al [Poll 00, Corn 01, Poll 99] is amongst the most established SfM methods up to date. It uses feature-matching for estimating epipolar geometry and a weak calibration between consecutive image pairs, i.e., a calibration where only the fundamental matrices that relate points from one image to the other are known, and not the actual intrinsic parameters of the camera (see Table 2.1 for types of calibration). The weak calibration allows the definition of a projective referential using the first pair of images. Each new camera is represented in that projective referential. This results in a projective reconstruction of features and cameras' positions (Figure 2.18(b)). Using a self-calibration technique [Poll 99] and making some reasonable assumptions on some intrinsic parameters (no skew, constant aspect ratio, principal point close to the image center), intrinsics and extrinsics are estimated, providing a strong calibration, and making it it possible to obtain a metric reconstruction.

Having a strong camera calibration, dense depth maps are obtained (Figure 2.18(c)) using a correlation-based stereo method applied to pairs of images. The images are rectified, such that epipolar lines correspond to image scan lines, thus simplifying the enforcement of the epipolar constraint. Other constraints that only apply to small-baseline settings are also used, such as the preservation of the ordering of neighboring pixels – if a point A is on the left of point B in image I^1, A cannot be on the right of B in image I^2. The independent dense depth maps computed between pairs of close-by cameras are *fused* together and undergo interpolation and regularization to obtain a single 3D model, based on the assumption of a connected surface with a parametric surface model.

Intrinsics	Extrinsics	Calibration	Reconstruction
Unknown	Unknown	Weak	Projective [a]
Known/Estimated	Estimated	Strong	Metric [b]
Known	Known	Full	Metric [c]

[a] Up to an unknown projective transformation.
[b] Up to an unknown scaling factor.
[c] Unambiguous.

Table 2.1: *Types of calibration and associated reconstructions*. The type of camera calibration is determined by the knowledge of the camera parameters. It also determines the type of reconstruction possible.

The results presented in their work show reconstructed surfaces that are triangulated from a single view, and the results of regularization are noticeable in the bumps of the the untextured surfaces. As the reconstruction itself is correlation-based, the method suffers from the common limitations of these methods, referred to in Section 2.2.2, such as limitation in baseline or problems in areas of low texture. Partial occlusions may also be an issue, since the small differences between images in the neighborhood of occlusion discontinuities cause problems to the correspondence. Furthermore, the assumption of a connected surface does not hold in several scenarios, such as those where self- or inter-occluding objects exist.

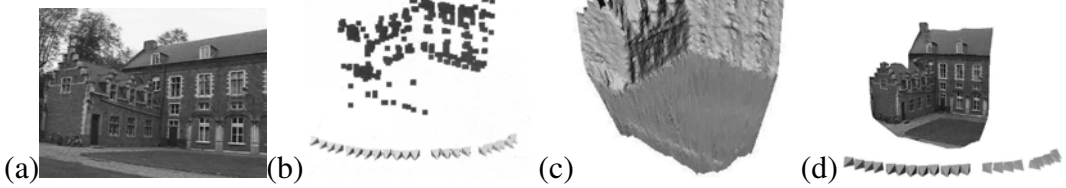


Figure 2.18: *Pollefeys' SfM technique [Poll 00].*
(a) One image of the scene
(b) Points and cameras reconstructed by point correspondence
(c) Dense model generated by correlation
(d) Textured model and recovered cameras

Although it is possible to consider wider baselines by using pairs of images that are separated in the sequence (the first and the last images, for example) this is not used in the process of reconstruction itself, only in the fusion process. This is due to the use of correlation-based reconstruction and the associated requirement of small baseline, and it means that the basis depth reconstruction does not really benefit from the wider baseline.

The applicability of this method to scenes where there are multiple objects occluding each other is not shown, and it is probably low due to the assumptions of parametric

connected surfaces used in the regularization.

Zisserman et al. [Fitz 98, Ziss 99] use a self-calibration procedure similar to Pollefeys, but correspond points and lines, and use image triplets and the trifocal tensor in addition to image pairs (Figure 2.19). This method is limited in terms of baseline, because although it establishes correspondences between corner points (and lines, as referred below), it does so using a correlation-based method applied to sets of pixels around the corner points (lines). This requires the camera motion between images to be relatively small, otherwise the perspective effects will be high and the correlation will be low (although a form of correction using projective warping is proposed for the case of lines and not-so-small baseline, in [Schm 97]).

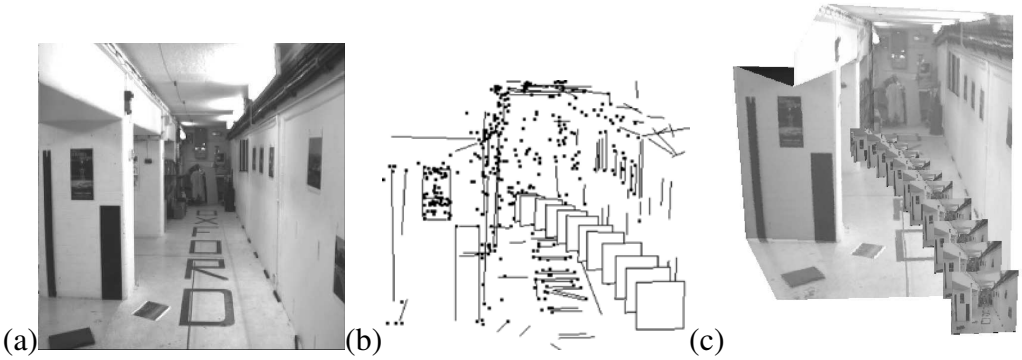


Figure 2.19: *Zisserman's SfM technique [Fitz 98].*

(a) One image of the scene

(b) Points and cameras reconstructed by point correspondence

(c) Recovered cameras and textured model after plane fitting

Correspondence is first determined between pairs of images, using the epipolar constraint, and later performed between triplets, using the trifocal tensor. These two steps provide point correspondences that are used for camera calibration and also provide reconstruction for those points. The use of the trifocal tensor over three images provides a geometric constraint for line correspondence, similar to the epipolar constraint over two images for point correspondence. This is used to test lines using the same type of correlation-based correspondence method, applied to points sampled from the line.

To provide a final 3D model, an approach oriented to polyhedral objects is proposed: to fit planes to the feature reconstruction and extract textures from the images for recovering a planar texture-mapped model. This option is not completely explored, and is proposed as a starting point for providing input to photogrammetry software requiring human intervention, such as [Debe 96].

Seitz and Dyer [Seit 95] proposed a feature (point and line) reconstruction algorithm that searches for correspondences between features in a reference image and features in other images using epipolar geometry. In the case of points, each feature p of the reference image is associated with a depth interval. By scanning the epipolar line of that feature on a second image, with a given tolerance, only the subsets of the initial

depth interval that are within the given tolerance of any features in the second image are kept. At each new image processed, only the set of intervals that contain features in *all* processed images are kept. After processing all images, the reconstructed points are obtained by minimizing an error function over the features associated to the final depth intervals. In the case of lines, a point p_l of each line l in the reference image is selected to represent the line. The epipolar scanning method is similar. The final depth interval for this point will correspond to the intersection of the projections of all the lines from the other images, thus establishing correspondences between p_l and line segments in all the images. The error function to minimize is adjusted to lines. The algorithm relies on rectification for the scanning of epipolar lines, which limits the freedom in camera placement. Furthermore, by keeping only the depth intervals for which features are observed in *all* images, occlusion or errors in feature detection are not supported. In the case of lines, if the representative point selected is missed in one of the other images due to noise or problems in edge detection, it is most likely that the corresponding depth interval does not survive to the minimization stage.

Rother and Carlson [Roth 02] proposed a linear algorithm for computing the position of 3D feature points and camera positions, assuming that a reference plane is visible in all views. The focus is set on the proposed linear system relating correspondences between points and the known plane, by using *singular value decomposition* (SVD). Later Rother [Roth 03] extends the linear constraints to lines and planes. The issue of correspondence is not directly addressed.

Multi-baseline methods

These techniques are distinguished from the previous ones in the sense that they are oriented to settings of multiple static cameras, for which there is no particular order, and use larger baselines, when compared with SfM methods.

Initial work on correlation-based multi-view reconstruction was reported by Tsai in [Tsai 83]. He placed eight cameras in a coplanar grid, with parallel optical axes (Figure 2.20), to simplify the process of deriving epipolar lines between cameras. Correct matches were determined by computing a window-based measure, for a given point over the epipolar lines. In the ideal case, a maximum (or a minimum, depending on the type of measure used) would correspond to the correct match. In the simple two-image case, this assumption would often fail, due to ambiguity and a number of other factors. However, if more images are added and the measurements accumulated, the peaks at the correct match should get sharper. The method was only tested with very simple synthetic scenes (as the one shown in Figure 2.20), and it was highly dependent on the fixed eight camera arrangement.

Okutomi and Kanade [Okut 93] extended this work by using the sum-of-squared-differences (SSD) as an error measure, and accumulating it over multiple image pairs with respect to inverse depth , defining the sum of the SSD (*SSSD-in-inverse-depth*). This measure allowed them to reduce the matching over multiple images to a one-dimensional search in inverse-depth space, instead of multiple searches between pairs of images in disparity space.

They identified a set of pathological cases and showed how their multi-baseline

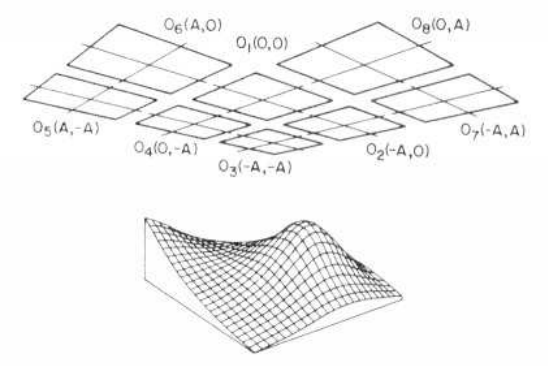


Figure 2.20: *Tsai's multi-baseline stereo [Tsai 83].*
Projection planes of the fixed 8-camera setup and synthetic model

system would help reducing their influence, such as the existence of repetitive patterns in the images.

This method was however quite limited in terms of requirements: cameras had to be aligned along a straight line, with their optical axes perpendicular to that line and parallel to each other, and with a small baseline between consecutive cameras.

Later, Kanade et al. [Kana 97, Nara 98] adapted this method to slightly less restricted camera configuration – non-parallel cameras – for using in their Virtualized Reality system. Their set-up consisted of a 5-meter diameter geodesic dome with 51 cameras evenly distributed around it (Figure 2.21).

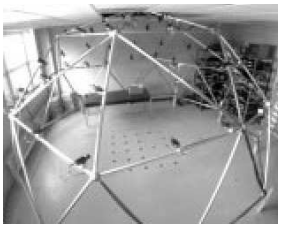
(a)  (b) 

Figure 2.21: *Kanade et al's virtualized reality [Kana 97].*
(a) The 3D Dome, with 51 cameras and calibration pattern
(b) Some images from an example scene

They use the modified multi-baseline stereo algorithm with sets of three to six cameras simultaneously, and state that adding more arbitrary cameras may not improve reconstruction quality due to the increased difficulty in matching.

The method suffers from problems in areas of low textures and along depth discontinuities, which made them incorporate an interactive depth map editor, to manually edit the obtained depth maps.

Mellor et al [Mell 96] generate dense depth maps from large sets of images (thousands) taken from calibrated cameras in arbitrary positions. Their method is also partly

inspired by [Okut 93].

The estimation of depth for an image's pixel p is performed by extracting the epipolar lines of p from the other images. These epipolar segments are stacked to form an *epipolar image*[8], such that each row is an epipolar segment taken from an image, and each column corresponds to a candidate depth for p (Figure 2.22). This image is then analyzed per column (candidate depth).

The match quality associated to a candidate depth d is the accumulation of an individual match quality between the color value of p and the values in the column corresponding to d[9]. The result is a function of depth candidates vs match.

Ideally, this function will have a sharp, distinct peak at the correct depth. Detecting a global maximum should provide the correct depth, but occlusions and repetitive patterns can cause false negatives or the absence of a peak. For this reason, the authors propose to perform an initial reconstruction, and compute a rough estimate of the surface normal of points that are considered reliable. This information is then used to disambiguate other points, by reasoning about occlusion.

The results presented in their work show only the reconstruction of a small area of a synthetic scene, and present a significant number of outliers, using 100 images. The cameras are placed apparently at arbitrary positions, but there are no results with real scenes, neither any considerations regarding the influence of camera calibration errors in the reconstruction.

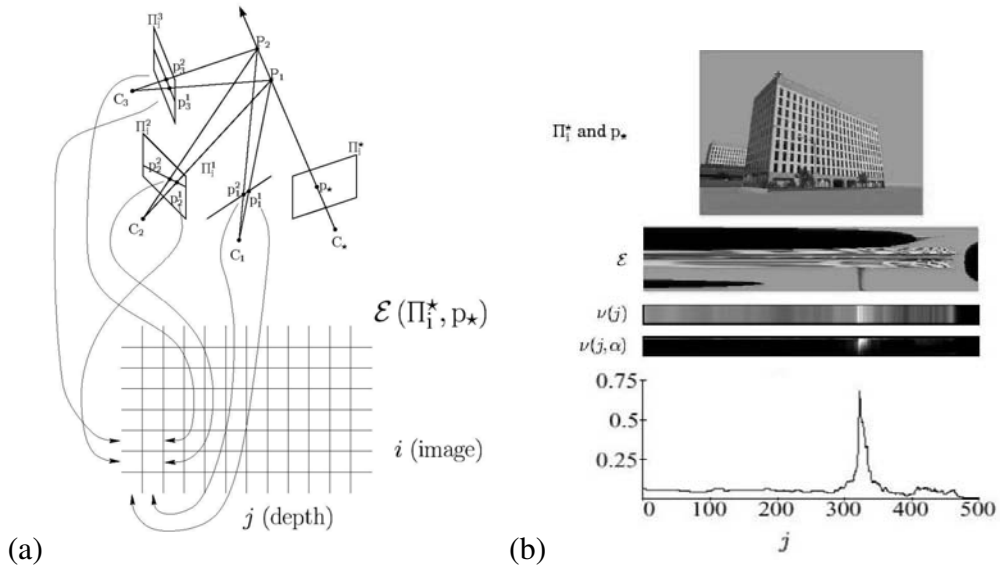
A dynamic programming approach was used by Cox [Cox 94]. The algorithm detects a *best set* of pixel correspondences between a pair of calibrated cameras – the *principal pair* – by performing minimization of a cost function that applies to pairs of points (one in each principal camera) and takes occlusion into account. This first minimization provides a cost for each pair of points. In the second minimization step, a second cost function is applied to the pairs that resulted from the first minimization, this time using also the information from the other $N-2$ images. This cost function is not minimized for each pair, but over all the feasible combinations of the initial pairs. In this second optimization, the number of depth discontinuities (either horizontal – along each scanline – or vertical – between scanlines) are also minimized, as an attempt to remove outliers without a regularization procedure that would smooth depth discontinuities. Still, this second optimization is computed as an approximation that takes into consideration horizontal and vertical discontinuities separately, resulting in overall problems in discontinuities.

The use of a single principal pair also limits the method, as the choice of the pair may influence the stability of correspondence (e.g. by selecting two cameras too close to each other). Typically the cameras which are further apart would be used.

Tests only show examples of simple camera motions with small baseline, namely horizontal and vertical translations, with 7 to 13 cameras. Camera calibration is assumed to be exact, and the problem of sensitivity to calibration errors is acknowledged,

[8]The epipolar image concept is an extension of Bolles et al's epipolar-plane images [Boll 87] to arbitrary camera positions.

[9]More specifically, the negative of the absolute difference of colors in the hue, saturation, value (HSV) color model.



(a) (b)

Figure 2.22: *Mellor's epipolar image approach [Mell 96].*

(a) Creation of an epipolar image

(b) (top to bottom) One Image of the scene, the epipolar image ε for a selected point, the normalized match quality $\upsilon(j)$ (coded in greyscale) and the updated match quality $\upsilon(j,\alpha)$ including an estimate α of the surface normal (greyscale and graph)

but not addressed in any way. This technique has the high computational complexity inherent to global optimization techniques.

Szeliski and Weiss [Szel 98] try to reconstruct the apparent contours of an object using a sequence of images captured with calibrated cameras (Figure 2.23) placed around the object. They detect and link edges in the images, and track (correspond) edge elements (*edgels*) across consecutive images. Given three or more edgels tracked, they try to compute the location and curvature of the surface by fitting a circular arc tangent to the lines defined by the view directions at those edgels.

They are able to reconstruct a network of linked surface points, that include not only points from the apparent contours but also other points on surface marking or creases (points that are detected by the edge detection algorithm used). The method requires small baseline so that the contours are close to each other in consecutive views.

Each test presented consisted of an object captured with 72 cameras around it, with a difference in orientation of 5 degrees. The results show apparent contours nicely reconstructed in scenes with sparse contours, such as the mug in Figure 2.23, but also a number of outliers, especially in scenes with higher contour density, where there seems to be some cluttering in reconstructed points.

Jung et al [Jung 02] extract 3D edgels from aerial images to characterize and reconstruct buildings and other man made structures. Contour points with an orientation – *edgels* – are extracted from an undistorted image and chained (linked). Each point is then reconstructed by analyzing its epipolar lines in other images, and the edgels that intersect those lines. The view rays of intersecting edgels taken from different

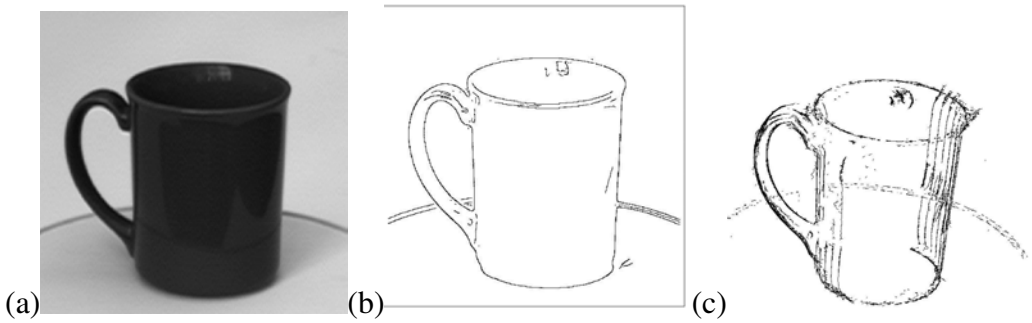


Figure 2.23: *Szeliski and Weiss' apparent contour reconstruction [Szel 98].*
(a) One image of a scene
(b) The extracted contours
(c) One view of the resulting reconstruction

images form a bundle, for which some rays are expected to intersect close to the original point's view ray. The best bundle of rays that intersect close to the original ray is selected using a Least Median of Squares technique. After 3D location has been estimated, orientation in 3D for edgels is also estimated.

The problem is highly constrained by the use of aerial images. This allows them to define a tightly bounded section of the epipolar lines to be analyzed, according to a predefined maximum and minimum height. Tests were performed with small sets (around 9) of high (but unspecified) resolution images, with an overlap of 60% between adjacent images, and very similar viewing angles. Such constraints, however, limit the method's applicability to any other type of scenes, making it too specific.

The work by Bauer et al [Baue 04, Klau 03] consists of a plane-sweeping method based on Jung's work. For a chosen reference camera, a plane is "swept" across a predefined region of space. In each iteration, each edgel extracted from the reference camera is projected in the sweeping plane and reprojected to the other images. These projections are evaluated according to their distance to other edgels, and if they fall outside a threshold in a significant percentage of images, they are rejected. The result is a set of hypothesis that are then refined according to the gradient direction of the edgels.

Hypothesis that have a gradient nearly orthogonal to the epipolar line of the original edgel are rejected, as they provide ambiguous hypothesis. Another criteria is that the angle of the gradient in the reference image and the angle in another image have to be close, up to a certain threshold, for the hypothesis to be accepted. With this criteria, it is assumed that there is no severe rotation between images, thus imposing a limitation in the camera placement.

The surviving hypothesis' locations are refined on a neighborhood of the sweeping plane. Still, many outliers subsist. To remove them, an image-based similarity measure is used. The image-based descriptor used is not rotation invariant, thus requiring rectification of the area surrounding the edgels to characterize. Results are presented for a synthetic and two real scenes. Camera calibration is performed in what seems to

be a semi-automatic method, but no considerations are made on errors on the camera calibration. The images used for the real scenes use resolutions over 2.5 mega pixels. It is mentioned that only a small number of false positives is present in the results. However, these are not quantified.

Many feature-based methods are oriented to the structure and motion problem.

Tomasi and Kanade's factorization method [Toma 92] is an important work in this area. It used a series of points tracked over a sequence of images, and represented the points in a measurement matrix, that can be factored into two matrices, one representing cameras' rotations and the other representing the points in a common 3D referential. Cameras' translations can be obtained from the original matrix. Albeit seminal, their work was oriented to the case of orthographic projection, and the proposed method for tracking features was very simple, limiting its applicability to more complex camera motions and the type of scenes. This work inspired extensions to other types of projections – e.g. paraperspective [Poel 94], perspective [Stur 96] – and to other variants in terms of the choice of entities (e.g. lines) and their tracking methods, such as [Mart 03, Agui 01].

The use of lines for solving structure and motion has also approached as a minimization problem e.g. by [Tayl 95].

The main drawbacks of these techniques is that although the aim is to solve camera calibration and 3D reconstruction simultaneously, the correspondence estimation in which they rely is often neglected. Correspondences are typically either assumed given and mostly correct, or the methods employed to estimate them are constrained to specific scenarios (e.g. small baseline), or are applicable only to a relatively low number of features (those that are trackable).

Volumetric methods

Volumetric reconstruction methods operate in world space (or scene space). The *visual hull* and *voxel coloring* volumetric approaches and some variants will be described in more detail. Other relevant techniques, such as the *space sweep*, will also be referred. A more detailed survey of these techniques can be found in [Slab 01].

Visual Hulls

The visual hull of an object is defined as the maximal shape that gives the same silhouette as the actual object, for all views outside the convex hull of the object [Laur 94]. Visual hull techniques are also known as *volumetric intersection* or *shape from silhouette*.

Visual hulls are typically constructed from a set of images with known camera positions. Each image is segmented into a foreground object and the background. The foreground regions (the *silhouette*) are back-projected into 3D space, forming a set of volumes (or "cones"). The intersection of these volumes is the *inferred visual hull* (Figure 2.24).

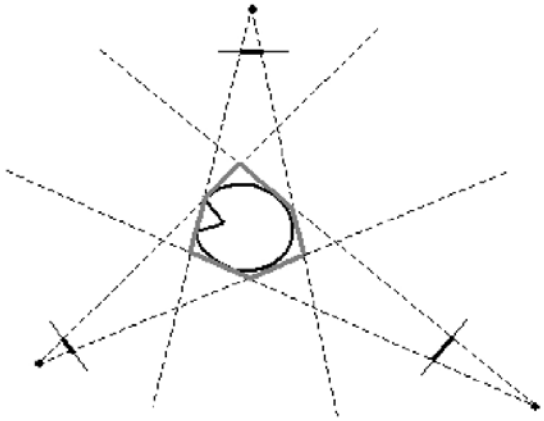


Figure 2.24: *Example of a visual hull.*
Silhouettes are back-projected from the corresponding cameras as cones; the intersection of the cones is the inferred visual hull (gray polygon)

Visual hulls have some important properties, namely:

· It is guaranteed to enclose the object;

· Its size decreases monotonically with the number of images used.

One of the issues in visual hulls is their representation. Some early works on visual hulls used orthographic projections and parallelogram structures to represent visual hulls [Mart 83]. Later, other authors [Chie 86, Veen 86, Sriv 90, Potm 87] used octrees to store the hulls generated from each image, by building an octree from each image and then combining the various octrees to obtain the visual hull. In this sense, Szeliski's work [Szel 99] was innovative, as he used a single octree that was iteratively refined, with each new view. This implied a significant increase in processing speed over other octree-based techniques,

A series of works in the field of image-based visual hulls emerged later, namely Matusik et al's work [Matu 00, Matu 02, Matu 01, Matu 04]. They construct polyhedral visual hulls using *constructive solid geometry (CSG)* to intersect silhouette "cones" and provide a textured polyhedral model (Figure 2.25). It already performed at 15 frames per second, but using only four cameras at low resolution (320x240), each connected to a computer that captured and pre-processed images, and a separate server that would receive the processed images and performed visual hull computation and rendering. The quality of the results without texturing was relatively low, in particular due to the small number of views available.

Other proposals include combined solutions for reconstruction refinement using visual hulls and stereo [Cheu 03], or visual hulls and color consistency [Mlay 03], among others.

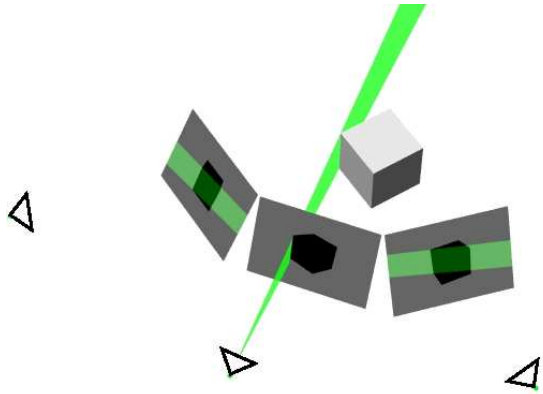


Figure 2.25: *Silhouette cone intersection*.

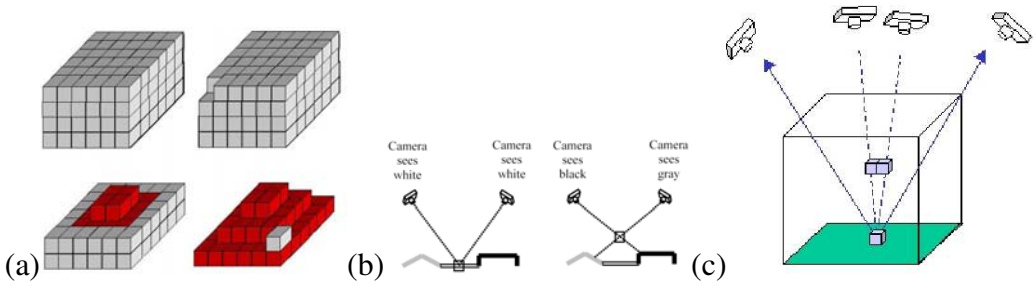


Figure 2.26: *Voxel coloring*.

(a) Volume in several stages of coloring. Voxels are successively removed

(b) Voxel removal is decided based on color consistency between cameras

(c) Visibility has to be taken into account when determining color consistency

Voxel Coloring

In voxel coloring algorithms, an initial volume in world space is defined and subdivided in smaller volume elements – *voxels* (Figure 2.26(a)). The goal is then to identify *empty space voxels* and remove them from the voxel space. To this end, it is assumed that the scene is Lambertian (diffuse or non-specular) and that points on surfaces are *color-consistent* (or *photo-consistent*): they project onto similar colors in all the images they are visible. Therefore, if a voxel projects to consistent colors in the images where it is visible, it is assumed to be a *surface voxel* and assigned a color that is also consistent with its projections, otherwise it is removed (Figure 2.26(b)). The algorithm finishes when all visible voxels are color consistent.

Visibility is a key issue in these algorithms, as depicted in Figure 2.26(c). Some voxels are not visible in certain cameras due to occlusion by other voxels. Those cameras have to be ignored when evaluating color consistency for the occluded voxel, otherwise they will (in the general case) erroneously cause its removal. For this reason, the order in which voxels are visited is crucial, and it is one of the main distinguishing factors between variants of voxel coloring algorithms.

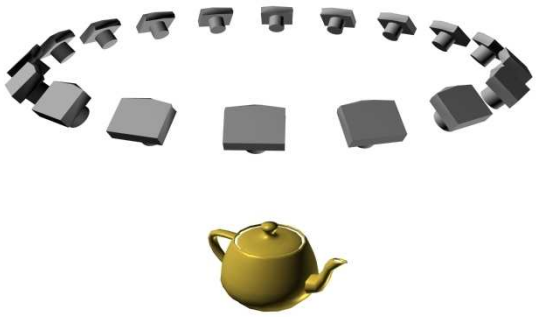


Figure 2.27: *Seitz and Dyer's voxel coloring [Seit 97].*
The algorithm requires that the reconstruction volume is outside the convex hull of the cameras, as in this example

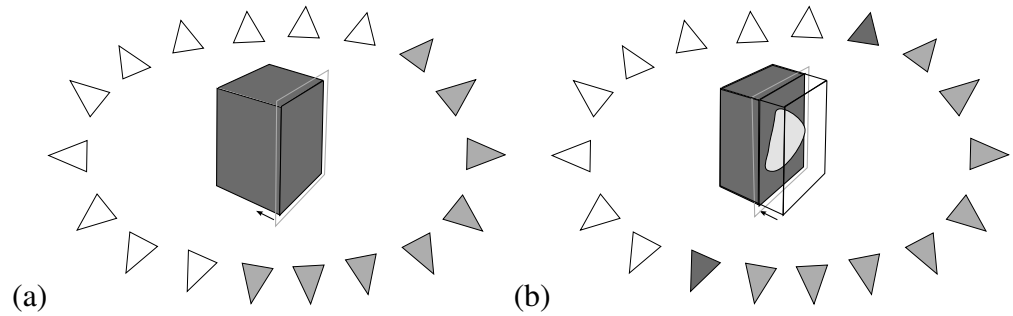


Figure 2.28: *Kutulakos and Seitz's space carving [Kutu 00].*
(a) Initial step: only cameras behind the sweeping plane (marked in gray) are used
(b) After some iterations: the sweeping plane moved, allowing for more cameras (darker gray) to be used

The basic *voxel coloring* technique was introduced by Seitz and Dyer [Seit 97]. They show that if the reconstruction volume is totally outside the convex hull of the cameras (a common example of such a setting can be seen in Figure 2.27), it can be swept by a plane in front-to-back order, away from the cameras, and this guarantees that occluding voxels are visited before occluded voxels. Each time a color consistent voxel is identified, the corresponding image pixels where it is projected are flagged, and will not be used in the consistency check of any other voxel.

Seitz and Dyer's technique was extended in several ways. Kutulakos and Seitz proposed the *Space Carving* technique [Kutu 00], that removes the camera placement restrictions through a multi-sweep approach, using axis-aligned planes and using only the subset of the cameras that are behind the sweeping plane (Figure 2.28). This solution is computationally more expensive, and does not use all the available visibility information. There may exist cameras that are not behind the sweeping plane that "see" a voxel in the sweeping plane, yet those cameras are not used in the consistency check. This potentially results in uncarved voxels, when the ignored cameras are the only ones that have the evidence for carving a given voxel.

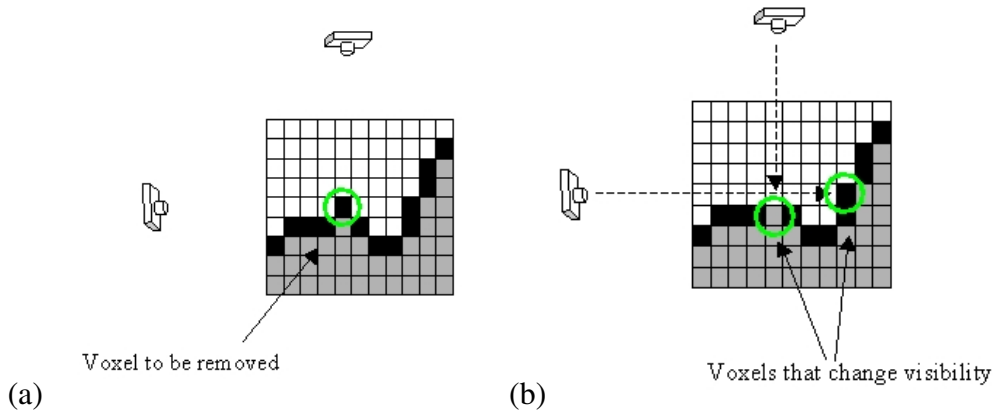


Figure 2.29: *Generalized voxel coloring [Culb 00].*
Visibility is recomputed every time a voxel is removed.
(a) A voxel is selected for removal
(b) Two voxels are exposed

A *Generalized Voxel Coloring (GVC)* algorithm was then proposed in [Culb 00]. This algorithm deals with visibility in a more sophisticated way, ensuring its proper handling. As the authors note, the carving of one voxel potentially changes the visibility of other voxels (Figure 2.29). When the visibility of a voxel changes, its color consistency should be reevaluated using all the cameras in which the voxel becomes visible, and only then should the voxel be carved if it is found to be inconsistent. In the basic GVC algorithm, when a voxel is carved, all the other voxels' color consistencies should be re-evaluated. Since this is a time-consuming operation, the authors propose a variant of the basic GVC algorithm using *Layered Depth Images* (*GVC-LDI*). A LDI stores for each pixel a list of all the surface voxels that project to that pixel. When a surface voxel is carved, visibility can change for two types of voxels: (1) interior voxels adjacent to the carved voxel or (2) surface voxels that are on the lists of the LDI's corresponding to the carved voxel (in fact, in the second position of those lists). This provides an effective way for dealing with complete visibility information for consistency checks.

Further extensions have been developed. Multi-hypothesis approaches were proposed in [Eise 99, Bhot 02], that assign to each voxel a probability of being occupied, based on visibility information. The idea is to keep multiple hypothesis of the visibility of each voxel, and later refine or remove those hypothesis. The work in [Bone 99] takes this occupancy probability concept further, assuming voxels are semi-transparent, and mapping the occupancy probability to a transparency value. In [Sait 99], a projective grid space for the voxels is proposed, based on two basis views. It has the distinctive flavor of being based on projective geometry, requiring only weak calibration. Another extension to the voxel space is presented in [Slab 00], to allow voxel coloring without being limited to a bounded volume in space, but rather to use an infinite or semi-infinite volume. Some extensions to the method used to check color consistency between voxels and images have been proposed in [Broa 00, Stei 00b], namely by directly using an extended neighborhood of the projections of a voxel's centroid in each image, or

by projecting the voxel instead of its centroid. [Mlay 03] presents an hybrid approach using visual hulls and color consistency measures to join the lighter requirements of the former with the improved quality of the latter.

Other volumetric methods

Collins' space sweep approach [Coll 96] is a feature-based method that operates in world space. In practice, the world space is divided into a set of bounded planes parallel to each other (Figure 2.30), each plane divided into cells. Features (edges) are extracted from each image, and the view ray corresponding to each feature is projected as a pixel-shaped cone, contributing with votes to each cell that is contained in that cone. The technique is associated to the name of "space-sweep" because the voting procedure takes place for one plane at a time, and each cone is processed for each plane. After the features of all images are projected on the cells of one plane, the cells that have a number of votes above a given threshold represent the reconstructed features. The relevant cells are stored, and the next plane is processed. Baseline issues are not referred, but given the use of a single plane direction and the test presented with aerial images with small baseline, it is expected that wide baseline may pose problems e.g. with cameras with orientation close to parallel to the sweeping plane.

A similar method is used by Taillandier and Deriche [Tail 02], contemplating lines. Using a plane-sweeping technique, sets of possible correspondences are collected if two or more labeled line segments from different images intersect in at least one voxel. They determine 3D infinite lines by intersecting the planes defined by the labeled segments and the corresponding cameras, using an iterative method. The final line set is pruned to correspondences matching segments from three or more images. A unicity constraint is imposed iteratively, such that a 2D segment can only be associated with one 3D line, starting from the larger segments. Each actual 3D line segment then results from union of the projections of the 2D lines over the corresponding 3D infinite line. This technique was presented with a focus on aerial and small baseline settings, and it is not clear how good it would perform in wider baseline settings. It also does not consider errors in camera calibration, which may cause problems e.g. when the the planes corresponding to two corresponding 2D line segments do not intersect within the swept volume, leading to missed correspondences. The use of the distance-weighted metric proposed in this thesis would help in such cases.

Kawamoto and Imiya [Kawa 01] propose a reconstruction method based on random sampling of feature points in images and a voting procedure in world space. If a pair of points selected randomly from two different images obey the epipolar constraint (within some noise tolerance), a vote is casted for the corresponding 3D point in an accumulator space (similar to an Hough transform [Illi 88]). After a predefined number of iterations, peaks are detected in the accumulator space. These peaks correspond to the reconstructed points, and the accumulator space is dynamic (only bins with at least one vote are actually created). The authors propose a similar scheme for 3D line reconstruction, using two pairs of points and using a six-dimensional positive-unit semi-sphere as the accumulator space. This method presents some problems. On one hand, the choice of pairs is performed randomly, without considering epipolar

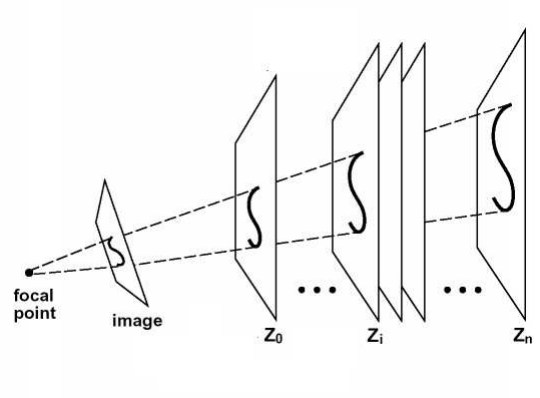


Figure 2.30: *Collins' space sweep approach [Coll 96].*
An axis-aligned direction is chosen and a plane perpendicular to it is "swept" over space for gathering votes

constraints. If after selecting the pair, the epipolar constraint is not observed (within a tolerance) the pair is rejected. Given the random choice, this process will be unnecessarily repeated for many pairs that could be ignored from the start. Furthermore, 3D points that may have projections in multiple images may not benefit from all the votes of those images, if the random choice does not contemplate the corresponding pairs of projections.

Model-based methods

Model-based reconstruction techniques take a predefined model of the scene (or scene objects) and iteratively adjust it using information from the images.

Fua presented model-based techniques using particle systems [Fua 95b] and meshes [Fua 95a]. The particle-based approach [Fua 95b] consists of an initialization step based on correlation stereo or other source of 3D data of the scene (Figure 2.31(a)). Disc-shaped particles are fitted to the 3D points (Figure 2.31(b)). The position and orientation of the particles are refined using an image-based objective function. In a final stage, spurious particles are eliminated and others are clustered if they appear to belong to the same surface (Figure 2.31(c)). The particles can then be shaded for visualization (Figure 2.31(d). The mesh-based approach [Fua 95a] is initialized with an initial estimate surface shape, which could be provided by a triangulation of results from a conventional stereo algorithm. An iterative adjustment of the surface shape and reflectance properties takes place, in order to minimize an objective function comprising information from multiple images. This function is a weighted sum of stereo, shading and smoothness components. The goal is to give more importance to one of the components in settings where it is more reliable (e.g. stereo on highly textured areas)

Another mesh-based approach is the one presented by Zhang and Seitz [Zhan 00]. An initial triangle mesh (such as a sphere or a cube) is iteratively deformed (Figure 2.32) until it minimizes an image consistency metric. On each iteration, all images are projected on the mesh, and each triangle is sampled into a set of points. A displacement

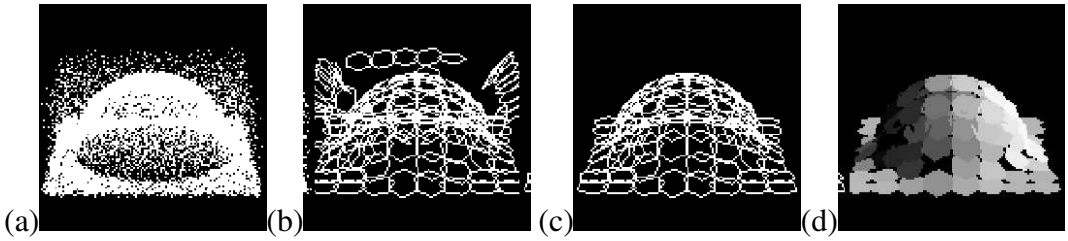


Figure 2.31: *Fua's particle-based reconstruction [Fua 95b].*
(a) Noisy point cloud resulting of initialization
(b) Particles instantiated by fitting procedure
(c) Particles remaining after clustering
(d) Particles after shading

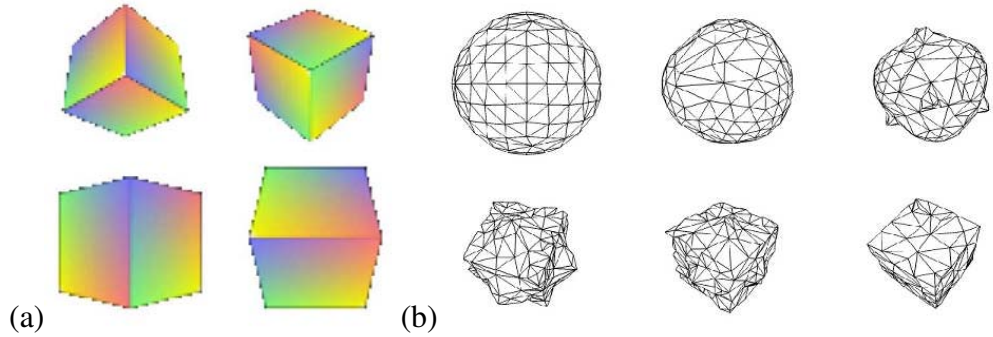


Figure 2.32: *Zhang and Seitz's mesh refinement approach [Zhan 00].*
(a) Four views of the scene
(b) Six stages of mesh deformation

is computed for each of these points according to its color variance in all the images and its normal (if the variance is large) or the variance gradient (if the variance is small). If the resulting set of displaced points cannot be approximated by a single triangle, triangle subdivision takes place, followed by mesh simplification. These steps are iteratively repeated until the consistency metric is optimized.

Kahl and August [Kahl 03] propose an energy minimization technique over parameterized curves. An initial set of curves is generated based on edges of a reference image. Each of these curves has a potential computed as a cost between its projection in the images and the gradients, and the corresponding energy is minimized over the parameterization of the curve. The method has high computational complexity, e.g. in the selection of the initial line set, and there are problems dealing with occlusions.

Faugeras and Keriven [Faug 98] use a surface as a model, that is implicitly defined as the zero set of a function over $\mathbb{R}^3$. A functional is then maximized based on pairwise correlation using variational methods. Camera calibration is assumed to be known. The method is sensitive to lighting variations and specularities. Only two textured reconstructions are provided, and no performance or other quantitative measures are

provided to allow comparison to other methods.

Spatiotemporal methods

Video sequences can be represented as a three-dimensional volume (2D plus time). The 3D volume consists of the set of consecutive frames stacked in the time axis direction. The goal is to try to apply volume analysis and processing techniques to infer motion information from this data. The idea is that motion patterns in the sequence are represented by geometric patterns in the spatiotemporal volume. Although motion estimation is the main output of these techniques, it is possible to derive spatial 3D information, if camera parameters are known.

These *spatiotemporal* techniques have been mostly applied to very limited motion models (mainly translational) as referred below, and many of them use techniques that are computationally quite expensive, such as Markovian models and 3D Hough transforms.

The seminal work by Adelson and Bergen [Adel 85] is oriented to visual perception rather than image processing. It was one of the first to suggest the spatiotemporal volume and the evidence of the paths formed in the time direction (referred as the "motion as orientation" effect). It also mentions the step effect arising from time under-sampling (Figure 2.33). It proposes the detection of motion models based on the application of energy and impulse response filters to the spatiotemporal volume.

Peng [Peng 91] slices the spatiotemporal volume in predefined orientations (horizontal, vertical and diagonals) in order to get a dense estimate of the velocity field. Each slice is segmented into strips (regions of similar intensity) (Figure 2.34), representing areas of similar motions. An adaptive smoothing algorithm is used to segment slices into strips, and adds significant computational load to the overall process.

Caplier and Luthon [Capl 95] propose a Markov Random Fields extension called Markov Random Volume (MRV), which is an extension of concepts like Markov chains and MRF to the 3D case, and apply it to a spatiotemporal volume.

Duc, Schroeter and Bigun [Duc 95] present a spatiotemporal framework that allows estimating a fully parametric motion model over an image sequence. An image sequence showing one motion is considered as the transformation throughout time of a 2D pattern. By defining a group of possible transformations and expressing it as a parameterized equation based on Lie's theory [Blum 89], motion estimation consists in finding a set of parameters that solve that equation. They use a robust estimator – the minimum volume ellipsoid (MVE) - and a luminance-based static segmentation to cope with multiple motions.

The technique of Otsuka, Horikoshi and Suzuki [Otsu 97] focuses on the surfaces formed by edges and contours of images in spatiotemporal space, called the *trajectory surface*. The trajectory surfaces are built by pixel-based frame differences followed by thresholding and selecting. The planes that are tangent to the trajectory surface are extracted using a 3D Hough transform. The orientation of the tangent planes provide the motion information required.

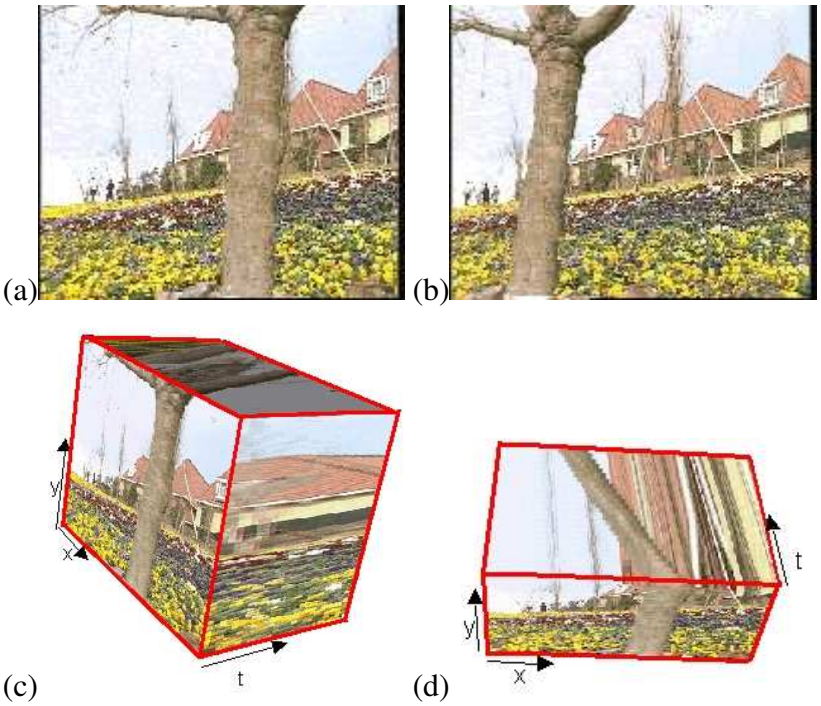


Figure 2.33: *Example of a spatiotemporal volume and the effect of (apparent) motion.*
(a) First frame of a total of 25 in a video sequence (the camera is translating from left to right)
(b) Last frame of the same sequence
(c) Spatiotemporal volume generated by "stacking" the frames along the time axis
(d) The same volume, sliced horizontally. The paths formed by the apparent motion are especially noticeable by the difference in orientation between the area corresponding to the tree and the other areas

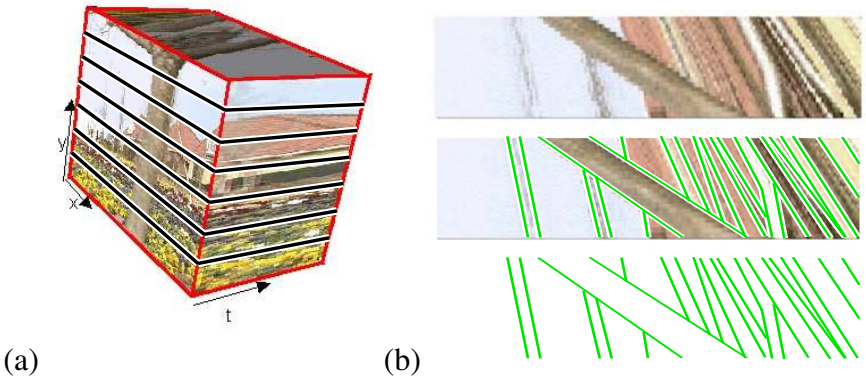


Figure 2.34: *Peng's spatiotemporal slices and strips [Peng 91].*
(a) Multiple slices in a spatiotemporal volume
(b) Extraction of strips from a slice: the slice (top), the detected contours superimposed on the slice (middle) and the resulting strips (bottom)

Moscheni, Sushil and Kunt [Mosc 98] use a region-merging algorithm, based on a spatiotemporal similarity measure, and a graph-based clustering technique to perform spatiotemporal segmentation on pairs of consecutive frames.

As a preliminary work for this thesis, Rodrigues et al [Rodr 02, Rodr 04c] published a method that created a spatiotemporal volume – the *videocube* – using image contours, and used a distance-based metric to evaluate the fitness of candidate spatiotemporal curves to the volume. By applying distance transforms [Borg 84] to the videocube formed by the contours of the images, a distance-transformed videocube was obtained, where each point of the volume contained the closest distance to a contour. Assuming that camera calibration was available, a set of spatiotemporal curves were generated for each contour point in a reference image, based on a set of candidate depths. The error of each of the curves was then evaluated in the distance-transformed video cube, based on the distance values associated to the points traversed by the curve. The choice of a curve with low error determined a depth and hence a reconstruction for the point. Issues such as occlusion and the existence of multiple minima were acknowledged and dealt to some extent. The method was at the time only applied to small-baseline sequences.

2.3 Final remarks

Multiple methods for 3D reconstruction based on images were reviewed in this chapter. Considering what was presented throughout the chapter, several conclusions can be drawn.

Reconstruction methods based on few images are in general sensitive to all types of noise and artifacts (camera calibration, image noise, repetitive patterns, textureless regions). Adding more images implicitly adds more stability and robustness to such methods. However, scaling such methods to multiple images is not always trivial.

Various compromises can be identified in the various classes of multi-view methods presented in this chapter:

Correlation-based methods Most correlation-based methods are restricted in camera freedom, either by limiting (i.e. pre-defining) the position and orientation of the cameras, by restricting them to a given orientation, or by requiring small baseline. Correspondence is facilitated in these settings, and it is possible to obtain dense depth maps. But the resulting reconstructions are either limited in the volume of the scene that it is covered (in the case of a small number of cameras) or require merging procedures to join results from individual pairs or triplets of images. Furthermore, dealing with partial occlusion is also a common problem, since the occlusion or revealing of some parts of an image at occlusion borders may lead to problems in the correspondences such as incorrect matches, or blocking artifacts.

Feature-based methods Feature-based methods produce sparse outputs, but are less sensitive to varying camera setups with wide baselines, and are less affected by varying lighting and highlights than correlation-based methods. They also

provide more 3D information at objects borders, which are troublesome in many correlation-based methods.

Volumetric methods Volumetric approaches typically produce better visual results, but also have specific requirements and problems.

- Visual hull techniques are simpler to implement than voxel-based techniques, but they require background subtraction and their results are a conservative approximation of the actual shape of the object, which can be very coarse when there are only a few cameras. Moreover, the estimated shape is sensitive to noise in the silhouettes and camera calibration errors. Although they are less sensitive to camera calibration errors than e.g. correlation-based methods, they are still subject to some artifacts caused by such errors.
- Voxel-based methods require color consistency, which simplifies the task of coloring voxels and is responsible for the visual quality of the reconstructions. However, they can be quite sensitive to camera calibration errors and varying highlights that alter the required color consistency. These methods suffer from visibility issues that require complex algorithms to solve them.

For a practical and generic application, one will typically need to combine different reconstruction techniques. In practice there is no single method that is applicable or adequate to all types of scenes, input conditions and output requirements. Each method has problems in dealing with some of the following issues (represented in Table 2.2):

- Freedom in camera placement and orientation;
- Robustness to factors such as camera calibration errors, image noise, lighting variations or occlusion;
- Reduced requirements and interference with the scene (e.g. background separation, good bootstrapping).

Freedom in camera setup is important to allow more general applicability of the reconstruction. Strict or regular camera placements (such as e.g. regularly translating moving cameras) may provide some additional constraints that simplify some calculations. However, the probability of having parts of the images that are aligned with epipolar lines is high in many of those cases (e.g. horizontally aligned cameras capturing a scene with horizontal lines). This also includes some cases of small baseline methods such as SfM, where camera translation is implicitly small. Also in the case of images pairs or triplets-based reconstruction, the probability of ambiguity due to the alignment of parts of the image with epipolar lines may also be high. An irregular camera placement coupled with the use of multiple cameras typically provides more information and disambiguating power about the scene to be reconstructed, as opposed to regularly placed cameras. Only a few methods allow that, and those that do are typically highly sensitive to camera calibration.

Most reconstruction results suffer from errors and outliers that are treated with strong smoothing or regularization. This normally also leads to the loss of detail in

	Camera Placement			Robustness to				
	Freedom in orientation	Allows large baseline	Robustness to errors in camera calibration	Highlights, light changes	Partial Occlusion	Repetitive Patterns	Type of reconstructed data	Other Remarks
Fixed Nr. of views								
Feature	+	+	-	+	-	i	P,L	
Correlation	-	-	-	-	-	i	DM,S	
Multi-view								
Cumulative								
Structure from motion								
Pollefeys	0	-	i	-	-	-	DM,S	
Zisserman	0	-	i	+	0	+	P,L	
Seitz/Dyer	0	-	i	-	-	+	P,L	
Rother	+	+	i	i	i	i	P,L	(a)
Multi-baseline								
Tsai	-	-	i	-	-	i	DM,S	
Okutomi/Kanade	-	-	i	-	-	+	DM,S	
Kanade	0	-	i	?	0	+	DM,S	
Mellor	+	+	-	-	+	i	DM	
Cox	-	-	i	0	+	i	DM	
Szeliski/Weiss	-	-	i	+	0	i	P,L	
Jung	-	-	i	i	i	i	P,E	
Bauer	-	+	i	+	i	i	P,E	
Volumetric	+	+	-					
Visual Hulls	+	+	-	+	-	+	S	(b)
Szeliski	+	+	-	+	-	+	S	(b)
Matusik	+	+	-	+	-	+	S	(b)
Voxel Coloring/Carving	+	+	-	-			V	
Seitz/Dyer	+	+	-	-	i	i	V	
Kutulakos/Seitz	+	+	-	-	i	i	V	
GVC	+	+	-	-	i	i	V	
Others	+	+	-	-	i	i	V	
Other								
Collins	-	+	0	0	i	i	P	
Taillandier/Deriche	0	i	-	+	i	i	L	
Kawamoto/Imiya	+	+	i	i	i	i	P,L	
Model-Based								(c)
Fua	+	-	i	i	i	i	S	(c)
Zhang/Seitz	+	+	i	i	i	i	S	(c)
Kahl/August	+	+	i	+	-	i	L	(c)
Faugeras/Keriven	i	-	i	-	i	i	S	(c)
Spatio-temporal								
Various	0	-	i	i	i	i		

+	Good	i	Inconclusive	P - Points	(a) Correspondence
0	Fair	(blank)	Unknown	L - Lines	not addressed
-	Poor			E - Edgels	(b) Requires back-
				DM - Depth map	ground separation
				S - Surfaces	(c) Requires good
				V - Voxels	3D initialization

Table 2.2: *Comparison of reconstruction methods*.

multiple parts of a scene, especially if separate objects exist in the scene, and in other places where discontinuities occur. Occlusions are also a common problem, that is dealt in ways either too naïve (e.g. ignored) or too complex.

The main motivation behind this thesis was the search for a method that could be suitable to multiple and wide baseline settings and robust to camera calibration errors and other errors sources, as well as occlusion situations, with as little interference in the scenes as possible. Such a method will be proposed in Chapter 3.

Chapter 3

Multi-view 3D point reconstruction using contours and distance functions

3.1 Introduction

In Chapter 2, key concepts underlying 3D reconstruction were presented, together with highlights of the major issues that separate theory from practice. Existing reconstruction methods were also reviewed and compared, and it was observed that some important issues are still open or in some cases are difficult to tackle, namely:

Camera placement In many cases, camera placement is limited to specific setups, or requires small baseline.

Scene requirements Some methods require strong constraints on the scene, such as a distinguishable background, or no lighting variations.

Occlusion situations In some cases, total or partial occlusion of an object leads to incorrect reconstruction of parts that are actually visible.

Reduced robustness to errors The process of reconstruction is subject to multiple error sources, ranging from image noise to errors in the camera calibration. The robustness of reconstruction methods to such errors sources is in many cases low.

Requirements in camera placement are normally imposed to facilitate the use of specific geometric constraints, or to reduce problems in matching when there are strong perspective effects.

Requirements on the scenes translate either into a form of *a priori* knowledge of the scene (e.g. knowing that there is a single object, and that a blue background may be removed) or as a controlled reduction of the uncertainty (e.g. assuming color consistency between images). When assumptions are too restrictive or requirements are too intrusive, the usability of the method may be compromised.

Occlusion situations can be a significant problem, especially in cases where the ordering of the images is important and correspondence is established pair-wise or

triple-wise, such as in SfM techniques. In these cases, an entity that is visible in small subsets of consecutive images does not fully benefit from all the information available, as correspondences are only established within the subset, but not established between the non-consecutive sets; each subset of consecutive images implicitly ends up treating that entity independently. Furthermore, when an entity from an image I^0 is being reconstructed using another image I^1, and it is occluded in this second image, it is most likely that an incorrect correspondence will be established, resulting in an incorrect correspondence. Occlusions are therefore a typical source of outliers in reconstruction.

Robustness was defined as the degree to which a system can function correctly in the presence of invalid inputs or stressful environment conditions [IEEE 90]. In the context of this work, the correct functioning of a reconstruction algorithm is considered to be its ability to provide a 3D representation of a scene captured by multiple images, with a fidelity that degrades gracefully as the inputs or environment degrades. Invalid inputs correspond to error sources in the inputs (e.g. camera calibration errors, noise in the images, etc.). Stressful conditions can be less favorable yet correct conditions, such as a large number of occlusions occurring in many images of a scene, or a high number of highlights.

Therefore, robustness is not only associated to error sources, but also to predefined requirements such as camera placement and other scene requirements. The less constrained the environment is, the more robust the method has to be, in order to cope with uncertainty.

The goal of this thesis is to present a contribution for a better integration on the treatment of these issues in the form of a robust method. This method brings together different concepts based on their properties in dealing with the aforementioned issues. Those concepts include:

Contour reconstruction Contours are better suited to wide baseline reconstruction than e.g. blocks. They are also less sensitive to lighting and color variations than entities used in correlation-based methods.

Reconstruction using multiple images simultaneously The integration of information from multiple images provides statistical robustness, and helps to reduce ambiguity in correspondence.

World space formulation By using a single coordinate system to represent candidates from different images, geometric constraints such as the epipolar constraint are implicitly taken into account, and integration of information from different images is facilitated.

Weighted voting mechanism The reconstruction is based on a voting mechanism, such that the candidates for reconstruction are evaluated based on votes collected from different images. This implicitly helps in dealing with cases of missing information such as occlusions, that do not vote, but also do not compromise the candidate. The votes a candidate receives are weighted based on distance to contours. This provides a controlled tolerance to offsets caused by various sources, namely camera calibration errors and contour detection errors.

The combination of these concepts, and in particular the use of distance functions as the basis a metric for correspondence, make this method innovative, and results presented in Chapter 5 show that it is possible to successfully and robustly reconstruct points using this method.

The method proposed in this thesis assumes that camera calibration has been performed. There is no requirement for an exact camera calibration, and a limited number of cameras may even have a high calibration error. There are no restrictions to the baseline, or camera path/orientation.

The method is based on the localization of contours, hence there are some considerations about the scene. It is assumed that the part of the scene to be reconstructed is static, i.e. the reconstruction can only be correctly performed for non-moving objects.

In voxel-based or correlation-based methods, color or pattern changes caused by noise or lighting variations (e.g. fluorescent lighting, varying sunlight, automatic white balance on digital cameras) will lead to failure in consistency checking or matching. Since the method presented here is not color-dependent, the lighting conditions between the scene's images may vary, as long as contour detection is not affected negatively.

There are no strict restrictions regarding highlights as long as these do not interfere severely with contour detection.

The remaining of this chapter is structured as follows: contour points, the methods to obtain them and their use in the context of reconstruction are discussed in Section 3.2. This is followed by the description of the proposed contour point reconstruction algorithm, in Section 3.3. Section 3.4 contains two extensions to the algorithm to further increase robustness, while in Section 3.5 a method to reduce computational load is suggested for less demanding applications. The chapter ends with final remarks in Section 3.7.

3.2 Contour points in the context of reconstruction

An image of a scene comprises a series of color or intensity variations. High variations (e.g. above a certain threshold) give rise to what will be referred as *image discontinuities*. These *image discontinuities* originate from a variety of sources, such as geometric discontinuities, colored textures on a surface, or noise in the input images.

Not all the types of discontinuities have the same relevance for 3D reconstruction. The first part of this section comprises a classification of different types of discontinuities and what their importance is in terms of reconstruction.

The second part of this section focuses on the process of identifying image discontinuities using *contour detection* or *contour extraction* algorithms. The outcome of such algorithms are *contour maps*, that will be used as inputs for the point reconstruction algorithm presented in Section 3.3.1.

The third part presents the advantages and practical issues of corresponding contour points for 3D reconstruction.

3.2.1 Classification of discontinuity sources

The discontinuities existent in images can be originated from different sources. Some of those sources are *view-dependent* i.e. although they seem to relate to a particular position in the world when seen from a point of view, they will not be related to the same world position if seen from another view. Take as an example a light bulb's reflection on a static snooker ball: the bright spot seems to move over the ball's surface if the viewer moves, although both the ball and the light are static.

Given that view-dependent sources of discontinuities are in general not present on the same 3D location in more than one image, the amount of 3D information provided for reconstruction is not only low, but it also leads to inconsistency in correspondence.

On the other hand, *view-independent sources* of discontinuities are consistently located on the same position in the world, regardless of the view point. Such sources are the most relevant for reconstruction.

The following is a possible classification for image discontinuity sources according to view dependence and associated geometric discontinuities (if applicable) (see also Table 3.1 and Figure 3.1):

View-Independent Discontinuities that are associated to specific points or properties of the scene, and have a fixed 3D position (assuming a static scene).

- **Edges** 3D edges of objects, resulting from their geometric discontinuities (e.g. the edge between two adjacent faces of a cube).
- **Textures/Shadows** Variations in the apparent coloring of a surface caused by different textures or cast shadows.

View-Dependent Discontinuities whose source does not have an absolute position in the scene, and rather depends on the camera position. Possible sources for this type of discontinuities are:

- **Apparent Contours** Some occlusion situations may give rise to contours whose position is dependent on the viewing position. These situations occur when the occluding object has a smooth-curved surface that is tangent to the viewing direction. These contours are called *apparent contours*, as they occur even though there is no actual texture, lighting or geometry discontinuity in the surface of the objects.
- **Highlights/Reflections** Highlights or reflections present in specular surfaces.
- **Noise** Image discontinuities resulting from the image acquisition process.

View Dependence	Origin of image discontinuity	
	Geometric discontinuity	Non-Geometric discontinuity
Dependent	Occlusion	Highlights, Reflections, Noise
Independent	Edges	Textures, Shadows

Table 3.1: *Sources of image discontinuities*.

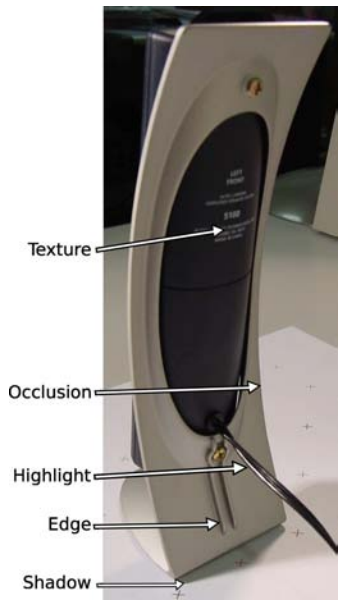


Figure 3.1: *Different types of image discontinuity sources* .

3.2.2 Contour extraction

As stated before, view-independent sources of image discontinuities are most important for image-based reconstruction. Unfortunately, there is no trivial way to distinguish, for a given image discontinuity, between the types of sources that caused it.

Image discontinuities are identified using algorithms known by the names of *contour detection*, *contour extraction*, *edge detection* or *edge extraction*[1]. These can be seen as operators that are applied to an image and produce a *contour map*. This binary map has a discrete grid, in most cases with the same dimensions of the original image, in which points representing meaningful image discontinuities are assigned a value of "true", and other points a value of "false".

There is not an exact definition of what a meaningful image discontinuity is. The search for the optimal edge detection algorithms, and for the optimality criteria itself, is in the origin of the vast literature in the field.

In general, one basic assumption lies at the heart of edge detection algorithms: meaningful discontinuities are associated to points with strong variations of color or

[1] Although some authors distinguish between edges and contours, this distinction will not be made here.

intensity. How to detect variations and how to decide which are strong enough are the questions that are answered differently be each method.

Most of the methods explicitly apply *smoothing* to the images, either before the edge detection itself, or implicitly in the detection process. Smoothing is expected to reduce the effect of noise. In most cases, the smoothing function is based on a Gaussian function, due to some of its properties, such as its separability (the ability to decompose the smoothing as two convolutions with orthogonal kernels), which are discussed e.g. in [Fors 03].

The most relevant edge detection algorithms can be divided in two major classes:

Gradient-based These methods detect variations by computing an approximation of the *gradient* or *first derivative* of an image, based on the color or intensity differences between neighboring pixels in the vertical and horizontal directions. Such computation is implemented by convolving the image with an appropriate kernel (an example can be seen in Figure 3.2, center). Two types of processing may be performed on the gradients, after their computation, to obtain a contour map:

Thresholding In this case, the absolute gradient values are simply thresholded, such that points with a high gradient value are marked as contour points, ant the others are rejected. An examples of these methods is the Roberts operator [Robe 65].

Gradient analysis These methods perform a more detailed analysis of the gradient, to try to extract thin lines that better approximate the contours of the image. One of the most known methods is the Canny operator [Cann 86], which applies techniques such as *non-maximum suppression* to obtain thin lines, *edge following* to obtain chains of contours, and *hysteresis* (as a type of double threshold) to simultaneously reduce the number of spurious contours while not breaking strong contour lines that may have weaker gradient values in some points.

Second-derivative-based Instead of looking for extrema in a first derivative, these methods search for *zero crossings* of the second derivative of the image. If the values of the second derivative for points adjacent in the image have opposite signs, a zero-crossing can be found by interpolating on those values, providing a sub-pixel position for contours (Figure 3.2, right). Second derivatives are normally approximated by the Laplacian, partly due to the fact that it is rotationally invariant. An example of this class of algorithms is the Marr-Hildreth operator [Marr 80], that incorporates the smoothing function in the computation of the second derivative, and thus convolves the image with a Laplacian of Gaussian, to later detect zero-crossings.
Second-derivative methods behave in an undesired way in some situations. For instance, in junctions, where three or more contours lines should meet, the output of second derivative methods typically produces a "hook" effect: one or more lines are diverted from the others. A particularly bad case occurs at T-shaped junctions, as illustrated in Figure 3.3. This makes them less suitable for applications that require high accuracy in such feature points.

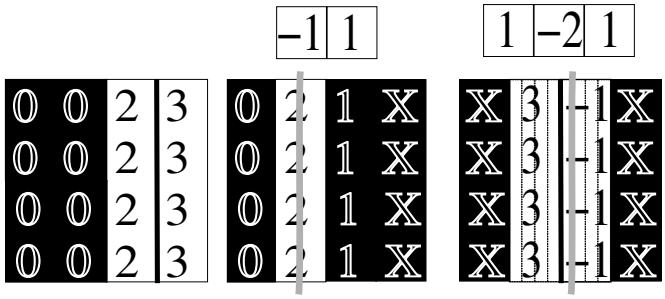


Figure 3.2: *Example of edge detection based on gradient vs second derivative*.
Left: Example of part of an image, with pixels of intensity 0, 2 and 3;
Center: A gradient-based operator for detecting vertical contours is applied to the image, by convolving the image with the kernel (-1,1) (centered on the left pixel, undefined values marked as 'X'); contours (marked as a thick grey line) are located at pixels of gradient value equal or above a predefined threshold (in this case, 2);
Right: A second derivative operator is applied to the image, using the kernel (1, -2, 1) centered in the middle; contours are detected by zero-crossings interpolated *between* pixels where the signal of the second derivative changes

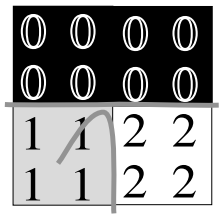


Figure 3.3: *Problems of second derivative methods at T-shaped junctions*.
Instead of obtaining a T-shaped contour in the area where three different colors meet, one of the contour lines is "diverted" (its prolongation's length is scale or threshold dependent)

Edge detection algorithms are not object of research in this work, nor is the presented reconstruction algorithm dependent on a particular edge detection algorithm. It is assumed that an edge detection algorithm is applied to the images, and that its output is an edge map where there are possibly some missing contours, some fake contours, as well as small offsets caused by incorrect response of the algorithm to some discontinuities, or due to the discretization of contour points into a grid.

Gradient-based techniques will be favored, since they do not suffer from the problems at corners and junctions that hamper the methods based on second derivatives. More specifically, the Canny [Cann 86] edge detector was the one mostly used, due to its properties referred above.

3.2.3 Advantages and practical issues in the correspondence of contour points

Using contour points has advantages for reconstruction that are recognized in other feature-based methods [Torr 00]: the sensitivity of contour points to perspective effects is low, thus being adequate for wide-baseline settings; and their sensitivity to mild lighting variations is also relatively low.

Some practical issues also arise regarding the correspondence of contour points.

On one hand, the attributes that can be used in a matching function to compare two contour points are mostly limited to their position in the image and in some cases (depending on the contour extraction algorithm used) the magnitude and orientation of the associated gradient. Using more information from the surroundings of contours in the original images, such as color or intensity information, would lead to the same limitations and problems of correlation-based methods. Furthermore, the impact of such limitations is usually higher precisely at discontinuities (e.g. the case of occlusion). The disadvantage is that that type of information has, by itself, less disambiguating power than e.g. the type of information used by correlation-based methods.

On the other hand, missing, displaced or false contours may occur due to noise or faulty contour extraction. Errors in camera calibration may also introduce displacements on contours.

Therefore, a reconstruction method based on contours should be able to efficiently deal with both the ambiguity introduced by the type of information available for correspondence, and also the possible errors in the contours. This will allow to benefit from the advantages of higher freedom in camera orientation and lighting variations than the one provided by existing methods.

3.3 Reconstruction of contour points

In Section 2.2, it was seen that there are major limitations in existing methods such as the strict camera placement, the requirement of special scene settings such as color consistency or separable background, and the assumption of low camera errors.

In the previous section, important advantages of using contours for 3D reconstruction were presented, namely the low sensitivity to wide baseline and to lighting variations. If the practical issues of contour correspondence regarding low disambiguating power and errors in contour extraction and localization would be overcome, contour points would be a suitable basis for a reconstruction method with significant freedom in camera placement, lighting variations and background.

An important goal of the work presented here was therefore to overcome to a great extent those issues and, by doing so, provide an implicit degree of robustness to noise and camera calibration errors.

Disambiguating power can be increased by integrating information from multiple images. When only two images are used, a point in one image may have high similarity

to n points of the other image. However, not all of those n likely correspondences will be consistent with a third image, and even less with a fourth, and so on. If an hypothetical correspondence is consistent with the information contained in multiple images, the confidence in that correspondence should be high. Conversely, incorrect correspondences will be inconsistent with a significant number of images.

This principle has been explored in different ways in multiple-baseline approaches, as referred in Section 2.2, but typically in limited baseline settings, using correlation-based methods, and in some cases inefficiently. It will be shown how match values between contour points can be efficiently integrated over multiple images.

The problem of false or missing contours is also partly overcome by the use of multiple images. Adding more images may lead to an overall increase of false contour points to be reconstructed. But by increasing the number of images, the probability of a false contour persisting in all (or most) images is reduced. On the other hand, the probability of an important visible contour being missed by contour detection in all (or most) images is also reduced. This can be seen as a mean of increasing robustness by integrating information from multiple images.

The problem of displaced contours, however, is not significantly improved by the use of multiple images. Contour offsets may be such that correct candidates for a correspondence may not intersect epipolar lines. This requires that a tolerance is provided for such offsets by e.g. considering a neighborhood of the epipolar lines when looking for correspondences. Typically, methods that do consider a tolerance, within a neighborhood of the epipolar line, value candidates in that range equally. Such a tolerance should value more contours that are closer to the epipolar lines than those that are farther away. An elegant and efficient approach to achieve this will be proposed, which integrates a candidate's distance to the epipolar line directly in the correspondence metric.

The next part of this section outlines the generic algorithm that integrates the concepts mentioned above. In the parts that follow it, each individual component is described in detail.

3.3.1 The main algorithm

As mentioned before, correspondence-based reconstruction methods typically search for correspondences in images, and the correspondences that have higher match values are used to estimate 3D coordinates.

In a two-image, small-baseline setting, the implementation of such a search using the epipolar constraint is relatively straightforward, but it is not sufficient for a robust reconstruction. Using a trifocal approach to verify ambiguous matches over a third image may improve the result if the ambiguity does not hold and if the correct candidate is also visible (or represented e.g. in the case of contours) in that third image, which may require low-noise situations. Methods that perform pair- or triplet-wise reconstruction incrementally, such as SfM, do not fully benefit from information from multiple images to reconstruct a given point or entity, as each pair or triplet is considered individually. By taking into consideration as many images as possible simultane-

ously, and in a common referential, rather than e.g. processing a pair individually and carrying the result to the next iteration, the information from multiple images is better integrated and explored.

For this reason, a *world-space formulation* will be used, such that the candidates are directly chosen from the view rays and defined in 3D, rather than being selected in 2D from the epipolar lines in each image. This is in a way similar to what happens in volumetric approaches. Such a formulation greatly simplifies the process of integrating information from multiple cameras simultaneously, as it will be shown later.

Assume a contour point (*c-point*) $\mathbf{p}_q^i$ chosen from an image I^i. A *set of candidate points* $\mathbf{v}_{q,x_1}^i,..,\mathbf{v}_{q,x_n}^i$ is selected from the 3D points of the view ray V_q^i associated with $\mathbf{p}_q^i$ and the corresponding camera $\mathbf{C}^i$ (Figure 3.4).

Each candidate must then be evaluated using a match function. This function will have to rate how good a candidate is for being selected as the reconstruction of $\mathbf{p}_q^i$. One solution would be to use the number of images in which that candidate projects on another contour point as the match value.

In an ideal situation, without contour offsets caused by noise or camera calibration errors, correct candidates would in general project on contours on most images – high match value – and incorrect candidates could coincidently project on contours in some images, but very few – low match value. This can be seen as a *voting* procedure, where each candidate collects votes from various images.

Figure 3.4 illustrates this process: each candidate $\mathbf{v}_{q,x}^i$ is projected into the images as a *projected candidate* $\mathbf{p}_{q,x}^{i,j}$. Each of the images contributes with an individual voting $v_{q,x}^{i,j}$ for the candidate $\mathbf{v}_{q,x}^i$, depending on the contour information contained in the image. These votes are accumulated over all images on a per-candidate basis.

Using 3D points as candidates, and votes accumulated over multiple images as match values has some important properties:

- Information from multiple images is automatically integrated and assigned to the same 3D point, without the need for explicitly tracking correspondences between pairs of images;
- By fusing information from multiple images, the implicit ambiguity inherent to contour point reconstruction is reduced;
- Epipolar constraints are implicitly used.

As robustness to contour offsets is sought, the enforcement of the epipolar constraint, implicit on the choice of candidates in the view ray, is too strict. Suppose that a contour in an image is slightly displaced, such that a candidate that should project on that contour actually projects right beside it. Due to this small offset, the candidate would not collect any vote. Some controlled tolerance to these offsets needs to be introduced, that differentiates candidates in the neighborhood of an epipolar line according to their closeness.

The solution proposed here uses a *weighted voting* mechanism to reward more accurate candidates: when a candidate is projected in an image, it gets a weighted vote

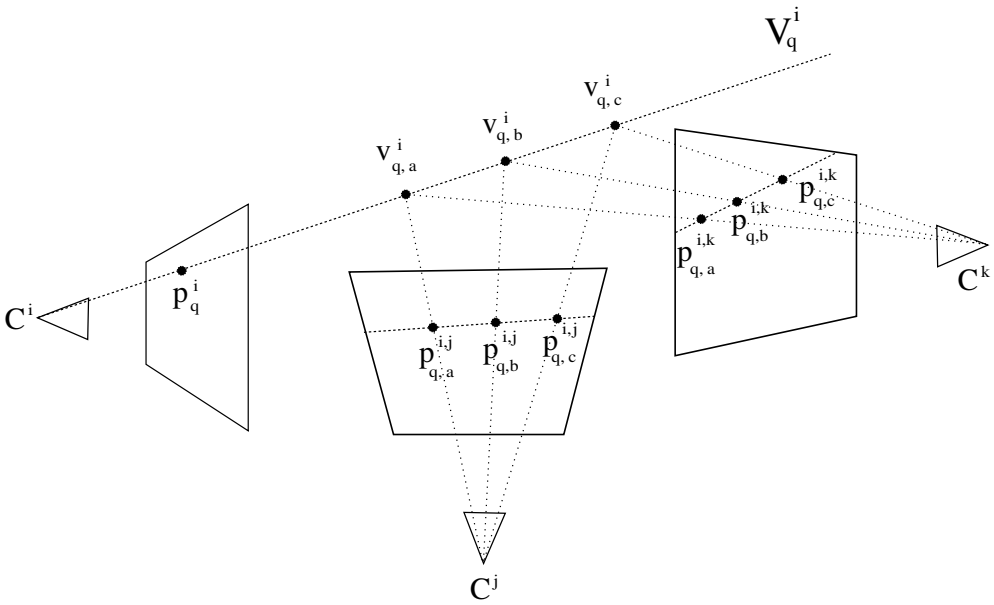


Figure 3.4: *Reconstruction using view ray candidates*.

related to its closeness to contour points. In this way, a candidate that is projected close to (or on) a contour gets higher voting than a candidate that is distant from contours. This approach is less strict than considering only candidates in the epipolar line, but also less liberal than giving equal importance to any candidate within a close neighborhood of the epipolar line. This weighted voting mechanism thus provides a degree of robustness to contour offsets.

Algorithm 3.1 describes the process in general. Details on the particular components of this method are presented next. Section 3.3.2 describes how to select the candidates of the view ray. Section 3.3.3 proposes a voting function based on closeness to contours, and a method to efficiently compute it. After collecting and accumulating votes for all candidates and all images, the reconstructed point has to be chosen by analyzing the candidates' accumulated votes. In Section 3.3.4 it will be shown why the candidate with the highest voting is not always the best choice, due to some pathological situations that can occur. In Section 3.3.5 a method for selecting the winning candidate will be proposed, taking into account the aforementioned situations.

3.3.2 Selection of the candidate set

Typically, in a reconstruction, a region of interest in space can be identified outside which there is either no desire to reconstruct any geometry, or there is no information available to reconstruct. This region of interest provides boundaries for the parts of a view ray from which to choose candidates. Determining a meaningful *range* of the view ray from which the candidate set is selected is the first step to establish a set of candidates. The range is defined by a lower and upper limit, and may be expressed in terms of the distance to the camera – *depth*. If some region of interest or other bounding information is available on the scene, this can be used to limit the range of

- Let V_q^i be the view ray of $\mathbf{p}_q^i$ associated to $\mathbf{C}^i$
- Choose a set of candidate world points $\mathbf{v}_{q,1}^i$, $\mathbf{v}_{q,2}^i$, ... from V_q^i
- For each image I^j $(j \neq i)$
  - For each candidate world point $\mathbf{v}_{q,x}^i$
    - Project $\mathbf{v}_{q,x}^i$ in I^j as $\mathbf{p}_{q,x}^{i,j}$
    - Let $\upsilon_{q,x}^{i,j}$ be the voting of $\mathbf{p}_{q,x}^{i,j}$ in I^j
    - Add $\upsilon_{q,x}^{i,j}$ to the total voting $\lambda_{q,x}^i$ of $\mathbf{v}_{q,x}^i$
- Select the r-point $\mathbf{r}_q^i$ corresponding to the reconstruction of $\mathbf{p}_q^i$ from the candidates with higher total number of votes $\lambda_{q,x}^i$

Algorithm 3.1: Reconstruction using view ray candidates

the candidate set. If it is not available, it might be estimated by e.g. intersecting the cameras' view frusta.

Due to the limitations in precision and computational resources, only a discrete and finite set of the points within a view ray's range can be evaluated. Two options for selecting a candidate set from the initial range would be (1) to start with a small candidate set and iteratively add more candidates, based on the voting of the initial sets, or (2) to densely sample the initial range with a fixed number of candidates.

The first option is in line with function maximization algorithms, such as e.g. the Brent algorithm [Bren 73]. Such algorithms attempt to find the maximum of a function by iteratively evaluating the function in a set of points and use those results to determine the next point to evaluate. Ideally each new guess is closer to the function's maximum, and when the algorithm meets a convergence criteria, the maximum is found.

Function maximization can be computationally expensive, and for this reason multiple algorithms were developed to try to reduce the number of function evaluations required to automatically find a global maximum. However, when the function is not smooth enough, simpler maximization algorithms may be easily trapped in local maxima. More complex maximization algorithms can improve on this, at the cost of higher computational load, and in some cases requiring the availability of function derivatives. Therefore, the use of maximization algorithms may be inadequate for very irregular functions, with multiple maxima, and for which there is no analytic definition. In Section 3.3.4, it will be shown that the voting function for a point is indeed irregular and contains several local maxima.

Furthermore, it will also be shown that the global maximum may correspond to an incorrect solution for the problem. This points in the direction of a more exhaustive search for maxima. Intuitively, brute force approaches are not highly efficient, as they require more function evaluations than e.g. a function maximization algorithm. Nevertheless, even if a function maximization would provide the best solution, when the cost of evaluating the function for multiple points is very low, brute force can prove to be more efficient than a robust maximization algorithm. It will be shown in Chapter 4

that indeed the evaluation of a correspondence function for a set of candidates regularly sampled from a range of a view ray can be made quite efficiently. The sampling density will influence the quality of the reconstruction, and the computational load. This trade-off is demonstrated and analyzed in the results presented in Section 5.3.7.

3.3.3 Voting function

Distance as the basis of the voting function

The simplest voting function for corresponding contour points is a binary one, that simply indicates if a candidate is a contour point or not. A candidate projected in an image will collect one vote if it is on a contour and no votes otherwise (Figure 3.5(a)).

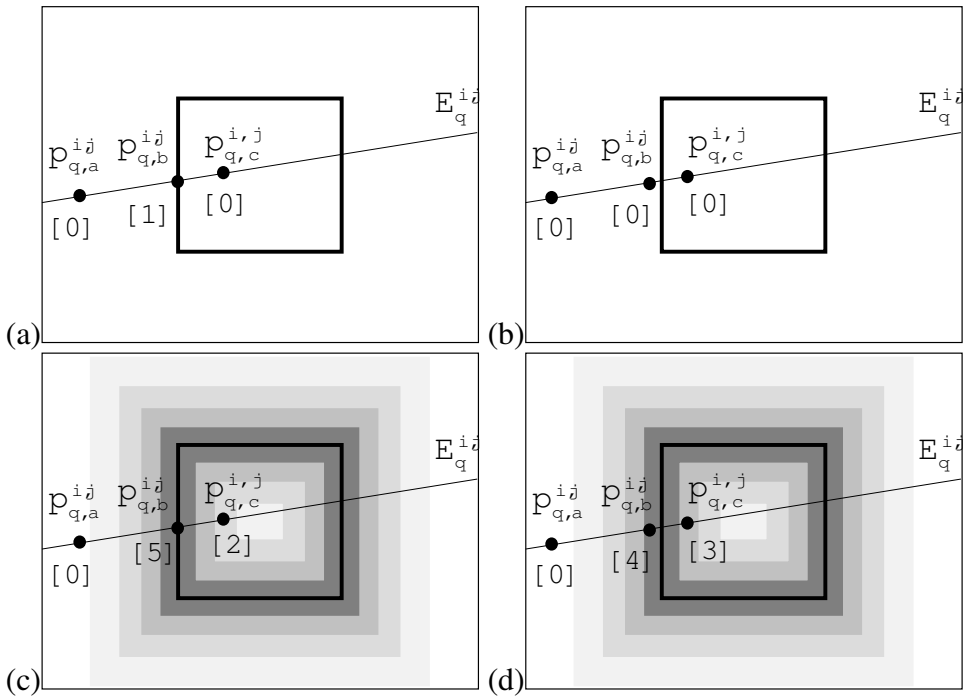


Figure 3.5: *Binary-based vs distance-based voting function.*
The black square represents a contour, and voting values for each case are represented in square brackets.

(a) Binary voting: The point lying on a contour collects one vote, while other points collect zero votes.

(b) Binary voting with offsets: A small offset caused by errors could cause all candidates to lie outside contours, thus receiving no votes.

(c) Distance-based voting: The votes are weighted according to their closeness to contours, represented by the different shading - points on contours have maximum closeness (e.g. a maximum weight of 5 votes) and points very far, on the white, area have a null voting.

(d) Distance-based voting with offsets: In this case, even though no candidate is on a contour, close candidates may still collect votes.

However, this approach is too strict. It does not take into account the offsets that contour points suffer caused by discretization, noise, aliasing or faulty contour extraction, and the offsets of epipolar lines due to errors in the camera calibration. In these cases, it is possible that no candidate gets a vote (Figure 3.5(b)).

To account for these error situations, the voting function should not be so strict, but allow a finner-grained voting. A candidate that is projected close to (or on) a contour should get higher voting than a candidate that is distant from contours. This can be achieved by weighting the voting of a candidate with its *closeness* to a contour. By doing so, a candidate projected on a contour would collect the maximum voting, candidates close to a contour would also get a voting, yet lower than the maximum, and candidates very distant would get a very low or null voting (Figure 3.5(c)).

In the case of small offsets, good candidates that are projected close to a contour would not be ignored, as in Figure 3.5(b); they would still collect a significant number of votes (Figure 3.5(d)).

The voting function should therefore be weighted by the closeness of the projected candidate to a contour. One such voting function will be defined here: the *complement of the distance of the projected candidate to the closest contour*.

There are several advantages in the using a voting function based on the distance, as opposed to a binary voting. First, it introduces an implicit tolerance to account for missing, incorrect or displaced contour points.

Second, the distance to contour points can be easily and quickly computed using well-known algorithms for computing *distance transforms* [Borg 84], as will be shown in Section 3.3.3. Such algorithms produce *distance maps* where each point of the map indicates the distance to the closest point of interest.

In the context presented here, contour points are the points of interest. Therefore, a distance map computed from a contour map indicates, in each point of the map, the distance to the closest contour point.

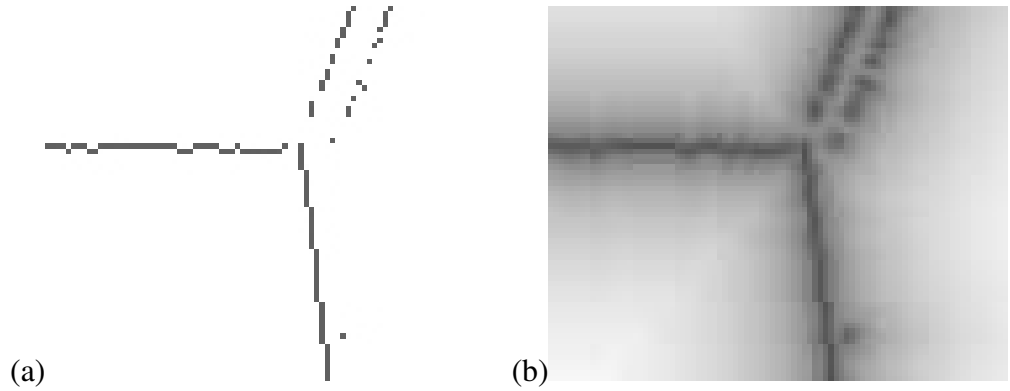
Since the distance to a contour is a metric independent of the contour point being reconstructed, distance maps can be independently pre-computed, and only once for each image, regardless of the number of points to be reconstructed and the number of candidates per point.

Distance transforms and a method to produce distance maps will be introduced next, followed by the description of a method to convert distance maps into *vote maps*.

Pre-computation of distance maps

Computing the distance between a point in an image and its closest contour can be done in an efficient way for all points of an image using *distance transforms* [Borg 84].

Distance transforms take as input an initial map, where a set S_i of points of interest are marked. In this case, the points of interest are the contour points. The distance transform produces a *distance map*, where each point has stored the distance to the closest point of interest – the distance to the closest contour, in this case – according to Definition 3.1.



(a) (b)

Figure 3.6: *Example of a binary map and the corresponding distance map.*

(a) Black pixels represent pixels of interest

(b) Darker pixels are closer to pixels of interest than lighter ones (intensity scaled for better visualization)

Definition 3.1 *Distance to the closest point of interest of* S_i

$$distance(p) = \min_{s \in S_i} ||p - s||$$

□

An example binary map and the resulting distance map are shown in Figure 3.6.

There are several algorithms for computing distance maps. The better known and most commonly used is attributed to Bourgefors [Borg 84], and will be used here.

It consists of a two-pass algorithm that is applied to an initial map created from an original binary image. In this initial distance map, the points of interest – contours in this case – are marked with the value 0 (zero) representing distance 0. Other points are marked with a very high value, representing unknown distance (Figure 3.7(a)).

Each pass runs in opposite directions over the distance map e.g. one pass from left-to-right/bottom-to-top, and the other from right-to-left/top-to-bottom.

At each pass, each point is labeled with the smallest distance to points of interest previously visited. Therefore, after the first pass it is guaranteed that the distance to the closest point of interest that is below or to the left of a given point is known (Figure 3.7(b)).

The second pass completes the distance map. At each point in this pass, the distance to the closest point of interest that is above or to the right is known, but also the distances obtained from the first pass. Therefore, only points that have points of interest closer than those identified in the first pass are updated (Figure 3.7(c)). After the two passes, distances will have been propagated in all directions.

The algorithm takes two parameters: the distance between two adjacent horizontal or vertical pixels d_a, and the distance between two diagonal pixels d_b. These parameters define the type of *distance metric* used, and correspond to the *distance mask*. In

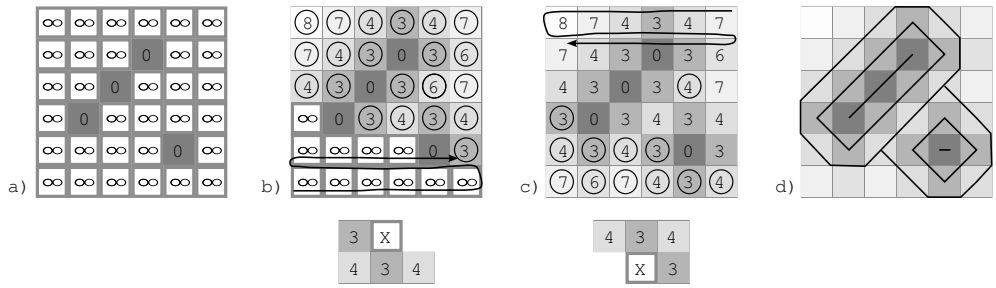


Figure 3.7: *Distance map computation*.

(a) Initial distance map: Points of interest marked with distance 0, other points with infinity.

(b) First pass performed left-to-right, bottom-to-top, with the distance mask provided under; updated values marked with a circle.

(c) Second pass performed in the opposite direction, with the symmetrical distance mask; notice that a value updated in the first pass was also refined in this pass.

(d) An intensity-coded version of the final distance map, with an approximation of isocontours (the lines of equal distance) superimposed as black lines.

Figure 3.7, the metric used was of $d_a = 3$ and $d_b = 4$.

Different distance metrics can be used, depending on the application and requirements. For instance, the "Manhattan" or "city-block" distance is computed using $d_a = 1$ and $d_b = 2$. For an approximation of the Euclidean distance that keeps a low error at larger distances, [Borg 84] suggests values of $d_a = 0.955$ and $d_b = 1.3693$ or, if integer arithmetic is preferred $d_a = 3$ and $d_b = 4$ (in this case, the final results will be scaled by an approximate factor of 3).

Although better approximations of distance can be achieved by using floating-point values for the metric or considering larger neighborhoods for computing the smallest distance for each point (i.e. larger distance masks), in many applications this is not required. This is true for the voting function sought in this section, where the focus is more on obtaining a relative weight rather than an accurate distance, and higher distance values are to be clamped anyway. In this case, the higher computational load associated to floating-point masks (with the additional storage requirements for the maps) or larger neighborhoods does not compensate the minor increase in accuracy at close range.

For these reasons, the distance maps in this work will be based on integer-valued distances, such as the "city block" ($d_a = 1$, $d_b = 2$) and Bourgefors' ($d_a = 3$, $d_b = 4$).

Conversion of distance maps to vote maps

It was referred earlier that the voting function should produce votes for points that are close to contours, by weighing them according to their closeness to a contour. This would reward candidate points that project very close to contours, while still providing tolerance to points that are at some acceptable distance.

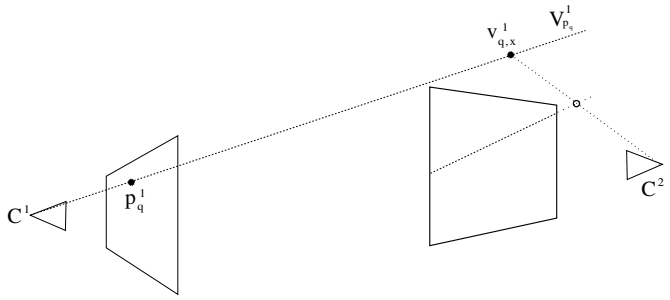


Figure 3.8: *Out of frustum*.

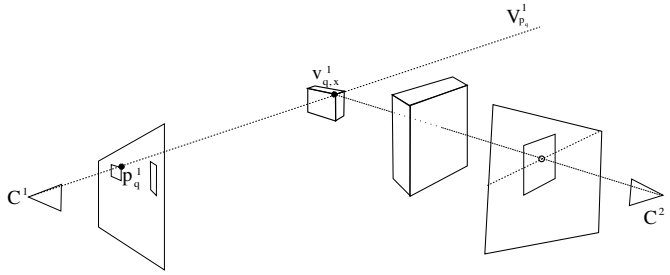


Figure 3.9: *Occlusion of a candidate*.

However, this tolerance should be limited, such that there is control to the magnitude of the offsets that are acceptable. There should be a limit to how far a candidate can be from a contour and still receive votes. This ensures a type of robustness: if a candidate projects at a close distance from a contour, it still collects an amount of votes, albeit small; if it is too distant it does not collect any votes.

Another situation where vote limiting is important is when a candidate projects outside the limits of a given image (meaning the candidate lies outside the view frustum of the camera). In these cases, there is no useful information available on that candidate in that view (see Figure 3.8).

At most, one can determine the maximum distance at which the projected candidate may be in that image from a contour. But this information is not useful for reconstruction purposes. If the image was larger, the correct contour could coincide or be very close to the projected candidate. A similar problem occurs in the case of occlusion of a correct candidate in a given image. In this case, the candidate's projection could lie at a considerable distance from a contour (Figure 3.9).

In the previous section, a method to pre-compute the distance to the closest contour was presented. The "closeness to a contour", required for the voting function, can be defined as the complement of the distance to the closest contour.

It is therefore proposed that the voting function is computed as a complement of the distance map, *clamped* to a range that is considered acceptable in order to reduce the influence of out-of-frustum and occlusion situations, while still providing enough tolerance to contour offsets and camera calibration errors. Being R the *vote range*, the

vote value for a given point x in the vote map is defined according to Equation 3.1:

$$v_x = \begin{cases} R - distance(x) & \text{if } distance(x) < R, \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

This clamping approach maintains one of the main advantages of the use of the distance function – tolerance to errors and contour displacement – while limiting that tolerance to acceptable values, and attributing a null weight to undetermined situations like occlusion or out-of-frustum cases.

3.3.4 Vote analysis - multiple high maxima

After accumulating votes for all candidates of a point, a winner needs to be picked. The obvious choice for the reconstructed point would be the candidate with the highest voting. However, due to the multiple error sources and some pathological situations, the voting function's curve may present multiple high maxima, with voting values close to the highest voting. Moreover, the correct candidate may not have the highest voting, although it is close to the highest.

An example of a typical voting curve is presented in Figure 3.10. As mentioned in Section 3.3.2, maximization algorithms are not adequate for this type of functions with multiple maxima and the need to detect and analyze multiple high maxima.

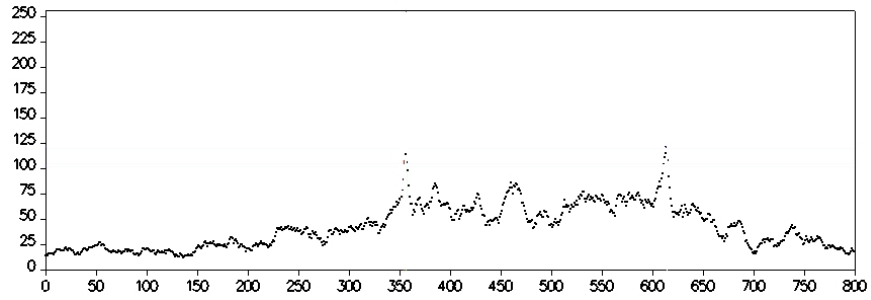


Figure 3.10: *Example of a voting curve.*
The curve exhibits multiple local maxima, at least one with values close to the global maximum

Two particular situations may cause this multiple high maxima situation: *cross voting* and *overlapped points*.

Cross voting

The phenomena of *cross voting* occurs when one candidate repeatedly accumulates high voting from contours in two or more images that actually correspond to different world points (Figure 3.11(a)).

Ideally, the correct candidate would always be projected on a contour in all images, granting it the highest voting possible, so cross-voted candidates would have in the vast

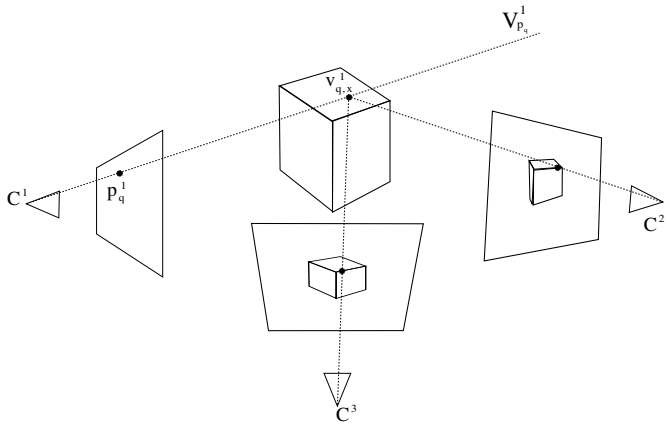


Figure 3.11: *Cross-voting problem*.
The candidate point on the top surface of the cube gets votes from contours corresponding to the projection of different points.

majority of cases a lower voting, and only in very coincidental situations (when they would lie on as many contours as the correct candidate) they would have the same voting.

However, factors like occlusion and faulty, noisy or displaced contours yield a sub-optimal accumulated voting for the correct candidate. If a cross-voted candidate happens to coincidentally project on contours in a significant number of images, it can have a voting higher than the correct candidate. This may even happen for more than one of the candidates for a given point. The result is a set of well-defined maxima in the voting function, one for each of those candidates, close to the global maximum (and including the correct candidate, which might not be the global maximum).

Particular cases of this degeneracy occur when there are repeating patterns aligned with epipolar lines in multiple images (e.g. when cameras are horizontally aligned and the scene contains an horizontal pattern). In the limit, if there is a contour line aligned with epipolar lines of multiple images, it yields an interval of depths with very high and close votings: a "*plateau*" (Figure 3.12).

The probability of such occurrences is high in settings where the camera placement or movement is regular, or camera positions are very close. In the general case, where cameras are not aligned with a particular direction, such situations are not as common. An irregular camera placement, together with the use of multiple cameras, reduces the probability of such alignment, and typically provides more information and disambiguating power about the scene to be extracted, as opposed to e.g. regular translations of the cameras. Still, these situations may occur, and some measures to reduce their effect in the reconstruction will be proposed in Section 3.3.5.

Overlapped points

If two w-points $\mathbf{w}_a$ and $\mathbf{w}_b$ belonging to object discontinuities are aligned in the same view ray associated with camera $\mathbf{C}^i$, they will project in the same contour point

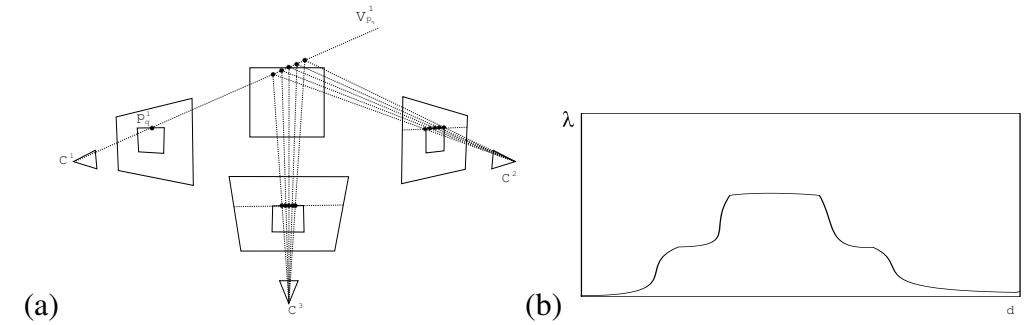


Figure 3.12: *Plateaus*.

(a) The epipolars of $\mathbf{p}_q^1$ in $\mathbf{C}^2$ and $\mathbf{C}^3$ are aligned with the horizontal contour; all candidates get maximum voting from both images.

(b) The vote chart as a function of depth exhibits plateaus in the regions corresponding to contour/epipolar overlap.

$\mathbf{p}_q^i$ in I^i (Figure 3.13). In many cases, these situations represent occlusions of objects with non-parallel contours, which normally result in junctions in the image contours. When reconstructing $\mathbf{p}_q^i$, it is expected to find at least two candidates with high accumulated voting, namely for the candidates corresponding to $\mathbf{w}_a$ and $\mathbf{w}_b$. It is not trivial to distinguish this situation from the multiple local maxima originated by cross voting. In particular, as in cross voting, any of the candidates corresponding to the overlapped points may be the highest voted. Possible solutions will be discussed in Section 3.3.5.

3.3.5 Vote analysis - winner selection

The conditions described – cross voting and overlapping points – may lead to the existence of multiple, distinct high maxima in the voting function of a point, with voting values close to the global maximum. The result is a highly irregular function curve (Figure 3.10), that is not analytically defined, and for which the global maximum may be an incorrect choice for the sought reconstructed point. This requires that some of the high-valued maxima other than the global should be taken into account when choosing the winning candidate.

Two parameters could be proposed to select which maxima should be considered (if any).

One is the minimum voting threshold for the global maximum. If the global maximum is below this threshold, then the voting of any maxima (including the global) is too low for reconstructing the point with confidence. Such points should be rejected.

The second parameter is a tolerance range relative to the global maximum. Only local maxima whose voting value is sufficiently close to the voting of the global maximum should be considered for further analysis. This range will be defined as a percentual value of the votes of the global maximum: *Vote Percent Threshold*. For instance, if the Vote Percent Threshold is set to 90%, for a point whose global maximum has 100 votes, all local maxima with 90 votes and above should be considered. If there

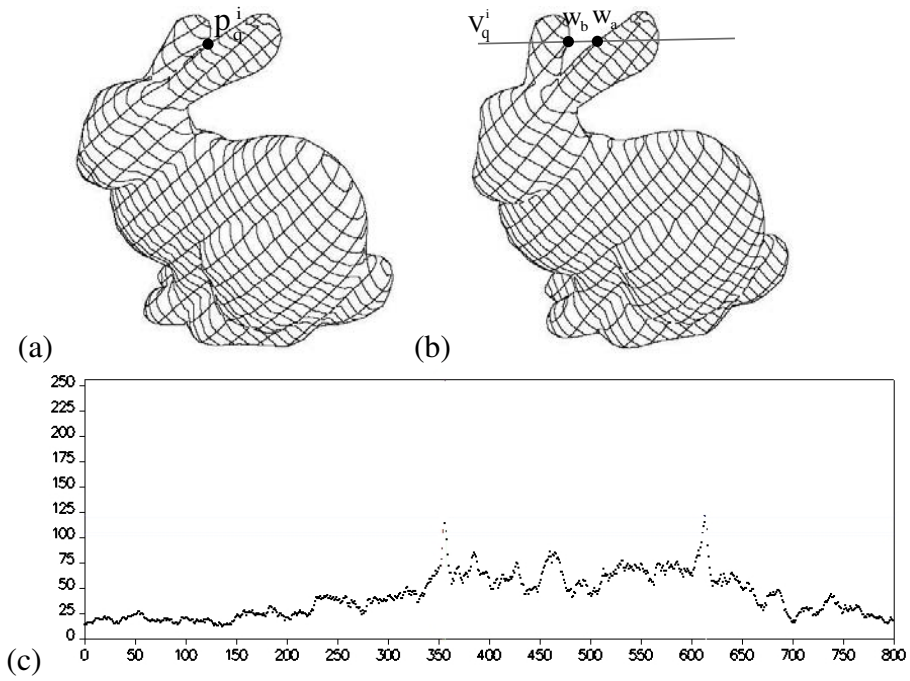


Figure 3.13: *Overlapped points*.

(a) Reference view and selected point

(b) A view showing the view ray and the points at the surfaces (the point on the right ($\mathbf{w}_a$) is the correct one)

(c) Graph of the votes as a function of the candidates; notice the two higher peaks corresponding to the points on the two ears

are no local maxima within that range, then the global maximum is considered clearly distinct, and the corresponding candidate is selected as the winner.

If, on the other hand, other local maxima exist within range, there are several possible approaches for evaluating the point and its candidates:

Reject the point The existence of multiple high maxima suggests the occurrence of cross voting or overlapping points. This conservative solution would result in a high number of rejected points.

Accept all relevant maxima In this case, the maxima are assumed to correspond to multiple overlapped points. This option would generate as many reconstructed points as the number of maxima, and consequently many false positives, especially in cases of cross voting.

Accept the global maximum As mentioned before, this option leads to inconsistent results, both in cross voting and overlapped points situations. In these cases of multiple high-valued maxima, the probability of each of them being the correct candidate (independently of being the global maximum) is very similar.

Accept the high maximum closest to the camera This option, although not optimal, is acceptable for the case of overlapped points, as it has the highest probability

of choosing the visible candidate. If other high-voted candidates actually correspond to world points, they are likely to be reconstructed from other images. In cross-voting situations, the probability of this choice being correct is similar to the option of choosing the global maximum.

The two first options are respectively too conservative and too liberal. Accepting the global maximum is suitable for many cases but it still leads to false positives in cross voting and overlapped points situations. Without performing any additional checking, the option of choosing the high maximum closest to the camera is the best compromise, since it correctly reconstructs most cases of overlapped points, and some of the cross-voting cases. To improve over these results, it would be necessary to further validate the candidates associated to these maxima. This is what is proposed in the robustness extensions presented in Section 3.4.

Otherwise, the set of heuristics here discussed already provides some satisfactory results. The criteria are here summarized:

Minimum Voting If the global maximum has a very low accumulated voting, the point should be rejected.

Global maximum close range All the maxima with voting values in a close range of the voting value of the global maximum should be considered potential candidates for reconstruction. The limit of this closeness is defined by votePercentThreshold.

Closest candidate to camera From the maxima contained in the global maximum close range, the candidate closest to the camera is selected as the reconstructed point.

These heuristics should be applied after the accumulation of all votings for all candidates of a point, using Algorithm 3.2.

- Let $M_q^i = \max_x \lambda_{q,x}^i$, for a given $\mathbf{p}_q^i$
- If $M_q^i < minVotes$
  - Reject $\mathbf{p}_q^i$ as invalid
- else
  - Let $S_{max} = \left\{ \mathbf{v}_{q,x}^i \;\middle|\; \begin{array}{l} \mathbf{v}_{q,x}^i \text{ is a local maximum} \\ \lambda_{q,x}^i \geq M_q^i \times votePercentThreshold \end{array} \right\}$
  - Select the candidate $\mathbf{v}_{q,\bar{M}}^i \in S_{max}$ (if any) that is closest to camera $\mathbf{C}^i$

Algorithm 3.2: Winner selection algorithm

3.4 Robustness extensions

Robustness has been discussed throughout this thesis as an important requirement for reconstruction. The multiple error sources that affect the inputs have an influence in the quality of the reconstruction results. That influence depends on the amount of errors in the inputs and on the ability of the algorithm to detect and deal with them.

A robust reconstruction algorithm should be able to correctly estimate a significant number of points (or other entities) in the presence of acceptable errors such as noise in the images or camera calibration errors. However, as robust to errors in inputs as a method may be, there will always be some estimates will be incorrect. Furthermore, some ambiguities may also lead to incorrect estimates. In the case of reconstruction for e.g. visualization, such incorrect estimates can be significantly noticeable, even if in a small amount. Therefore, a robust method should also be able to identify and reject as many incorrect estimates as possible.

In the previous section, a method for reconstructing contour points using a voting function based on distance transforms was proposed. The voting function provides a degree of robustness for missing, faulty or displaced contours, by implicitly taking into account possible offsets of contours, and by being integrated over multiple images.

But in some cases, the voting function causes ambiguous results, as in the case of overlapped points or cross voting situations. Some solutions to make a good choice of the winner were suggested, such as the use of a minimum voting that reject points that do not have strong candidates, or the choice of the high maximum that is closest to the camera when multiple high maxima exist.

These solutions already provide acceptable reconstruction results, but for some applications, there is a need for a more robust solution that filters the erroneous points resulting from situations such as cross voting. To improve the robustness of the core method described in the previous section, two extensions are proposed in this section: *cross-checking* and *merging*. These extensions were published in [Rodr 04b].

3.4.1 Cross checking

One of the major sources of ambiguity is the occurrence of cross voting. Cross voting is caused by the coincidental projection of a candidate on contours corresponding to different points in different images. Although this produces a high voting value, the votes were not meant for that candidate. In cases of cross voting, there are high probabilities that a correct candidate will project on a contour in cameras which are *similar* to the reference camera, while an incorrect (cross-voted) candidate will not. This suggests that, in many cases, cross-voted candidates are distant from the correct candidates, and the images that cast the incorrect votes often are those with positions and orientations significantly different from the reference camera.

It is therefore assumed that, if a candidate fails to project close to a contour on similar cameras, the probability of being incorrectly reconstructed (e.g. by cross-voting) is high. Such points should be rejected. This is the basis of the cross-checking algorithm: if all the projections of a candidate in similar cameras are below a pre-defined

pixel distance – *cross-check threshold* – from a contour, then the candidate is accepted. Otherwise the candidate is rejected.

Two cameras are considered similar if they capture similar areas of the scene. One way to quantify this similarity is comparing their frustums, which in turn can be performed by determining the distance between the camera centers and the angle formed by the look-at vectors. The smaller the value of these two parameters, the higher the similarity between the two cameras.

The sets of similar cameras are computed only once, since they only depend on the camera calibration and the two similarity parameters (maximum baseline and maximum angle).

The cross-check threshold is implemented based on the vote maps used for individual voting. The point is rejected if there is at least one similar camera in which the point's voting is below the threshold. Assuming that camera similarity has been determined for $\mathbf{C}^i$, and that a candidate $\mathbf{v}^i_{q,\bar{M}}$ has already been selected using Algorithm 3.1 and Algorithm 3.2, cross-checking is applied according to Algorithm 3.3.

The need for this concept of similar cameras arose from the fact that cameras may be placed in such a way that the scene is covered from multiple and even opposing directions (e.g. all around one or more objects). This implies that a reconstructed point may not be visible in some cameras. Since there are no assumptions on the object(s) being reconstructed, or the camera setup, there is no rule to identify this visibility relationship between cameras and reconstructed points. From all the cameras presented in the setup, the ones that are most likely to "see" the reconstructed point are its reference camera and the similar cameras. Thus, using all cameras to compute the estimates and the similar cameras to validate guarantees some more robustness. The tightness of the similarity measure is determined by the scene and the camera setup.

- Let $\mathbf{v}^i_{q,\bar{M}}$ be the candidate selected in Algorithm 3.2
- For each image I^s such that similar($\mathbf{C}^s$,$\mathbf{C}^i$)
  - Project $\mathbf{v}^i_{q,\bar{M}}$ in I^s as $\mathbf{p}^{i,s}_{q,\bar{M}}$
  - If $\upsilon^{i,s}_{q,\bar{M}}$ < crossCheckThreshold
    - Reject $\mathbf{v}^i_{q,\bar{M}}$

Algorithm 3.3: Cross-checking

Integrated cross-checking

In Algorithm 3.3, only the candidate $\mathbf{v}^i_{q,\bar{M}}$ that was selected from Algorithm 3.2 is evaluated. In the case of a point with multiple high maxima, if that candidate does not pass the cross-check, none of the other high maxima are checked, and the point is rejected.

Instead of using cross-checking simply as a rejection mechanism, it is possible to

integrate it with the maximum search as a selection mechanism: if the candidate closer to the camera that is above the votePercentThreshold does not pass the cross-checking, the second closest candidate is checked. The process repeats until a candidate passes the cross-checking, or no more candidates are left (Algorithm 3.4).

Hence, with integrated cross-checking a larger number of valid points is obtained, at the cost of cross-checking more candidates per point, when compared with the regular cross-checking presented in Algorithm 3.3.

- Let $M_q^i = \max_x \lambda_{q,x}^i$, for a given $\mathbf{p}_q^i$
- If $M_q^i < minVotes$
  - Reject $\mathbf{p}_q^i$ as invalid
- else
  - Let $S_{max} = \left\{ \mathbf{v}_{q,x}^i \;\middle|\; \begin{array}{l} \mathbf{v}_{q,x}^i \text{ is a local maximum} \\ \lambda_{q,x}^i \geq M_q^i \times votePercentThreshold \\ v_{q,\bar{M}}^{i,s} \geq crossCheckThreshold, \text{ for all } \mathbf{C}^s \text{ similar to } \mathbf{C}^i \end{array} \right\}$
  - Select the candidate $\mathbf{v}_{q,\bar{M}}^i \in S_{max}$ (if any) that is closest to camera $\mathbf{C}^i$

Algorithm 3.4: Integrated cross-checking

Early experiments have also shown that, in practice, cross-checking (either regular or integrated) eliminates the need for a minimum voting threshold, as that minimum is already guaranteed by the voting from similar cameras.

3.4.2 Merging of reconstructed points

The method described so far can be used to estimate a set of r-points chosen from a reference image, using data from all the other images. By applying the method using different reference images, multiple sets of points are obtained.

This section presents an approach to merge the r-points' sets resulting from reconstructing using multiple reference images. The advantages of merging are twofold: to provide higher integration of information between images, and reduce the resulting data sets; and to further increase robustness.

Consider a scene reconstructed from a reference camera $\mathbf{C}^i$. Assume a second reconstruction using a camera $\mathbf{C}^j$, similar to $\mathbf{C}^i$. A significant number of the points that are correctly reconstructed from camera $\mathbf{C}^i$ should also be correctly reconstructed from $\mathbf{C}^j$. Moreover, the two reconstructions of a given world point should overlap (in the ideal case) or be very close in world space (allowing a tolerance for e.g. small errors in camera calibration or noise). By merging such pairs of reconstructed points originated from different reference cameras and representing the same world point, the resulting data set is simplified.

The existence of a pair of points sufficiently close to be merged is also an additional evidence that there should be a world point in that area. Conversely, a point reconstructed from e.g. $\mathbf{C}^i$ that does not merge with any point reconstructed from $\mathbf{C}^j$

lacks supporting evidence that it has been correctly reconstructed. This information may be used to reject that point, and serve as an additional layer of robustness against incorrect estimates.

Intuitively, merging should take place between points that are very close in 3D space. This definition of closeness is context-dependent, and it should be defined in relation to the scene size.

However, in some cases what is considered as a small 3D distance at scene scale, may represent a large 2D distance in a close-up image of a part of the scene. For this reason, a merging procedure should also take into account the 2D distance of the projections of points that are close in 3D space. This ensures that points are merged only if they are sufficiently close on a global scale (3D), as well as on a local scale (2D). Merging should therefore be performed at two levels: 3D and 2D. For efficiency reasons, it is proposed to perform merging in two steps: first in 3D, as it is less expensive (does not require point projection) and it will reduce the number of points that survive to the more expensive 2D stage.

In the first stage, pairs of r-points that are close in 3D space and come from *distinct* reference images are searched. This closeness is limited by a threshold: merge3DThreshold. This threshold should be set relative to the scene's dimensions.

Pairs of 3D-close r-points ($\mathbf{r}_q^i$, $\mathbf{r}_t^j$), associated respectively to the corresponding c-points $\mathbf{p}_q^i$, $\mathbf{p}_t^j$ in cameras $\mathbf{C}^i$ and $\mathbf{C}^j$, are compared in 2D space.

In order to merge between $\mathbf{r}_q^i$ and $\mathbf{r}_t^j$, their projections must also be close in 2D space, i.e., the projection of $\mathbf{r}_q^i$ in the image plane from $\mathbf{C}^j$ must match $\mathbf{p}_t^j$, and *vice versa*. A 2D merge tolerance – merge2DThreshold – will be defined as the distance in pixels at which the two projected points may be in the images.

Merging is incremental, i.e. if the pairs ($\mathbf{r}_q^i$,$\mathbf{r}_t^j$) and ($\mathbf{r}_t^j$,$\mathbf{r}_r^k$) have been merged, then ($\mathbf{r}_q^i$,$\mathbf{r}_t^j$,$\mathbf{r}_r^k$) are merged as well. For each merged point, the set of r-points and its associated information is kept.

The resulting 3D merged point (*m-point*) corresponding to a set of points $\left\{\mathbf{r}_q^i, \mathbf{r}_t^j, ..., \mathbf{r}_r^k\right\}$ corresponds to the centroid of that point set.

The larger the number of independently estimated r-points that are associated with a single m-point, the larger the supporting evidence that the m-point is indeed correctly estimated. Inversely, if a given r-point is not merged into any m-point, it is discarded from the final reconstruction.

Hence merging provides yet another layer of validation of the estimates for each image, thereby adding robustness to the overall method, as will be shown in Chapter 5. However, it may also lead to sparseness, in two different ways: excessive rejection of non-merged points due to a too strict (low) value of the thresholds, and excessive merging caused by a too liberal (high) value of the thresholds. A diagram of these effects is depicted in Figure 3.14(a,d) and (c,f), respectively.

If the 3D or 2D thresholds are set too low (compared to scene scale or image resolution, respectively), only points that are very close are actually merged. This may lead to higher confidence in the merged points, but also result in many isolated

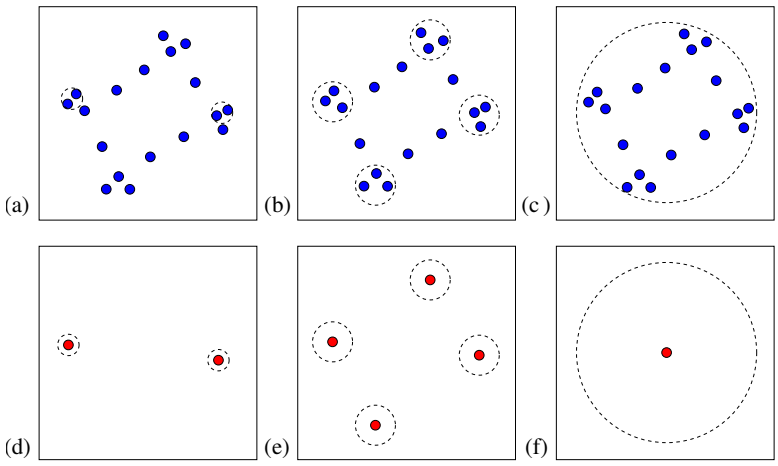


Figure 3.14: *Sparseness and the choice of merging thresholds.*
Top: the same group of points with different merge thresholds – tight (a), medium (b) and large (c)
Bottom: the resulting merged points - sparseness by excessive rejection (d), acceptable distribution (e) and sparseness by excessive merging (f).

points that are not merged and therefore are rejected.

The higher the value of the thresholds, the higher the number of points that can be merged into one. If the source points are scattered, the merged point may be placed in a position that is far from any of the original points. Consequently, the errors are higher and gaps appear in unexpected areas.

Therefore, there is a compromise in the choice of these thresholds, to balance merging and rejection and avoid the aforementioned problems of excessive merging or rejection.

3.5 Reconstruction of feature points

Reconstructing all contour points originating from multiple reference images represents a large number of point reconstructions. In many cases, it is possible to simplify the set of points to be reconstructed, while still being able to represent a good coverage of the scene. This significantly reduces computational load. Another use for a simplified set of points is to provide a "fast preview" of the reconstruction that, albeit simpler, it is representative of the full reconstruction and provides an insight of the outcome of the full reconstruction.

To achieve this, it is necessary to determine a subset of points that is *representative* of the contour lines, i.e. it is an approximation of the full contour point set. Ideally, one should be able to easily control how detailed the approximation should be, such that it could be adjusted to the requirements of applications: less demanding applications could use a coarse *level of detail*, while more critical applications would use a higher level of detail.

If one aims for a small reduction in the number of points (high level of detail, e.g.

half of the points), sampling the full point set can be an option, e.g. choosing one in every two points.

However, for higher point reductions (e.g. in the order of 1/5 or 1/10), using random or linear sampling can lead to the loss of important details of the scene, such as corners of objects. In these cases, a more guided choice of the points would be better suited.

In this section, a method to select contour points based on the topology and shape of contour lines is proposed. It favors topological and shape-defining feature points like junctions, extremities and corners of contour lines. The level of detail of the approximation can be easily controlled by a single parameter, and in some cases it can provide a significant reduction in the final number of reconstructed points, while maintaining the overall structure of the scene, as will be shown in the results (Chapter 5). Given its focus on line shape, using this method is also useful when extending point reconstruction to line reconstruction (Chapter 6).

The section is divided into two parts: the first part contains a classification of the types of feature points to be considered; the second part describes the extraction of feature points from contour maps, and the mechanism that controls the choice of the level of detail.

It is assumed that the contour lines are thin, as e.g. the output of a Canny edge detector, or the result of applying a morphological thinning operation [Sonk 99] to a contour map.

3.5.1 Classification of feature points

A thin-contour map can be described as a set of non-intersecting *contour paths*, where each path is a sequence of *contour points* which are adjacent considering an 8-connected neighborhood.

Each path either has a start and an end point, or is a closed loop. Start and end points can be any of the following (Figure 3.15):

extremities points with only one neighbor i.e., points where a single path ends.

junctions points with three or more neighbors i.e., points where three or more lines meet.

Due to their topological properties, extremities and junctions will be generally referred as *topological feature points*, or *topological features*.

Topological features represent important points in a contour map, and are usually a hint of important features of the scene. For example, a junction may be caused by the occlusion of a surface by another. Furthermore, in the case of thin-contour maps, topological features are relatively easy to extract using a local neighborhood, since their definition is only dependent on the number of neighbors.

Another type of important points in a contour map are *corners*. A corner is a point where two lines of different orientation meet. A path can be decomposed as a set of line segments of different orientation, meeting at corners. In other words, a path can

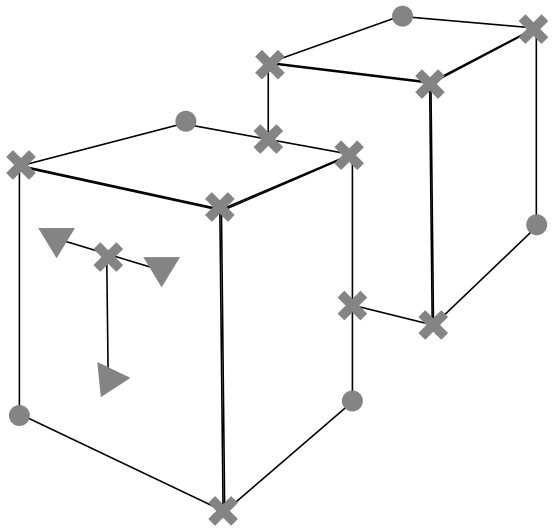


Figure 3.15: *Different types of contour points*.
Junctions (crosses), Extremities (triangles) and corners (dots)

be approximated by a set of corner points and the straight-line connections between them.

Corners and other features are recognized to be important and commonly used not only in shape representation, but also in other image processing applications [Schm 00], such as object tracking or camera calibration. In terms of reconstruction, a corner in a contour is a hint for the existence of a corner in the scene.

However, there is no exact definition of a corner: how significant is a curvature or the angle between two lines to consider the existence of a corner? Corner definitions are usually scale-dependent: what is considered a corner at a high resolution may be negligible in a lower resolution.

When decomposing paths as sets of lines and corners, this resolution influences the number of corners and the degree of approximation of the original path. Controlling the resolution at which corners are detected is a way to control the level of detail.

Corner points are therefore a viable complement to the reconstruction of topological features, both in terms of relevance of the type of points, and in terms of control of the level of detail. While all topological features should always be considered, the corners to use may vary depending on the desired level of detail.

Finally, for the sake of completeness, *isolated points* that may exist in a contour map will also be considered as feature points.

From this point forward, when referring to the set of topological features, corners and isolated points, the more generic term *feature points* will be used.

The next section details the process of extracting topological features from contour maps, and the extraction of corners from paths with control of level of detail.

3.5.2 Feature point extraction

In this section, a method for extracting topological features, paths and corners from contour maps is proposed. It is assumed that thin-contour maps have been extracted from images *a priori*, as it is the case with point reconstruction. The process is divided into the following steps, which are detailed later:

- Detect topological feature points.
- Extract the paths connecting topological feature points.
- Extract the corners from each path.

Detection of topological features

Topological feature points are identified by the number of lines that meet at a point. In thin contour maps, these lines can be identified by counting the number of contour points in a 3 x 3 neighborhood of the point.

Contour points with three or more neighbors are *junctions*, contour points with a single neighbor are *extremities*, and contour points without neighbors are *isolated points*.

Path extraction

A path is a contiguous collection of non-topological feature points, that are either in a closed loop or limited by two topological features (in some cases, these can be geometrically coincident, as e.g. in the symbol ρ representing the Greek letter *rho*).

Having detected topological features, path extraction amounts to following each individual sequence of contiguous contour points – paths – that leave from topological features, until the next topological feature is found. Each contour point followed is assigned to the corresponding path, and marked as visited, to avoid following the same path twice (starting from different topological features). The process is described in more detail in Algorithm 3.5, and depicted in Figure 3.16.

- For each topological feature point f_i
  - For each contour point p in the neighborhood of f_i that has not been visited yet
    - Start a new path L
    - do
      - Mark p as visited and belonging to L
      - Look in the neighborhood of p for a contour point q that has not been visited yet, to be followed
      - $p = q$
    - while p is not a topological feature point f_j

Algorithm 3.5: Following contour paths between pairs of topological feature points

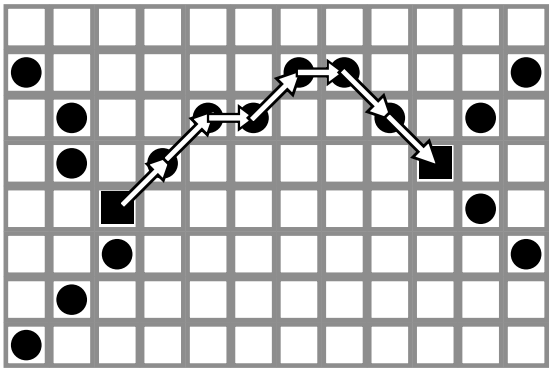


Figure 3.16: *Contour following*.
Starting from a topological feature point (represented as squares), a contour is followed by iterating over non-visited neighbors

After applying Algorithm 3.5, all paths connected to topological feature points will have been followed. Paths that do not contain any topological feature points – closed loops such as an isolated contour – must also be followed. Closed loops are easily identified as the collections of contiguous points that have not been visited. Following a closed loop is similar to following a single path. The difference lies in the fact that there are no clear starting and ending points for following. In this case, one or two arbitrary points of the loop may be chosen to play that role.

Corner extraction

As mentioned in Section 3.5.1, the type of corners which are relevant in the context of this work are those that can be used to approximate the shape of a contour line by straight line segments, up to a given resolution.

This definition is in line with the purpose of *line simplification* algorithms [Doug 73] used e.g. in cartography for road mapping: having a line defined by a sequence of points connected by line segments, try to obtain a smaller sequence of points[2] that, when connected by lines, approximate the shape of the original line[3].

Any path that connects two topological feature points can be seen as a sequence of small lines, each connecting a pair of adjacent contour points (Figure 3.17).

By applying a line simplification algorithm to this (typically large) sequence of small segments, these will be simplified into a smaller set of larger segments. The *corner points* are the end points of each of the segments of the final simplified set (excluding the topological points corresponding to the path's extremities).

Different line simplification algorithms were tested to assess which would perform

[2]In some line simplification algorithms, the points of the simplified set do not necessarily belong to the initial set. In this work, the simplified set is always a subset of the initial set, as it would not be appropriate to reconstruct points that do not belong to the original contours.

[3]There are also some similarities with the processes known as bitmap tracing, used to convert bitmap images into vectorial drawings.

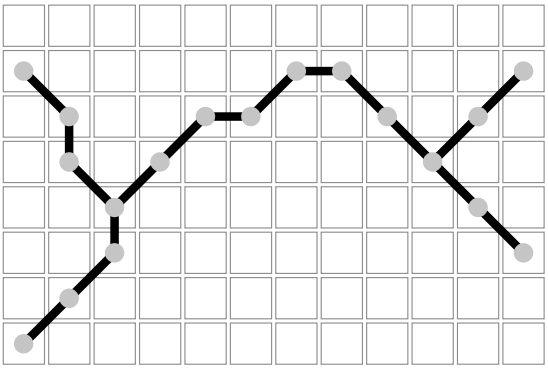


Figure 3.17: *Contour lines as sequences of small line segments.*
The sequences of contour points of Figure 3.16 can be interpreted as a set of line segments (black) connected at points (grey) located at pixel positions

best (i.e. produce better approximations with fewer points) with the type of small line segments originated from contour lines.

The best results were obtained with the Douglas-Peucker algorithm [Doug 73] (also known as Ramer's or sandwich algorithm). This algorithm is described in Algorithm 3.6 and depicted in Figure 3.18, and uses a threshold ϵ_l to control the level of approximation (low ϵ_l meaning detailed approximation, high meaning coarse).

- Connect the start point f_i to the end point f_j with a straight line l
- Determine the point c from the original set that is farther away from l
- If $distance(c, l) > \epsilon_l$
  - c is labeled as a corner point
  - l is divided in two parts $f_i - c$ and $c - f_j$
  - The algorithm is applied recursively to each of the two parts
- Else
  - Stop the algorithm for l

Algorithm 3.6: Line Simplification

After applying the line simplification algorithm to a contour line, a number of corner points will have been extracted.

The density of feature points or *level of detail* is controlled by the value of ϵ_l used in line simplification: lower threshold values imply a better approximation of the 2D curves (higher level of detail), and hence a higher number of corner points.

This is reflected in terms of point reconstruction, since a higher number of points

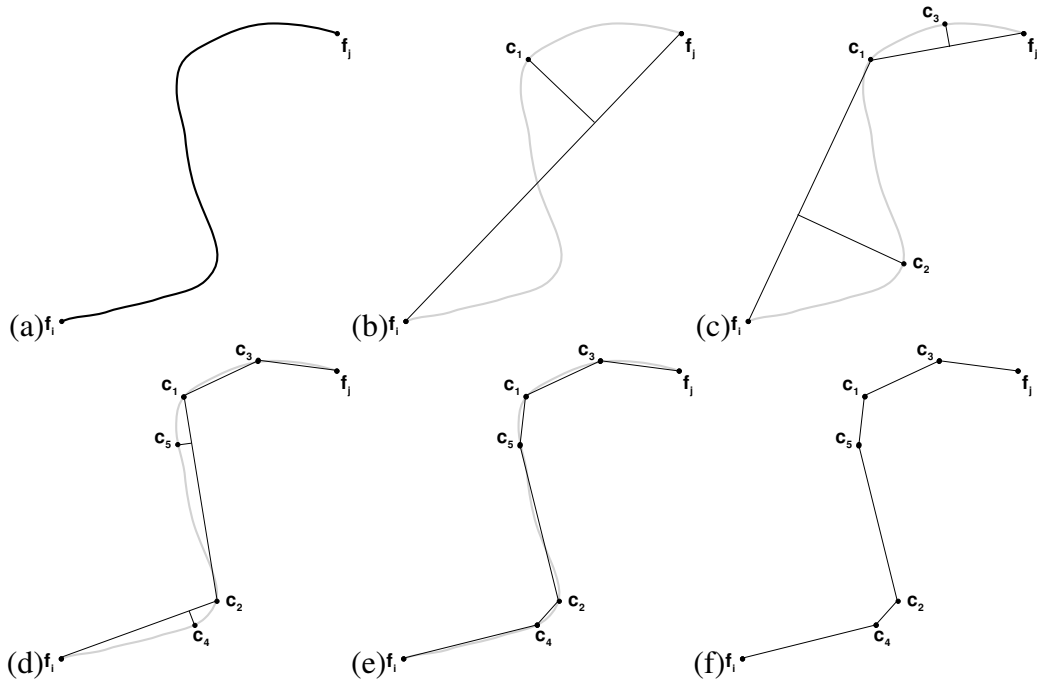


Figure 3.18: *Line simplification*.

(a) Original line with start (f_i) and end (f_j) points marked

(b) The distance between each point of the original line and the straight line connecting f_i and f_j is computed; the farthest point c_1 is chosen for the next iteration

(c) The algorithm branches and recurses for each of the two line segments defined by the two pairs of points (f_i, c_1) and (c_1, f_j)

(d) Another iteration; Recursion stops for a given branch when the distance to the farthest point is below a predefined threshold (e.g. (c_1, c_3) and (c_3, f_j) are not divided anymore)

(e) The process is finished when recursion ends on all branches

(f) The resulting approximation of the original line

represents both a higher number of point reconstructions, and a better approximation of 3D lines and curves. The threshold ϵ_l can therefore be used as a control of the trade-off between level of detail and computational and data complexity.

Experiments have shown that, to obtain a pixel-close approximation of the contour paths without excessive number of points, this threshold can be fixed at a value of 1.2 pixels (other values may be used if different levels of detail are desired). The effects of applying this simplification presented in the results (Chapter 5) show the significant reduction in the number of points (up to 13 to 1), and the comparable quality between full and simplified reconstructions (both visually and qualitatively).

3.6 Comparison with other methods

The reconstruction method presented in this chapter brings together different concepts from different methods, and combines them in an innovative way. The method also

introduces some novel ideas. The most significant differences and advantages over related methods are outlined in this section.

Pollefeys et al's method [Poll 00, Corn 01, Poll 99] uses a correlation-based stereo applied to pairs of rectified images, taken from closely-placed cameras (small baseline), to obtain sets of independent dense depth maps. The depth maps are then fused, interpolated and regularized to form a surface, that is textured using the original images. This technique provides good visual results for scenes that have few or no discontinuities, but not in scenes with multiple occlusions, and with cameras that have significant camera separation and angle variations. From their results, it was noticeable that even after regularization, surfaces contain a significant number of bumps (an example is presented in Figure 2.18, in Section 2.2). This indicates that regularization plays a strong part in the final aspect of their reconstructions, and therefore it is also likely to smooth out important depth discontinuities. Zisserman et al's method [Fitz 98, Ziss 99] shares the same limitations in terms of baseline as Pollefeys et al's method, as it uses image triplets and a correlation-based method for establishing correspondences. In contrast, the method presented here is able to deal with wide baseline and to use information from multiple cameras simultaneously. It is able to deal with multiple occlusions, and to provide good results without the need to resort to regularization, thus respecting strong discontinuities. This thesis' method, being based on contours, should also be less sensitive to lighting variations than correlation-based methods.

Seitz and Dyer's basis approach to the problem [Seit 95] is similar to the one used here, by selecting features from a reference image and gathering evidence from features in the other images, on the neighborhood of the epipolar lines. However, they only consider 3D features that are identified in all the images, which is a significant limitation in the case of problems in feature extraction or occlusion. The distance-based approach used by this thesis' method allows not only to account for missing or occluded features, but also to directly weigh the possible offsets caused by camera calibration errors, image noise or problems in edge detection. Furthermore, the dependency of Seitz and Dyer's method on rectification to simplify the epipolar search also poses some limitations in the camera placement, that do not apply to this thesis' method.

Tsai's [Tsai 83] and Okutomi/Kanade's [Okut 93] multiple baseline methods have acknowledged the importance of accumulating evidence from multiple images, which is a concept used in this thesis' method. Tsai's method was very preliminary and thus limited in camera placement and applicability. Kanade's work in generalizing its applicability to more flexible camera arrangements was important [Kana 97, Nara 98], but still limited the number of cameras used due to problems in matching. These limitations came essentially from the use of correlation based measures, and are reduced in this thesis' method. By using entities such as contours, which are less sensitive to wide baseline, a larger number of images can be used simultaneously, and with wider coverage of the scene, thus allowing the simultaneous integration of more information.

Mellor's approach [Mell 96] was also built on the idea of merging information from multiple images in a same view ray. Compared to this thesis' method, their use of a color-based match measure computed on a per-pixel basis leads to high sensitivity to

image noise, lighting variations and camera calibration errors.

Szeliski and Weiss' work [Szel 98] is interesting in its particular focus on occluding contours, but its dependence on small baseline to be able to track contours limits its applicability, which does not happen in the method presented here.

Jung et al's method [Jung 02] selects features (edgels in their case) from a reference image, and search for correspondences in the corresponding epipolar lines in other images. However, only features that *intersect* the epipolar lines are considered. This strongly limits the method in the case of errors and offsets caused by problems in camera calibration or feature extraction.

Bauer et al's work [Baue 04, Klau 03], based on Jung's, adds some tolerance to offsets, but it serves only as a rejection mechanism, not as a weighing factor. Furthermore, they are limited in camera placement due to constraints imposed in the gradient direction, and to the use of an image-based descriptor to remove outliers. In this thesis' method, by incorporating the distance in the metric, one achieves more robustness to the aforementioned error sources, without the need of very restrictive measures in terms of camera placement. The significantly higher freedom in camera placement is another advantage of this thesis' method, compared to Jung's, Bauer's, and also to Collins's work [Coll 96].

Compared to visual hull methods (e.g. [Szel 99, Matu 01, Matu 04]), the method presented in this chapter has the main advantages of not requiring background separation and being less affected by camera calibration errors – in visual hulls an incorrect camera displacement may cause significant changes in the hull.

Compared to voxel-based methods (e.g. [Seit 97, Kutu 00, Culb 00]), this thesis' method provides less sensitivity to lighting variations and image noise, as well as to camera calibration errors. It also does not require as elaborate occlusion reasoning as voxel-based methods. A related advantage is the fact that points can be estimated independently of each other, thus providing a degree of parallelization not easily achievable in voxel-based methods.

3.7 Summary and conclusions

In Chapter 2, different 3D reconstruction methods were reviewed, each representing a compromise between a series of factors, such as constraints of the type of scenes and objects, limitations in the placement of cameras, robustness to camera calibration errors or noisy images, and performance.

In this chapter, a point reconstruction method was proposed that tries to simultaneously:

- relax constraints that are common to many reconstruction methods, thus reducing compromises and requirements;
- be robust to error sources commonly found in real scenes, making it useful in real scenarios;

· be a robust complement to other reconstruction methods, by providing robust point estimates that could serve as a strong bootstrapping for higher dimensional reconstructions such as lines and surfaces.

The proposed method is based on the reconstruction of contour points. The choice of contour points was based on the expected benefit of reducing limitations in camera baseline, due to the low sensitivity of points to perspective distortions.

Using contour points also represents a problem of high ambiguity in correspondence. One challenge was to overcome this ambiguity. To achieve this, a voting strategy using multiple cameras was proposed. For each contour point from a reference camera, a set of candidates is selected from its view ray, as potential reconstructed points. Each candidate is voted by the other images, and a winner should be selected.

This process raised several issues: how to choose the candidates, what type of voting (metric) should an image contribute with, how to compute votes efficiently, how to accumulate votes from different images and how to choose a winner.

The candidates were chosen by sampling the view ray in points in world space. Sampling was chosen over methods based on maximization, due to the observed irregularity of the voting function. By defining candidates as points in world space, the problem of accumulating votes from different images was also simplified.

The votes attributed in an image to a candidate were related to the closeness of the candidate's projection to a contour on that image. This meant that correct candidates that would always project on a contour should have very high accumulated voting, and incorrect candidates that would project far from contours in multiple images would have a very low accumulated voting. Candidates that would not project exactly on a contour but close to one would still receive some votes. This implicitly provided a degree of tolerance to offsets in contours caused by e.g. noisy images, faulty contour extraction, or errors in camera calibration.

The computation of such a voting function was efficiently implemented as the complement of a distance function, for which efficient distance transform algorithms exist. The voting function based on contour closeness has the added advantage of being independent of the point being reconstructed. This means that the voting function of each image could be precomputed only once and stored as a vote map that can be used for the reconstruction of all points. Having the vote maps precomputed, the task of vote collection for a candidate is reduced to reading a value (voting) from each vote map and accumulate all the votings.

The problem of selecting the winning candidate is then addressed. The choice of the highest voted is not always the most adequate, namely in situations where multiple high voted candidates exist. These situations were analyzed to choose a course of action that reduces the probability of choosing an incorrect candidate over the correct one.

Still, it is not easy to detect cases of cross voting from direct analysis of the accumulated votes. To improve the results in those cases, two robustness extensions were proposed: cross-checking and point merging.

Cross-checking gives more emphasis to cameras that are similar to the reference

camera, when deciding if a candidate is correct or not. The result is an improved detection of cross-matching, that in some cases allows to recover the correct candidate.

Merging adds an additional layer of robustness to outliers, by joining points reconstructed from different images that are assumed to correspond to the same world point. Points that are not merged are rejected, under the assumption that a point correctly reconstructed from one reference image should also be reconstructed from other reference images. The expected result is a final point set with less scatter and with very few outliers.

The main algorithm, together with the robustness extensions, provides a robust method for reconstructing contour points, as will be shown in Chapter 5.

A final extension to the method was proposed, in order to further reduce the computational load. Instead of reconstructing all contour points, the reconstruction of a representative subset was explored, namely the set of feature points extracted from contours. It is expected that such a set can significantly reduce the number of points to be reconstructed, while still providing a representative subset of points that presents a distribution and reconstruction quality similar to the full point set. This will allow, among other things, to be able to provide with this smaller point set a "fast preview" of what a more complete reconstruction would look like. The results of tests with the reconstruction of feature points are also shown in Chapter 5.

Chapter 4

Accelerated reconstruction using commodity graphics hardware

4.1 Introduction

Performing 3D reconstruction based on images requires a large amount of processing, much of it related to geometric computations. In particular, the reconstruction algorithm presented in Chapter 3 requires a significant amount of 3D candidates for each contour point to be projected in multiple images. For a scene with tens of images, this may reach hundreds of thousands of point projections, each involving multiple operations, being matrix computations the most expensive ones.

In a traditional programming paradigm, this represents a significant amount of computation, and long computation times using off-the-shelf CPUs. Most of the computation tasks involved correspond to projections and other geometric operations. Such tasks are highly optimized in current *graphic cards (GC)* (also commonly referred to in the literature as *consumer-grade graphics hardware* or *commodity graphics hardware*).

Graphic cards are devoted to the processing of graphics primitives for visualization and rendering, as a form of relieving the main processor from most graphic tasks. The computational power of GCs and their availability has grown tremendously in recent years, and continues to do so, powered in great part by the gaming industry.

Taking advantage from a pipelined and parallel architecture, powerful *graphics processing unit (GPU)* with single instruction, multiple data (SIMD) and in some cases multiple instruction, multiple data (MIMD) capabilities and fast and dedicated memory, procedures such as matrix computations or texturing are highly optimized and can be executed much more efficiently in GCs than in CPUs. The difference is especially more noticeable in the case of operations that are applied to large volumes of data, such as computing projections for thousands of points.

Given the fast growing capabilities of off-the-shelf (or commodity) graphics hardware, interesting solutions for image-based rendering, new view synthesis and 3D reconstruction based on this type of hardware acceleration are coming to life.

The wide availability and programmability of graphics hardware, as well as its ori-

entation toward geometry processing, make it a tool for 3D reconstruction that should be highly considered. Focusing on the computational model of such hardware leads to new types of reconstruction techniques, which started to be explored by several researchers.

The following are some of the efforts done in the direction of implementing 3D reconstruction techniques using graphics hardware.

Yang, Welch and Bishop [Yang 02] propose a solution that synthesizes new views of a scene captured by five calibrated cameras in real-time. The possible location of the virtual cameras is limited to a close range of the original cameras. To accomplish real-time, they use the graphics board's texture and accumulation buffers in a pixel shader to implement a space-sweep approach. The technique is unable to handle occlusions, and is restricted to small baselines and a fixed number of cameras. Some changes are proposed to adapt the algorithm such that depth can be computed instead of color, but these were not explored by the authors.

Zach et al [Zach 03] optimize reconstruction of stereo pairs using commodity graphics hardware to accelerate the reconstruction. Main steps of the algorithm, such as image warping and differencing, sum of absolute differences and the search for optimal depth value are performed on the GPU. They also propose [Zach 04] an hierarchical disparity algorithm entirely executed on the GPU.

Visual hull rendering is accelerated in hardware by Li et al [Li 03b], while voxel carving is in the focus of the work by Sainz et al [Sain 02]. An hardware-accelerated version of visual hulls using current off-the-shelf graphics hardware can be seen in [Li 03a].

The GC-based implementations have to be oriented to the specific underlying reconstruction methods, making the referred implementations unsuitable to be adapted to the reconstruction method presented in this thesis. This chapter proposes an implementation of the reconstruction method presented in Chapter 3 that makes use of commodity GCs to accelerate a major part of the process. This work has also been published as [Rodr 04a].

Common reconstruction tasks such as view ray projection or vote accumulation will be translated to graphics primitives and operations available on most off-the-shelf graphics boards and accessed through standard *Graphics APIs (GAPI)* such as OpenGL [Shre 00]. More specifically, the following techniques used in the field of computer graphics will be applied to the reconstruction method :

- *Projective texturing* is used in the process of collecting votes from vote maps;
- *Additive blending* is used to perform vote accumulation using the graphics card

The mapping between the operations of the main point reconstruction algorithm (Algorithm 3.1) and the aforementioned techniques is represented in Algorithm 4.1.

Section 4.2 introduces the concepts of texturing and projective texturing, and explains how camera information and vote maps can be sent to the GC and used to perform vote collection from one vote map on the GC.

- Let V_q^i be the view ray of $\mathbf{p}_q^i$ associated to $\mathbf{C}^i$
- Choose a set of candidate world points $\mathbf{v}_{q,1}^i$, $\mathbf{v}_{q,2}^i$, ... from V_q^i
- For each image I^j $(j \neq i)$
  - For each candidate world point $\mathbf{v}_{q,x}^i$
    - Project $\mathbf{v}_{q,x}^i$ in I^j as $\mathbf{p}_{q,x}^{i,j}$
    - Let $v_{q,x}^{i,j}$ be the voting of $\mathbf{p}_{q,x}^{i,j}$ in I^j
    - Add $v_{q,x}^{i,j}$ to the total voting $\lambda_{q,x}^i$ of $\mathbf{v}_{q,x}^i$
- Select the r-point $\mathbf{r}_q^i$ corresponding to the reconstruction of $\mathbf{p}_q^i$ according to Algorithm 3.4

Algorithm 4.1: Reconstruction using view ray candidates (conceptual mapping to GPU).
Operations mapped to projective texturing are shaded as light gray, and the operation mapped to additive blending as dark gray

Section 4.3 extends vote collection from one vote map to vote accumulation from multiple maps, introducing the concept of additive blending. This allows that votes collected from different images for a given candidate get accumulated together in the same location in the GC's memory.

Vote collection and accumulation are performed in the GC's memory, and transferred to the main memory for the final processing. To efficiently structure and access the vote information coming from the GC and associate it to the candidates once it is in the main memory, it is necessary to properly set up the GC. This set up is detailed in Section 4.4.

The chapter is concluded with final remarks on this implementation (Section 4.5).

4.2 Vote collection using projective texturing

Texturing is a common graphical task supported in GCs[Shre 00]. It is a method to associate information (e.g. color), coded as a *texture*, to graphical primitives.

Figure 4.1 shows a simple example of texturing. The goal is to "paint" a triangle with part of an image that is stored as a texture. To achieve this, it is necessary to specify which part of the image should be mapped to each vertex of the triangle.

In the previous example, the texture coordinates of the vertices of the primitive were explicitly assigned, and other coordinates interpolated from it. This is the most common use of texturing, but there are different ways to assign texture coordinates.

Projective texturing [Sega 92, Shre 00] is a particular case of texturing in which the texture coordinates of a point are computed based on the point's position in the world,

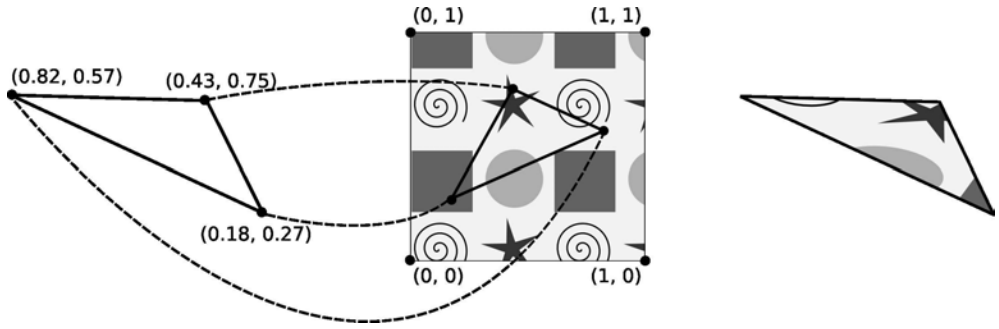


Figure 4.1: *Texturing*.
Texture coordinates are associated to the vertices of a primitive (left); The coordinates are used to locate information in the texture (center); When the primitive is actually rendered, the texture information is mapped onto it by interpolation based on the texture coordinates and values (right)

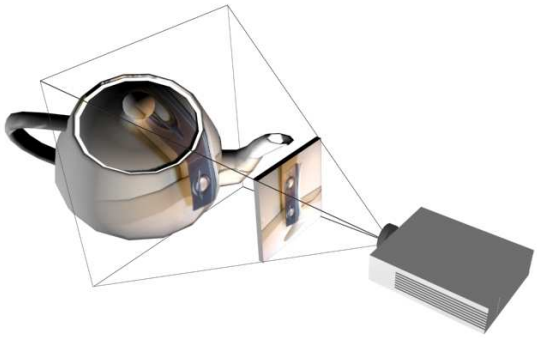


Figure 4.2: *Projective texturing*.
An image is cast over geometry as if it was a slide on a projector

and the parameters of a virtual *projector* which casts the texture over the primitives to be rendered. This allows, for example, for an image to be projected onto an object, providing an effect similar to a virtual slide projector (Figure 4.2).

This procedure can be used in the context of reconstruction to *project an image in a segment of a view ray*. This allows the reversal of the common approach of projecting a view ray in an image (directly or implicitly), to obtain the corresponding epipolar line. By performing this computation on the GC, one takes advantage of the GC's dedicated parallel architecture.

Let $\mathbf{J}^j$ be a virtual projector set with the same intrinsic and extrinsic parameters of a camera $\mathbf{C}^j$. Let V_q^i be the view ray associated to a point $\mathbf{p}_q^i$ in another camera $\mathbf{C}^i$. When an image taken from $\mathbf{C}^j$ is projected as a texture in V_q^i using $\mathbf{J}^j$, the part of I^j that actually covers V_q^i corresponds to the epipolar line $E_q^{i,j}$ of point $\mathbf{p}_q^i$ in $\mathbf{C}^j$.

In the context of the presented reconstruction method, the image to project as a texture is actually the vote map Λ^j. In this case, what is assigned to each rendered point of the view ray's segment are the corresponding votes from I^j, instead of the

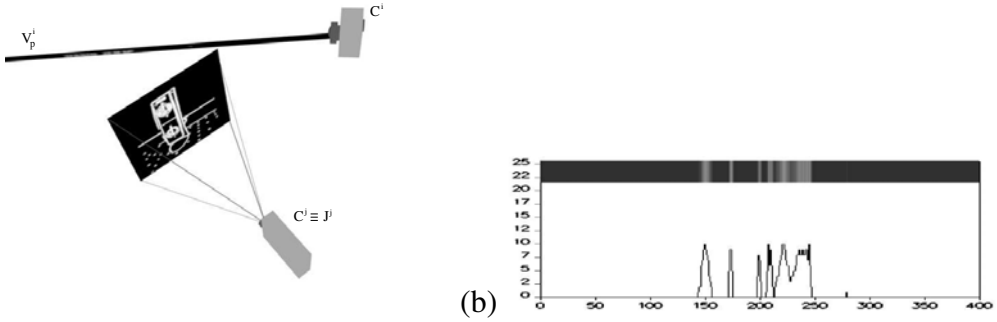


Figure 4.3: *Projective texturing for reconstruction.*

(a) A vote map is projected as a texture over a view ray (the ray is represented as a thick line to show the votes projected)

(b) A close-up of the ray with the projected votes coded as intensity (top) and the corresponding vote graph (bottom)

colors (Figure 4.3).

Projective texture coordinates (s, t, r, q) are computed by multiplying the point's coordinates in world space (x, y, z) by a transformation matrix $\mathbf{T}$ that corresponds to the projection of the point in the image plane of the projector (Equation 4.1). Note that the equations regarding transformations presented in this section use *homogeneous coordinates* (the reader is referred to [Aken 02] for more information on this topic).

$$\begin{bmatrix} s \\ t \\ r \\ q \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.1}$$

Computing projective texture coordinates is equivalent to computing a world point's projection in an image using the point's position and the corresponding camera's parameters. Hence, the projector's intrinsic and extrinsic parameters are the same as for the corresponding camera.

Therefore, in order to compute projective texture coordinates for a world point, the same matrices are used, namely the Model, View and Projection matrices. In the case of projective texturing an extra matrix, the normalisation matrix, is required to transform the range of projected points, [-1,1] into the range for texture coordinates i.e. [0,1].

Model Matrix (M) A given point may be specified relative to an object referential (the object space) which is in turn defined relative to a world referential. The model matrix is used to transform the point's coordinates from object space to world space.

View Matrix (V) The view matrix transforms the world-space 3D coordinates in projector-space 3D coordinates. The base referential for the resulting coordi-

nates is now the projector's position and orientation. This matrix contains the information of the projector's extrinsics.

Projection Matrix (P) The projection matrix converts the view-space coordinates into texture-space coordinates. This matrix encodes the projector's intrinsics.

Normalization Matrix (N) This matrix scales and offsets the image-space coordinates, such that they are mapped in the normalized range of [0..1] x [0..1].

In the context of reconstruction, for the purpose of view ray texturing, the indirection from object space to world space provided by $\mathbf{M}$ is not useful, since the points that define a view ray's segment are available directly in world coordinates. Therefore, the object space is coincident with the world space, making $\mathbf{M}$ constant and equal to the identity matrix.

The matrices $\mathbf{P}$ and $\mathbf{V}$ are dependent only on the projector's parameters, which are based on the corresponding camera's intrinsic and extrinsic parameters, and are therefore constant for each camera.

The product of the three matrices $\mathbf{P}$, $\mathbf{M}$ and $\mathbf{V}$ can be obtained directly from the GAPI, by setting up the projector using the corresponding camera's extrinsic and intrinsic parameters.

Finally, $\mathbf{N}$ is a scale and translation matrix, that also consists of factors which are constant for each camera. Since all the matrices involved are immutable for a given camera acting as a projector, the complete texture transformation matrix $\mathbf{T}$ may be computed in advance with the GAPI, according to Equation 4.2, and only once for each camera, and stored to save computation time (Algorithm 4.2).

$$\begin{aligned} \mathbf{T} &= \mathbf{NPVM} \\ &= \begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\mathbf{PVM}) \end{aligned} \tag{4.2}$$

- For each camera $\mathbf{C}^j$
  - Load the vote map Λ^j to the GC as a texture τ^j
  - Pre-compute and store $\mathbf{T}^j = \mathbf{N}^j\ \mathbf{P}^j\ \mathbf{V}^j\ \mathbf{M}^j$ based on the camera parameters for $\mathbf{C}^j$

Algorithm 4.2: Initialization of vote projection using projective texturing

The use of projective texturing for the computation of the votes of the candidates of a point $\mathbf{p}_q^i$ using a single vote map Λ^j associated to $\mathbf{C}^j$ is described in Algorithm 4.3.

After performing Algorithm 4.3, the GC's buffer will contain a set of points corresponding to l. Each of those points is a candidate. In the rendering process, for each point – candidate – the GC uses its 3D coordinates and the transformation matrix $\mathbf{T}^j$ to

- For each view ray V_q^i corresponding to a contour point $\mathbf{p}_q^i$
  - Load pre-computed matrix $\mathbf{T}^j$
  - Set texture τ^j as the active texture
  - Compute the line segment l of the view ray V_q^i
  - Render l on a buffer in the GC

Algorithm 4.3: Vote projection using projective texturing

get the corresponding value from the projected texture – the vote map Λ^j – and render it in the corresponding position in the buffer. In the end, each candidate, represented as a point of the buffer, will have associated its corresponding votes from Λ^j. The following section explains how to extend Algorithm 4.3 to accumulate votes from multiple images.

4.3 Vote accumulation for multiple cameras using additive blending

Section 4.2 describes a two-camera setting, where one camera is used as the reference from which a contour point and its corresponding view ray are selected, and the other camera is used to project a vote map over the view ray. To extend the method to a multiple-camera setting, textures (vote maps) projected from different projectors over the same view ray have to be accumulated. This accumulation can be performed in the same buffer, using *additive blending*.

Additive blending adds the texture values of newly rendered geometry to the values already existing in the buffer. In the context of reconstruction, a solution to incrementally accumulate votes (texture values) in the buffer is to render the same view ray multiple times using additive blending, each time using a different vote map as a texture projected from its corresponding camera (Figure 4.4). Newly projected votes are added to those already in the buffer.

There are limits to the maximum value that can be represented in textures and rendering buffers of GCs, and to the format in which they may be represented. Typically, each buffer has a predefined number of 8-bit channels that are treated independently i.e., *accumulation does not carry from one channel to the other*. For instance, an RGB (red-green-blue) buffer is composed of three 8-bit channels (one for each of the colors). This means that values stored on this buffer are limited in the range of (0, 0, 0) to (255, 255, 255).

Even though in more recent GCs there are other modes, such as 16-bit modes or single-channel 32 bit modes, their practical availability is still reduced. To benefit from them, it would be necessary to specifically program the GC. Furthermore, the use of such buffers would also imply a large number of floating-point operations for the accumulation, as opposed to the integer arithmetic used for accumulating per-channel values.

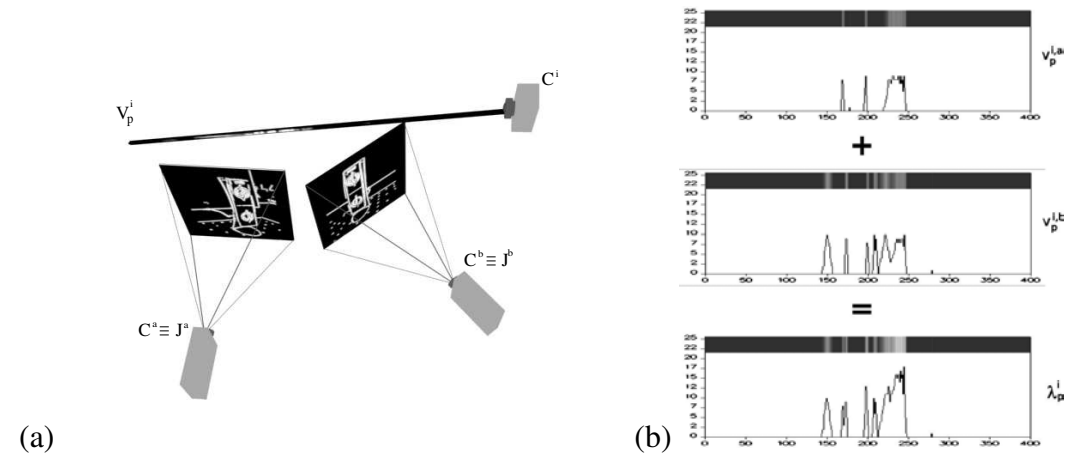


Figure 4.4: *Projective texturing with accumulation for reconstruction.*
(a) Two vote maps are projected as textures using accumulation over a view ray
(b) A close-up of the ray with the projected and accumulated votes

Therefore, a more generic solution is proposed to make use of the common RGB buffers using three 8-bit channels. Since each color channel is independent, accumulation can be done on a per-channel basis, using *color channel masking*.

4.3.1 Channel masking

Color channel masking sets which color components are actually rendered in a buffer. For instance, if a white line was rendered with the green and blue channels deactivated, only the red channel would be active, and thus the line would be rendered as red.

Channel masking can be used in vote accumulation, such that votes can be accumulated up to an effective range of $3 \times 256 = 768$ vote values. The way to achieve this is to accumulate on each channel only a safe number s of vote maps that are guaranteed not to overflow the channel's capacity. The fact that the vote values are clamped (see Section 3.3.5) provides a limit for this. For instance, if the maximum vote range in a vote map would be of 10 votes, votes from 25 vote maps could be safely accumulated on a single channel, as at most they would accumulate votes up to $25 \times 10 = 250$. Therefore, if a different set of 25 vote maps are used in each of the three channels, that would allow the use of up to 75 cameras.

Having a safe number s of cameras per channel, accumulation can then be performed by activating one channel at a time, and projecting only s vote maps in each channel (the cameras' distribution between the different channels is not relevant). When projecting a vote map, it has to be assured that the votes are properly represented in the active channel. To achieve this, vote maps are coded as grey values in a *luminance texture*. If all channels are active, a luminance value of e.g. 5 will be rendered as 5 in each of the three channels, as if it was an RGB triple of (5, 5, 5). When channel masking is activated, only the active channel will be changed.

Having votes distributed over different channels, a post-processing step is required

to add the values of each of the channels for each point in the buffer, to obtain the total vote.

These constraints pose some theoretical limits to the algorithm, namely in the number of cameras and vote range that can be used. In practice, these limits are sufficiently liberal, and will be difficultly reached. It would also be possible to do the accumulation in two or more passes, thus multiplying the maximum number of possible cameras.

Algorithm 4.4 describes the method to accumulate votes from multiple images using projective texturing and additive blending (assuming the initial setup described in Algorithm 4.2).

- For each view ray V_q^i corresponding to a contour point $\mathbf{p}_q^i$
  - Compute the line segment l of the view ray V_q^i
  - For each camera $\mathbf{C}^j$, $j \neq i$
    - Activate channel 0, 1 or 2: $activeChannel = j \bmod 3$
    - Load $\mathbf{T}^j$
    - Set texture τ^j as the active texture
    - Render l on the GC's buffer
  - Read votes from the GC's memory to the main memory
  - Add per-channel votes on a candidate basis (this can be integrated with other operations *a posteriori*)

Algorithm 4.4: Vote projection using projective texturing and additive blending

4.4 Definition of the candidates' rendering buffer

The view ray, with the projected and accumulated vote maps, has to be rendered in a buffer associated with a virtual camera such that the rendered geometry can be captured. The final accumulated votes have to be read from this buffer to choose the best candidate. Therefore, in choosing the representation of candidates, the buffer, and the virtual camera parameters, special care should be taken to ensure that:

- All the candidates in the view ray's segment corresponding to the range have to be rendered;
- The post-processing required for reading and analyzing the accumulated votes and selecting the best 3D candidate is minimized.

To render a segment of the view ray, only the *start* and *end* 3D points have to be passed to the GC to define the 3D line. The corresponding connecting line is then rendered into the buffer. Since pixels are the smallest entities that can be represented in the buffer, each pixel rendered from the segment actually becomes a candidate.

The *start* and *end* points of the segment can be defined based on previous knowledge of the dimension of the scene being reconstructed. As mentioned in Section 3.3.2, if such information is not available, a large segment can be used initially to obtain a conservative estimate of the scene's dimensions.

Given the collinearity of candidates and the rectangular shape of buffers, the best way to maximize buffer usage is to set it up such that the segment is rendered as a straight line aligned to one of the viewport's axis (e.g. an horizontal line). Furthermore, the final reconstruction is simplified if a pixel selected from the buffer can be easily associated to the corresponding 3D candidate.

These goals will be accomplished using an orthographic camera $\mathbf{C}^O$ properly placed relatively to the segment. The orthographic camera's parameters and the corresponding relation between buffer position and 3D position of a candidate will be detailed in the remainder of this section. The final setup will make it possible to have a segment filling an entire 1-pixel-high buffer, and to associate a pixel's x coordinate in the buffer to the 3D position of the corresponding candidate, using a linear relation.

4.4.1 Extrinsic parameters of the virtual camera

The extrinsic parameters of $\mathbf{C}^O$ are defined as a function of the position, direction and length of the view ray's segment (Figure 4.5). Let $\overline{V_q^i}$ be the normalised vector of the direction of the view ray V_q^i. The "look at" point $\mathbf{l}$ is the middle point of the segment in 3D. The orthographic camera $\mathbf{C}^O$ can be placed in any position $\mathbf{c}^O$ in the plane perpendicular to $\overline{V_q^i}$ that contains $\mathbf{l}$. $\mathbf{L}^O$ is defined as the unit vector pointing from $\mathbf{c}^O$ to $\mathbf{l}$. $\mathbf{H}^O$ is parallel to $\overline{V_q^i}$. The "up" vector $\mathbf{U}^O$ is set perpendicular to $\mathbf{H}^O$ and $\mathbf{L}^O$. A graphical representation of the setup of the orthographic camera $\mathbf{C}^O$ is depicted in Figure 4.5.

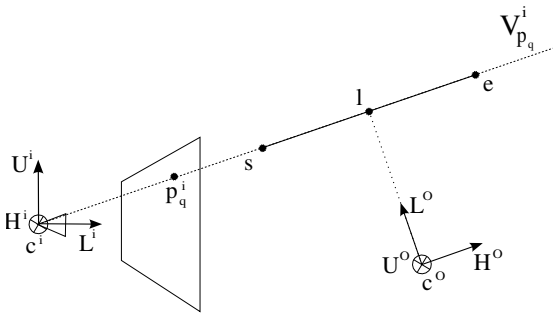


Figure 4.5: *Orthographic camera for view ray segment rendering.*
Assuming the view ray is fronto-parallel to the reader, the vectors $\mathbf{L}^O$ and $\mathbf{H}^O$ referring to $\mathbf{C}^O$ are also fronto-parallel to the reader, and $\mathbf{U}^O$ is pointing in the reader's direction.

Pseudo-Code 4.1 represents the setup of the extrinsic parameters of $\mathbf{C}^O$.

The *gluLookAt* call defines the camera's position to be $\mathbf{c}^O$, and orients it such that it is pointing (looking) directly at $\mathbf{l}$, which is the center of the segment, and the "up" vector is $\mathbf{U}^O$.

```
V_q^i = normalize(e - s);
H^O = V_q^i;
L^O = H^i × H^O;
l = (s + e) / 2;
c^O = l - L^O;
U^O = H^O × L^O ;
gluLookAt(c^O, l, U^O);
```

Pseudo-Code 4.1: Orthographic camera's extrinsic parameters' setup in OpenGL

4.4.2 Intrinsic parameters of the virtual camera

Assume that the segment's length in 3D, *segLen3D* has been defined. The segment size in pixels, *segSize2D*, corresponds to the number of candidates that will be rendered. This number is related to the final precision of the reconstruction itself. For a given predefined *segLen3D*, the larger the set of candidates, the higher the reconstruction precision[1].

The intrinsic parameters of $\mathbf{C}^O$ (namely the clipping planes and the viewport) are defined based on *segLen3D* and *segSize2D*. Each pixel corresponds to a 3D candidate of the view ray. Pseudo-Code 4.2 represents the setup of the intrinsic parameters of $\mathbf{C}^O$.

```
segLen3D = ||e − s||;
glViewport(0,0,segSize2D,1);
glOrtho( -segLen3D/2 , segLen3D/2 , -0.5 , 0.5 , 0.5 , 1.5 );
```

Pseudo-Code 4.2: Orthographic camera's intrinsic parameters' setup in OpenGL

The *glViewport* call defines the buffer's start coordinates $(0, 0)$ and dimensions $(segSize2D, 1)$ in pixels. The *glOrtho* call defines an orthographic camera with left, right, bottom, top near and far clipping panes, respectively. In this case, it defines that only the 3D geometry that lies in a volume of size *segLen3D* $\times$ *1* $\times$ *1* centered in front of the camera is to be rendered. This guarantees that the segment is contained within that volume, that it is totally rendered in the buffer, and that it completely fills the buffer.

An additional optimization may be performed. The transfer of memory blocks between the GC's memory and the main memory is a time-consuming operation, and it may only be performed when every request for rendering primitives in the buffer are actually completed i.e. when the buffer is *synchronized*. Rendering one line at a time and reading it into main memory would require requesting the rendering, waiting for synchronization, and reading the buffer. Such a way of operating would be a waste of the parallel capabilities of the GC.

To reduce the number of buffer synchronizations (i.e. waiting for everything being rendered to actually read the buffer) and buffer reading operations, multiple view rays may be rendered in a single buffer with multiple lines, varying the viewport's starting position (the y-coordinate) for each ray so that they do not overlap in the buffer.

[1] This increase in precision is limited by the resolution of the images.

By performing this batch rendering, the time spent in buffer synchronization should be reduced, as well as the time spent in memory transfers. For the sake of simplicity, the one-line buffer as described in Pseudo-Code 4.2 will be assumed in the remaining of this chapter, keeping in mind that it is straightforward to use a multi-line buffer to increase performance.

4.4.3 From pixel coordinates to 3D candidates

Having defined $\mathbf{C}^O$ as above, the segment's projection will completely fill the buffer, which will be one-pixel tall. In those conditions, the 3D coordinates of a point $\mathbf{r}_q^i$ in the view ray can be easily derived from its corresponding pixel with 2D window coordinates *(x,0)* using Equation 4.3.

$$\mathbf{r}_q^i = \mathbf{s} + x \left(\frac{segLen3D}{segSize2D} \right) \overline{V_q^i} \tag{4.3}$$

After setting the orthographic camera and the corresponding buffer, the view ray's segment can be rendered on that buffer using Algorithm 4.4. As a result, the rendering of the segment will completely occupy the buffer, which will contain an RGB value for each pixel of the segment. Each RGB value encodes the accumulated votes as described in Section 4.3.1. The GAPI provides a call to read the buffer data from the GC into a main memory array. If the votes were distributed over more than one channel, each candidate's final voting is computed by adding the values of each channel at the candidate's position in the array. After doing so, the vote analysis described in Section 3.3.5 can be performed iterating over the resulting array. The array positions correspond to the x coordinates in the buffer, and therefore Equation (4.3) can be used to directly determine the associated 3D candidate.

4.5 Final remarks

Given the computational load required for reconstruction from multiple images, and the desire to provide a reconstruction method that would be able to perform at interactive rates, an hardware-accelerated implementation of the reconstruction method was developed.

This chapter described the components of the point reconstruction method presented in Chapter 3 that were adapted to be executed in the graphics card. Vote collection was implemented by projecting a vote map over a view ray using projective texturing, a well-known technique from the computer graphics community. Vote accumulation was implemented with the use of additive blending, that allowed the accumulation of multiple vote maps projected over the same view ray by adding the corresponding values of the texture maps on a buffer.

The result was a buffer filled with the votes collected and accumulated for all candidates of a view ray, with all this process performed in the graphics card.

Some issues had to be solved due to limitation and specificities of graphic cards. Channel masking was used to overcome the fact that most common graphics cards have buffers limited to three or four channels of eight bits each. The buffer in which the candidates were rendered and the camera associated to it had to be properly set up so that the vote data for the candidates, accumulated in the GC's memory and copied to the main memory, could be aligned and easily accessed.

The execution of these tasks on the graphics card provided major speed improvements in the overall reconstruction process, when compared to a standard implementation on the CPU, as will be shown on Chapter 5.

Chapter 5

Tests and results of point reconstruction

5.1 Introduction

This chapter brings together the results of multiple experiments performed with the algorithms and implementations proposed in this thesis.

These experiments were carried out to evaluate several aspects of the reconstruction method proposed in Chapter 3 and Chapter 4, namely:

- The quality of the reconstruction;
- The robustness to errors in camera calibration;
- The overall robustness of the algorithm in adverse scene conditions, such as varying light conditions, shadows and apparent contours;
- The performance in terms of reconstruction time.

To numerically assess the accuracy and robustness, a set of synthetic test scenes with known 3D data and camera calibration was used. The use of synthetic scenes provided control over the data inputs and the noise introduced in the system, and ground truth data for error assessment.

Most literature performs error assessment based on re-projection errors of the reconstructions. Such evaluation does not allow to distinguish errors in reconstruction from errors in the camera calibration. A good reconstruction can have high re-projection error if there are strong errors in the calibration of some cameras. On rare occasions, the opposite may happen: a bad reconstruction may have low re-projection error if coincidentally incorrect camera calibration causes projections to fall in the correct place in 2D.

In the case of good camera calibration, some camera placement settings may be ambiguous, such that a good re-projection does not mean a good reconstruction (e.g. a

point on an horizontal line that always projects on a contour in different images taken by an horizontally-translating camera).

Due to the ambiguity caused by the use of re-projection metrics to assess the quality of the reconstruction, an error metric based on 3D ground truth is proposed to assess the reconstruction in synthetic scenes, that aims to be applicable even in the presence of camera calibration errors.

Section 5.2 introduces the context, error metrics and the tools used to assess the method's performance in terms of reconstruction quality and robustness.

The influence of the method's relevant parameters in the reconstructions was evaluated using the aforementioned metrics and tools. The results and corresponding analysis are presented throughout Section 5.3, providing insight on how to best tune the parameters. The response of the algorithm to the introduction of noise in the camera calibration was also tested, and it is reported in Section 5.4.

To assess the practical applicability of the methods, the algorithms were also tested with real scenes, as presented in Section 5.5. A commercial package [EOS] was used for user-assisted camera calibration. These scenes contain real error sources (e.g. image noise, camera calibration errors, highlights), thus becoming realistic tests for the proposed method.

However, in these settings, it is difficult to quantify reconstruction errors. Reprojecting points in images would not be of use, since the errors in camera calibration would influence point offsets. It would be difficult to distinguish between offsets caused by camera calibration, by reconstruction errors or both. Directly measuring points in the scene allows to do some assessment of the reconstruction quality, but it is a process that is also prone to inaccuracy due to human errors in the measuring, and the problem of knowing to which point in the scene corresponds a reconstructed point. Still, direct measuring provides a rough approximation of the quality of the reconstruction. Therefore, results on real scenes will be judged mostly visually, although some direct measurements are used to provide an approximate idea of the proportionality of the reconstructions, keeping in mind that there are measuring and approximation errors involved.

Reconstruction results of all the tests presented here, and of additional scenes can be found in the CD accompanying this thesis. The contents and usage instructions for the CD can be found in Appendix A, and some additional reconstruction images are presented in Appendix B.

Finally, tests were performed to analyze and compare the reconstruction times required by a CPU implementation of the method versus a graphics hardware implementation, in different machines. These results are presented in Section 5.6, and the chapter is concluded in Section 5.7.

5.2 Quality assessment on synthetic scenes

The reconstruction algorithm described in Chapter 3 produces 3D point estimates, which are analyzed and either accepted as *valid* or rejected as *invalid*. This classi-

fication is done based on the parameters presented in Section 3.3.5 and complemented with cross-checking in Section 3.4. These parameters represent the desired level of confidence on the estimates returned by the algorithm.

The algorithm may therefore be divided in two parts: the estimation and the classification.

For evaluation purposes, if the estimates resulting from the reconstruction can be compared to ground truth data, it is possible to determine if a given point was correctly or incorrectly reconstructed or not i.e., if its estimated position is the same as (or very close to) the world point that was being reconstructed in the first place. This can be done regardless of the algorithm classification as valid or invalid.

The algorithm will therefore be judged at two levels:

- the correctness (or quality) of the estimates, compared to ground truth data;
- the validity classification performed by the algorithm (this will be referred as *classifier's performance*).

The quality of the classification is important, because there will always be a number of incorrect points in the reconstruction, and the better the classification algorithm, the better the filtering of those points. On the other hand, good classification is of no use, if the point estimation is in itself bad. If, for example, only 1% of the points are correctly reconstructed, and classified as valid, then classification is very good, but the final reconstruction will be very poor.

It is therefore important that the algorithm:

- produces correct estimates in sufficient number;
- properly classifies estimates as valid or invalid i.e., it classifies most correct estimates as valid, and most incorrect estimates as invalid.

To assess the correctness of the estimates, it is necessary to have an error metric that allows the comparison between a point estimated by the algorithm and a real point. A low error between estimated and real data represents a correct reconstruction. The choice of this metric is discussed in Section 5.2.1. Section 5.2.2 shows how this metric can be used to evaluate the ability of the reconstruction algorithm to classify the validity of its own estimates. Section 5.2.3 describes the generic setup used for the tests presented in Section 5.3 and Section 5.4.

5.2.1 Error metric

A natural error measure for an estimate is the geometric distance between the estimated point and the real point. However, it is not always straightforward to determine which is the real world point (w-point) $\mathbf{w}_q$ corresponding to a reconstructed point (r-point) $\mathbf{r}_q^i$. Consider the case of Figure 5.1, where a contour point was incorrectly reconstructed from camera $\mathbf{C}^1$ as $\mathbf{r}_q^i$, being placed on the opposite ear of the bunny, whereas the

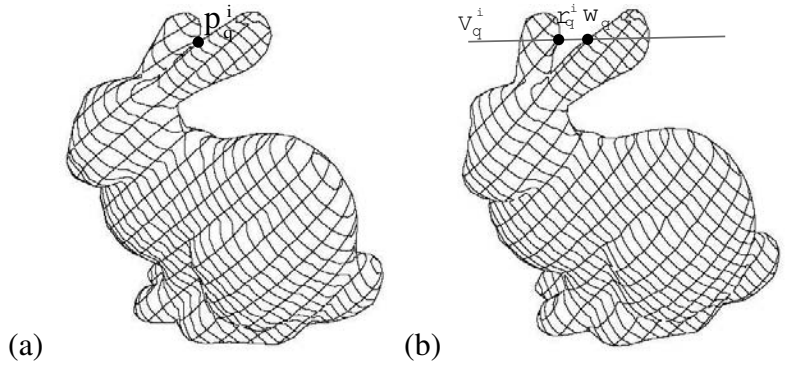


Figure 5.1: *Problems using all the known w-points to compute reconstruction error.*

(a) Reference view and selected point

(b) A view showing the view ray and the points at the surfaces. If all w-points were used to compute the error, the point reconstructed on the farthest ear ($\mathbf{r}_q^i$) would have a low error, since it is close to a surface, but the correct point should be on the closest ear's surface ($\mathbf{w}_q$).

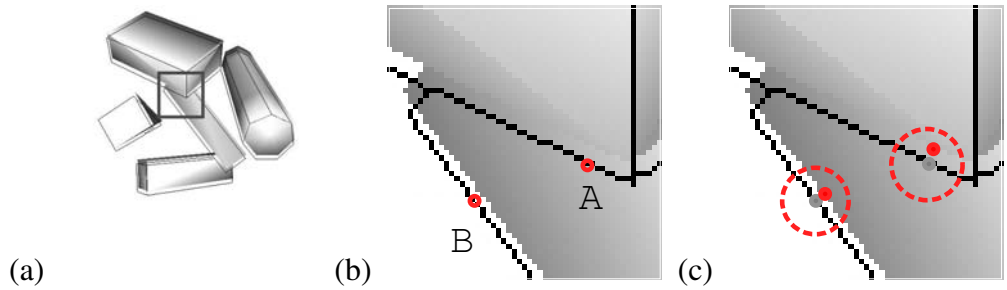


Figure 5.2: *Example of contour offsets.*

(a) Contours superimposed with incorrect view of a scene (the camera calibration is not correct)

(b) A zoomed view of the highlighted area on the left. Due to camera errors, the view ray of point A intersects the background object, while the view ray of point B does not intersect any object.

(c) Considering a neighborhood of the points in the image, the choice of which world points to use to compute the error is improved.

correct point should be $\mathbf{w}_q$. Although $\mathbf{r}_q^i$ is on a surface of the scene, the distance to the correct point is significant. In this case, selecting the point of the object's surface that is closer to $\mathbf{r}_q^i$ as the world point associated to $\mathbf{r}_q^i$ would result in an unfair low error.

This suggests that the measuring should be performed taking into account the reference camera from where the point was first selected, whenever possible. In an ideal situation, where camera calibration and contour extraction is perfect and there is no discretization, $\mathbf{w}_q$ would be the point closest to the reference camera where the view ray of $\mathbf{r}_q^i$ intersects some object of the scene.

In less-than-ideal situations, camera calibration errors and other factors may lead to displacements of the contour point that originated $\mathbf{r}_q^i$, resulting in displacements of the view ray relative to $\mathbf{w}_q$ (Figure 5.2).

This offset results in particularly misleading errors in cases where $\mathbf{r}_q^i$ is close to an occlusion contour. A small offset may cause the view ray to intersect a completely different object in the background (such as point A in Figure 5.2(a)) or not intersecting any object at all (point B in Figure 5.2(b)), while the estimate is actually close to $\mathbf{w}_q$.

To account for these situations, the $\mathbf{w}_q$ associated to $\mathbf{r}_q^i$ will be the one that is closer to $\mathbf{r}_q^i$, and visible to the reference camera within a neighborhood of 10 pixels of the corresponding $\mathbf{p}_q^i$ (Figure 5.2(c)).

Let $\mathbf{p}_q^i$ be the projection of a reconstructed point $\mathbf{r}_q^i$, $N(\mathbf{p}_q^i)$ the set of points in the neighborhood of $\mathbf{p}_q^i$ in the image, and $\mathbf{w}_s$ be the world point closest to $\mathbf{C}^i$ that projects as $\mathbf{p}_s^i$ in $\mathbf{C}^i$. In this case, the real world point that is associated to $\mathbf{r}_q^i$ is defined as presented in Definition 5.1.

Definition 5.1 *Real world point $\hat{\mathbf{w}}_q^i$ associated to a reconstructed point $\mathbf{r}_q^i$*

$$\hat{\mathbf{w}}_q^i = \underset{\mathbf{w}_s : \mathbf{p}_s \in N(\mathbf{p}_q)}{arg\,min}^{3} \; ||\mathbf{r}_q^i - \mathbf{w}_s||$$

□

The reconstruction error of $\mathbf{r}_q^i$ is its distance to the associated world point $\hat{\mathbf{w}}_q^i$ as presented in Definition 5.2.

Definition 5.2 *Reconstruction error of a point*

$$e_r(\mathbf{r}_q^i) = ||\mathbf{r}_q^i - \hat{\mathbf{w}}_q^i||$$

□

The *Z-buffering* capabilities of current graphic API's provide a straightforward mean to obtain the coordinates of the real points. By rendering the known scene using Z-buffering from the same reference camera used to reconstruct $\mathbf{r}_q^i$, it is possible to obtain a depth map that indicates the real depth for each w-point in the image[4]. In this way, it is possible to obtain the coordinates of the real w-points that project on the neighborhood of $\mathbf{p}_q^i$.

The real world point $\mathbf{w}_q$ that will be associated to the reconstructed point $\mathbf{r}_q^i$ will be the w-point with the smaller 3D distance to $\mathbf{r}_q^i$ on that close neighborhood, as per Definition 5.2.

5.2.2 The reconstruction algorithm as a classifier

For each contour point, the reconstruction process can be divided in two parts: first it *estimates* the point's 3D coordinates, and then it *classifies – without knowledge of*

[3] $argmin$ provides the value of the given argument (specified under $argmin$) for which the value of the given expression (to the right of $argmin$) attains its minimum value.

[4] The precision of the Z-buffer (i.e. the number of bits used to represent a depth value) should be taken into account. For the purpose of these tests, it was guaranteed that it exceeded by far the requirements for the magnitude of errors that were considered.

ground truth data – the reconstructed point as *valid* or *invalid*, according to e.g. evidences of cross-voting or plateaus. The output is a set of reconstructed points labeled as valid or invalid, of which only the valid are actually used, i.e. *accepted*, and the invalid are *rejected*.

The quality of the classification performed by the algorithm's *classifier* is an important factor in the quality of the overall final reconstruction. Inaccurate rejection or acceptance of points can lead to the loss of important points or the acceptance of outliers.

To assess the *performance* of a classifier, its classification should be compared with an ideal or expected classification. This ideal classification can be obtained by comparing the estimated coordinates with real data. Points with a very low reconstruction error (as per Definition 5.2) are ideally classified as *correct*, and other points are considered *incorrect*.

The two types of classification – the one estimated by the algorithm's classifier in the form of *validity* and the one provided by real data in the form of *correctness* – lead to four different combinations:

- incorrect points classified as valid - *false positives*;
- correct points classified as valid - *true positives*;
- correct points classified as invalid - *false negatives*;
- incorrect points classified as invalid - *true negatives*.

These four classes make up what is known as a *confusion matrix*, or *contingency table*, in the literature of different fields, ranging from medical decision making to machine learning, information retrieval and data mining. A series of ratios and tools have been derived from these matrices, to help evaluating and organizing classifiers. This section presents an overview of confusion matrices, and the ratios and metrics based on them that are relevant for the problem at hand. The reader is referred to the literature [Koha 98, Frak 92] for more detailed information on these techniques.

Confusion Matrices

A *confusion matrix* contains information about real and estimated classification performed by a classifier. Given a set of estimates and a classifier for a given class, each estimate is classified as either belonging to the class (estimated positive – EP), or not belonging to the class (estimated negative – EN)[5]. If real data is available i.e. if it is possible to know if an estimate is really negative (RN) or really positive (RP), one can determine in which cases the classifier correctly identified estimates as being positive and negative, and in which cases it did not.

An estimate correctly classified as valid is considered as a *true positive* (tp), and an estimate incorrectly considered as valid is considered a *false positive* (fp). Similarly,

[5]This case may also be seen as a two-class classifier, the positive class and the negative class.

		Estimated		
		Negative (invalid)	Positive (valid)	Total
Real	Negative (incorrect)	tn	fp	RN
	Positive (correct)	fn	tp	RP
	Total	EN	EP	T

Table 5.1: *A confusion matrix.*

an estimate correctly classified as invalid is considered as a *true negative* (tn), and an estimate incorrectly considered as invalid is considered a *false negative* (fn). These four cases are used to construct a confusion matrix as presented in Table 5.1. In the context of the problem at hand, the classifier is the algorithm that determines if an estimate is valid or invalid, and the correctness of the classification is determined based on the ground truth data and the error metric defined in Section 5.2.1.

Evaluation Ratios

A series of ratios can be computed from a confusion matrix, yielding different ways to analyze the performance of the classifier [Koha 98, Frak 92]. There are two ratios that are particularly meaningful in the context of this work, namely *precision* (PR) and *recall* or *true positive rate* (TP).

Recall or True Positive Rate (TP) The proportion of correctly classified positive cases, relative to the the total number of real positives:

$$TP = \frac{tp}{fn + tp} = \frac{tp}{RP}. \tag{5.1}$$

Precision (PR) The proportion of correctly classified positive cases, relative to the the total number of estimated positives:

$$PR = \frac{tp}{fp + tp} = \frac{tp}{EP}. \tag{5.2}$$

Recall measures how good is the classification of correct points as valid. A high recall value means that most correct points were classified as valid, and conversely, few correct points were rejected.

Precision represents a measure of how good the final reconstruction is i.e., how many of the points classified as valid are actually correct. A high value of precision means that the estimates selected by the algorithm as valid are mostly correct. It indicates the level of confidence that one may have in the quality of final reconstructed points.

In most cases there is a trade-off between recall and precision, governed by the choice of parameters. On one hand, if the focus is set to a high level of confidence in the final reconstructed points, regardless of their amount, one should use stricter, more conservative, parameter settings. This normally leads to a higher rejection rate, meaning that there will be more correct points being rejected as invalid – lower recall – but the percentage of the points accepted as valid that are correct is higher, and consequently the number of accepted outliers is low – higher precision. On the other hand, if there is no need for high precision, but there is a requirement for a higher number of points, using more liberal parameter settings does the reverse: more points are accepted as valid, meaning more correct points considered valid – higher recall – but also more incorrect points – lower precision.

These effects are reflected visually in the reconstructions: reconstructions with high recall show a higher number of points, and reconstructions with low precision show a higher number of outliers and scattered points. Experiments have shown a good relation between these ratios and the visual perception of what is a good or bad reconstruction. This was the main motivation to use them to evaluate the reconstructions.

Which is more important (precision or recall) depends on the particular application of the reconstruction (for example, for measuring objects, precision is more important, while for generating virtual low-resolution objects, a higher number of points could be of more importance).

The tests presented in this chapter were performed to assess the influence of some parameters of the algorithm in this trade-off, in order to provide information and guidance in the choice of those parameters.

To assist in this task, the results are presented in the form of *Precision/Recall charts*. These are graphical representations of the TP ratio over the PR ratio. Each chart refers to the testing of a given parameter with different values, in a set of scenes. The charts will show how each parameter influences the individual reconstructions.

5.2.3 Test setup

Several reconstruction tests were performed on a set of synthetic scenes, by varying key parameters of the algorithm to test the response in terms of the resulting geometric and classification errors. Table 5.2 contains a brief description of the parameters tested on those scenes.

Each test consisted of running the reconstruction algorithm on a scene, with the set of parameters specific to the test. The reconstruction was performed n times, each one using one of the n images of the scene as the reference. An error value was computed for each reconstructed point, according to Definition 5.2, and the classification that the algorithm attributed to that point (valid or invalid) was also stored. In the end, a confusion matrix was computed, containing all the reconstructed points collected from the different reference cameras. The corresponding ratios were also computed, and used to create the precision/recall charts.

To compute the ideal classification based on real data, a threshold was set on the

Parameter	Description	Values tested	Default Value
Set of points	Use either the full set of contour points or a set of feature points obtained with this simplification tolerance (Section 3.5)	{0 (full), 1.2, 2, 5, 10}	1.2
Vote Range	The maximum vote value (corresponding to a contour). Lower values correspond to thinner vote maps (Section 3.3.3)	{3, 10} (votes)	3 votes
Cross-Check Threshold	Points that project in the vote map of similar cameras into vote values lower than this are rejected (Section 3.4.1)	{0..10} or {0..3} (nr. of votes, depends on vote range)	1 vote (assuming a vote range of 3)
Similar Camera Angle	The maximum angle between two cameras' look-at vector to consider them similar (Section 3.4.1)	{10..50} (steps of 10 deg.)	30 deg.
Vote Percent Threshold	All local maxima above this percentage of the number of votes of the global maximum are considered as potential winners (Section 3.3.5)	{50%..100%} (% of votes of global maximum)	90%
Nr. of Candidates	The number of candidates (Section 3.3.2)	{200 .. 1400} (candidates, steps of 200)	800 cands.
3D Merge Threshold	Maximum 3D distance allowed between two points for merging them (Section 3.4.2)	{.01%, 0.05%, 0.1%, 0.5%} (percentage of scene scale)	0.5% of scene scale
2D Merge Threshold	Maximum 2D distance allowed between two points for merging them (Section 3.4.2)	{1,2,3} (pixel distance)	3 pixels

Table 5.2: *Parameters tested with synthetic scenes.*

error metric. Since this error is scale-dependent, the threshold was based on the scene's dimensions. Points for which $e_r(\mathbf{r}_q^i)$ is below 0.5% of the range used for reconstruction (which is typically also related to the scene's dimensions) are classified as *correct*, and other points are classified as *incorrect*. For example, a correct estimate in a scene with a range of one meter has an error of at most 5 millimeters (in practice many of them have a significantly lower error value).

5.3 Parameter testing with synthetic scenes

This section presents the results of testing five synthetic scenes with different settings and variations of selected parameters, and using exact camera calibration.

First, the individual scenes will be described, together with some highlights of important details and problem sources. The actual tests follow. The results of each test are presented in the form of a Precision/Recall chart, followed by a table containing *fp*, *tp*, *fn*, *tn* and the relevant ratios for each scene and parameter values. The results will also be judged visually, comparing views of the reconstructed points of selected scenes, with different parameter values.

A first test was performed to evaluate the differences between reconstructing all points and reconstructing only feature points extracted at different levels of detail, using the exact camera calibration available from ground truth.

The following set of tests was performed using the default values of the selected parameters, and varying one parameter in each test.

5.3.1 Description of synthetic scenes and their contours

Five synthetic scenes were used to evaluate the quality of the method in terms of point reconstruction and classification. Figure 5.3 shows views of the different test scenes. Each scene has been photographed with a virtual camera from 69 positions around it, at a resolution of 512x512 pixels. The cameras were spaced and with varying angles, in a sphere surrounding the scene, representing medium to large baseline settings.

Below is a brief description of each scene and their important features.

arcade (Figure 5.3(a)) Highly symmetric and regular object, with significant occlusion between the columns and from top and bottom views. Apparent contours on the roof and the column, with dense contours on the columns in many images. The lighting was fixed relative to the camera, meaning that the appearance of the object had slight variations from image to image. These variations would cause problems in e.g. color-consistency -based methods.

bunny (Figure 5.3(b)) The Stanford Bunny [Turk 94], with a synthetic and repetitive texture. Apparent contours, some occlusion caused by the ears and concavities. Smooth surface, texture provides contours. Even if small-baseline methods were applicable, the repeating pattern would lead to high ambiguity.

carriage (Figure 5.3(c)) A more complex object, with many occlusions. Some mild creases, such as the one at rooftop, are not always detectable. The distance to cameras of some parts of the object (e.g. the holding bars in the front) varies significantly.

chairs (Figure 5.3(d)) Separate objects, with self and inter-occlusion and mostly apparent contours.

solids (Figure 5.3(e)) Multiple isolated objects occluding each other. Some creases did not produce contours in some images.

Thin-contour extraction was performed on the input images, as it would be done in real scenes. This means that there are missing and displaced contours, as well as apparent contours, or incorrect contours. Some of these problems are shown in the contour maps and their close-ups in Figure 5.3, namely:

- In (a), the missing contours on the step of the arcade, the isolated small lines on the floor the base, and the artifacts and missing contours in the columns;
- In (b), the gaps in some contours and the existence of an overall apparent contour;
- In (c), the high density of contours in the wheel, and some missing ones;
- In (d), the same problem in the chair border, plus some contours from shading (the three vertical lines on the side);
- In (e), the gaps in the contour lines on the base of the pyramid.

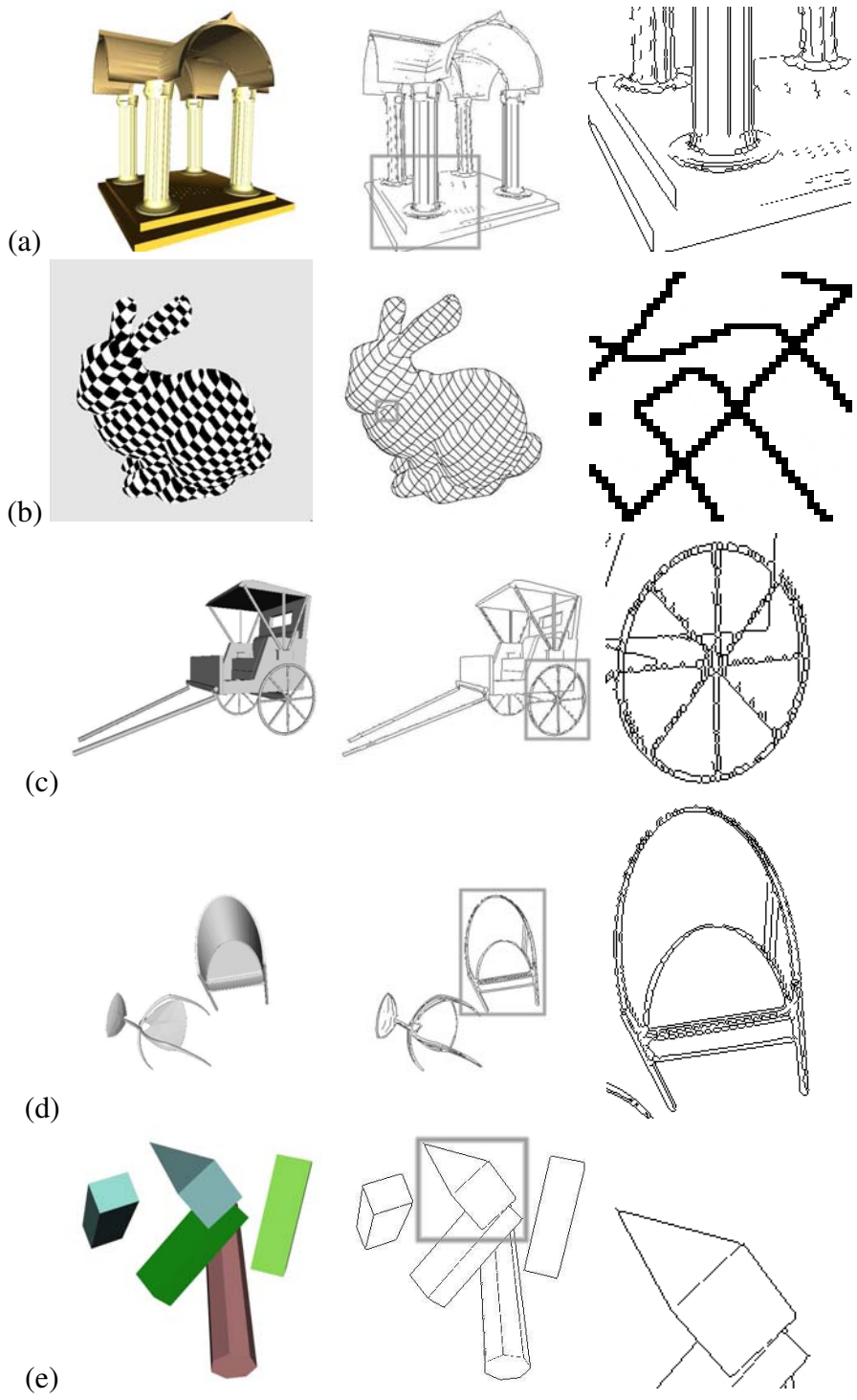


Figure 5.3: *Synthetic scenes*.
One image of each of the synthetic scenes used for testing, followed by the contour map and a close-up of the contour map.

These contour maps and associated problems represent a real source of errors for the tests.

The type of scenes chosen represent different situations that are typically not trivial to treat with a single reconstruction method. For instance, the perspective effects caused by the varying camera angles and positions, the multiple occlusions and the repetitive patterns would cause troubles in correlation-based methods. Color-consistency-based methods are likely to have some problems with the slight lighting variations in some scenes. The assumption of regularity explored to generate smooth surfaces by methods such as Pollefeys' would produce poor results in the scenes with separate or self-occluding objects, such as those used here.

5.3.2 Set of points

The use of feature points was proposed to reduce the set of points to reconstruct (Section 3.5). Such a reduction could be useful for e.g. fast previewing of a reconstruction, or for reconstructions with lower requirements on detail that could be used as robust initialization data for other type of reconstruction. A test was performed to assess if selecting only these points has a significant influence in the quality of the reconstruction, and if they are representative of the complete set of points. The test consisted of comparing reconstructions using all points and reconstructions using sets of feature points at different levels of detail. The level of detail was controlled by varying the tolerance of the line simplification algorithm, as referred in Section 3.5.2.

Figure 5.4 shows the difference in the reconstructed point sets of the solids and the chairs scenes, between the full point set (top) and a feature point set (bottom). For this test, the level of detail chosen for the extraction of feature points was high, meaning that a large number of features was identified.

In the chairs scene, the apparent reduction in the number of points is not very high, due to the high number of curved lines, but still it represents a reduction of about 3.8 to 1. On the other hand, in the solids scene, there is a more noticeable decrease of point density, explained by the high number of straight lines existent in the scene. Although intuitively one would expect a higher reduction in this case, the fact that there are contours that are broken in various points give rise to multiple extremities, which are not removed in the simplification. Nevertheless, the reduction is of about 13 to 1. In both cases, the reduced point sets provide a visually representative subset of the original sets.

Figure 5.5 shows the precision/recall chart comparing both types of reconstruction for the five scenes. The main chart on the left shows a *close-up* of the area containing all the data points, to better understand the relation between different data points. In this case, the precision axis ranges from 0.8 to 1.0, and the recall axis from about 0.95 to 1. To understand the global quality of the results, the *full chart with axes ranging from 0 to 1 is shown at a smaller scale, on the right*, with the close-up area marked by a rectangle. These two views will be used for all precision/recall charts presented throughout this chapter, with the scale of the axes of the close-up view adjusted according to the data range.

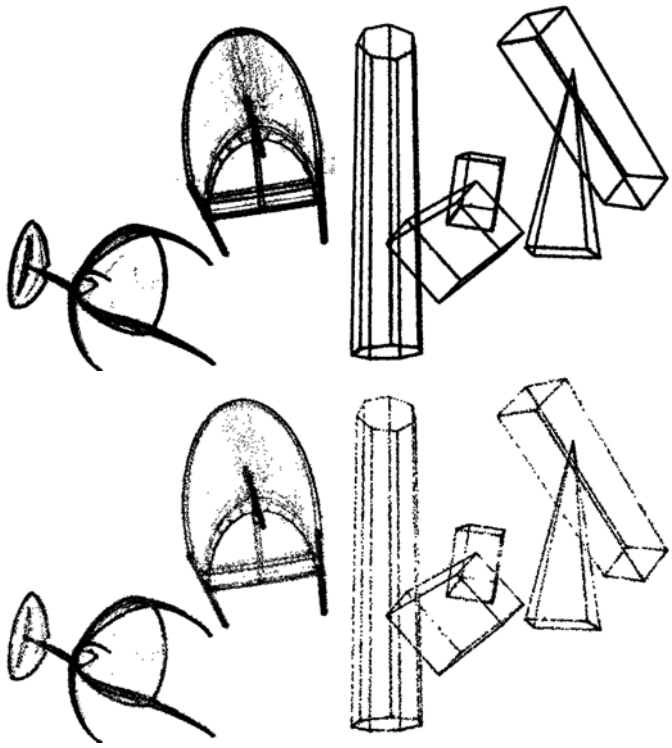


Figure 5.4: *Point set tests (reconstruction views).*
Reconstruction of all contour points (top) versus reconstruction of feature points only, using a line simplification tolerance of 1.2 pixels (bottom)

The values of the various ratios that served as basis for the chart are also presented in the tables of Figure 5.5. The maximum value of each column of a table is emphasized in bold face, and the minimum in italic. This type of table will also be used to present the results of the other tests.

As the chart shows, the recall values are high for all scenes, reaching its maximum with the bunny scene, regardless of the point set reconstructed. Precision is also high for most scenes, except for the arcade scene. This was expected, since the contours on the columns of the arcade are very dense and their extraction error-prone. Still, precision above 0.8 is achieved.

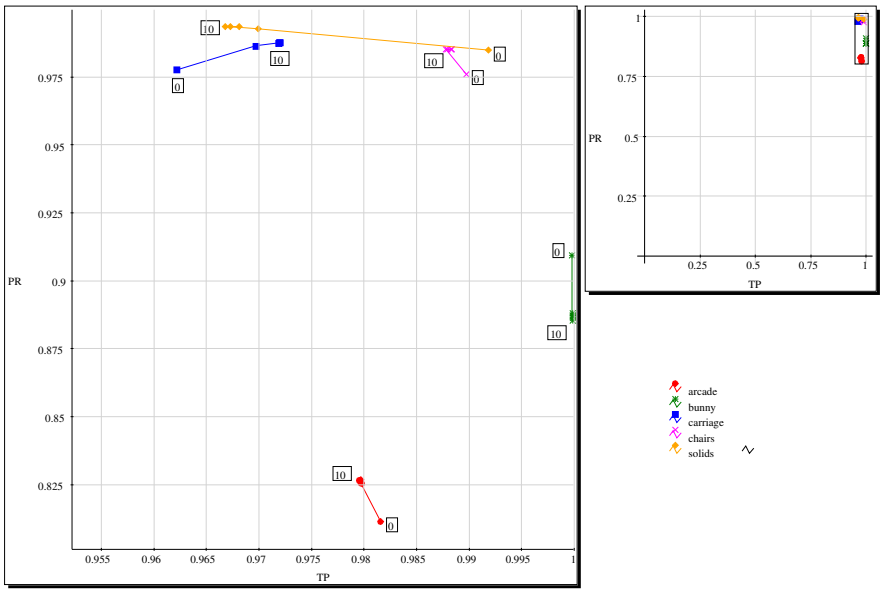
The differences in precision between reconstructions of the same scene are small, under 0.01 for scenes with lower contour density (carriage, chairs and solids) and under 0.03 in scenes with high contour density (arcade and bunny).

In terms of recall, the scenes also exhibit small differences, under 0.01 for most scenes, with the exception of the solids scene, with an increase of 0.022 when using all the points.

The reduction in the total number of points (including the rejected) varies from 3.8 to 1 (chairs and carriage with a tolerance of 1.2 pixels) over 15 to 1 (solids with a tolerance of 10). These ratios are approximately maintained if only the valid points are considered.

The visual analysis and the precision/recall values indicate that feature points are

a representative subset of the complete set of points, and may be used in cases where high point density is not required. This has the added benefit of reducing the number of computations and the reconstruction time.



arcade							
lineSimpTol	fp	tp	fn	tn	EP	TP	PR
0	**70843**	**304281**	**5708**	**259129**	**375124**	**0.982**	*0.811*
1.2	14020	66689	1391	60664	80709	*0.980*	0.826
2	13064	62247	1290	56700	75311	*0.980*	**0.827**
5	12847	60835	1259	55525	73682	*0.980*	0.826
10	*12832*	*60569*	*1248*	*55280*	*73401*	*0.980*	0.825

bunny							
lineSimpTol	fp	tp	fn	tn	EP	TP	PR
0	**68631**	**687646**	**134**	**236491**	**756277**	1.000	**0.909**
1.2	8658	66856	10	26038	75514	1.000	*0.885*
2	7548	59855	10	22811	67403	1.000	0.888
5	7107	55979	*8*	21376	63086	1.000	0.887
10	*7028*	*54961*	*8*	*21049*	*61989*	1.000	0.887

carriage							
lineSimpTol	fp	tp	fn	tn	EP	TP	PR
0	**6553**	**284833**	**11188**	**127164**	**291386**	*0.962*	*0.978*
1.2	1071	77221	2411	32930	78292	0.970	0.986
2	935	72756	2105	30260	73691	**0.972**	0.987
5	907	71559	2064	29593	72466	**0.972**	0.987
10	*898*	*71270*	*2049*	*29447*	*72168*	**0.972**	**0.988**

chairs							
lineSimpTol	fp	tp	fn	tn	EP	TP	PR
0	**4056**	**165332**	**1714**	**56640**	**169388**	**0.990**	*0.976*
1.2	671	44287	546	14269	44958	*0.988*	**0.985**
2	629	41589	507	13385	42218	*0.988*	**0.985**
5	614	40656	481	13045	41270	*0.988*	**0.985**
10	*613*	*40439*	*476*	*12992*	*41052*	*0.988*	**0.985**

solids							
lineSimpTol	fp	tp	fn	tn	EP	TP	PR
0	**1873**	**121064**	**995**	**71358**	**122937**	**0.992**	*0.985*
1.2	69	9294	288	4874	9363	0.970	**0.993**
2	58	8543	281	4386	8601	0.968	**0.993**
5	55	8257	*279*	4164	8312	*0.967*	**0.993**
10	*54*	*8133*	*279*	*4071*	*8187*	*0.967*	**0.993**

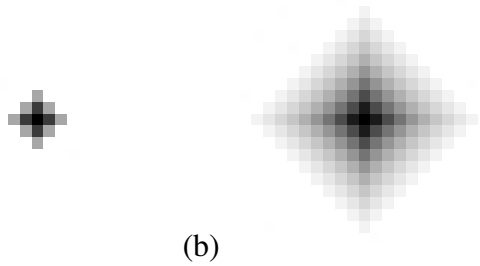
Figure 5.5: *Point set tests (data chart).*

5.3.3 Vote map range

The vote map range is associated with the limit of the distance transform used for computing a vote map (see Section 3.3.3). In practice, it defines how wide the distance transform extends from a contour. Consequently, it determines how far a projected point can be from a contour to still receive votes from it.

The practical results are that, for lower values of this parameter, one reduces the amount of cross voting, but also reduces the tolerance to offsets caused by e.g. camera calibration errors or faulty contour detection. For higher values, there is more tolerance to errors, but higher probabilities of cross-voting.

Figure 5.6 shows the neighborhood of one contour pixel in two different vote maps, with ranges of 3 and 10, and using a mask of $a = 1$, $b = 2$ (darker shades represent higher voting).



(a) (b)

Figure 5.6: *Vote map neighborhoods of one pixel using different vote map ranges*.
(a) Vote map ranging from 3 to 0 (dark to light)
(b) Vote map ranging from 10 to 0 (dark to light)

Figure 5.7 shows a part of the vote maps generated for one image of the bunny scene with each of the ranges. It is noticeable that, with a range limit of 3, most of the interior of the patches enclosed by contours have no votes. On the other hand, with a range limit of 10, the interior of most of the patches are filled with votes. This means that most candidates projecting in the interior of a patch will have no votes in the first case, and a significant amount of votes in the second.

Figure 5.8 shows a comparison of reconstructions with the two different vote map ranges, in two different scenes (solids and bunny). While in the solids scene there is small visual difference, in the bunny there are clouds of points cluttering the result of the bottom reconstruction (range limit=10), more evident inside the ears and in the base. This is in line with the problem referred before. In the solids scene, the contours are sufficiently apart, so the votes of one contour are not likely to affect the votes of other. That is not the case in the bunny scene, so there are some points that are incorrectly reconstructed by gathering votes from the interior of patches.

The precision/recall chart and the corresponding data table presented in Figure 5.9 confirm these results. For each scene, the variation in precision and recall is in general very small, regardless of the vote range limit. The bunny scene has a slightly higher decrease of precision in the order of 0.04, when the limit of 10 is used, that corresponds

to a significant increase of the proportion of false positives, compared to the proportion of true positives gained. These results suggest that if there is a significant number of contours that are close to each other, the range of the vote maps should not by set too high.

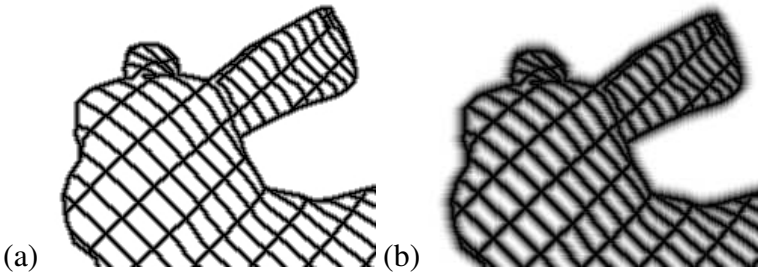


Figure 5.7: *Vote maps with different ranges in the case of close contours*.

(a) The smaller range (3) provides areas with no votes inside the regions enclosed by contours.

(b) The larger range (10) extends votes farther away from contours, such that a point projected in the middle of a closed region still collects a significant amount of votes

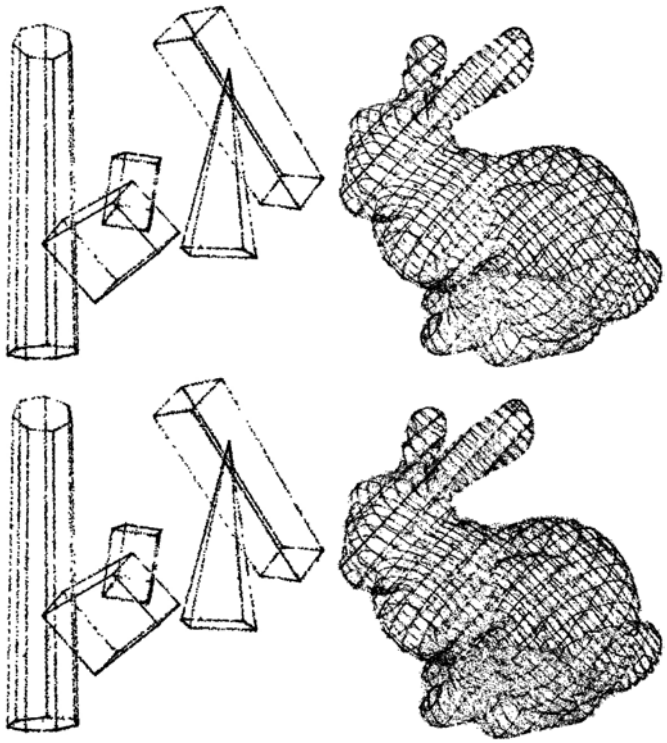
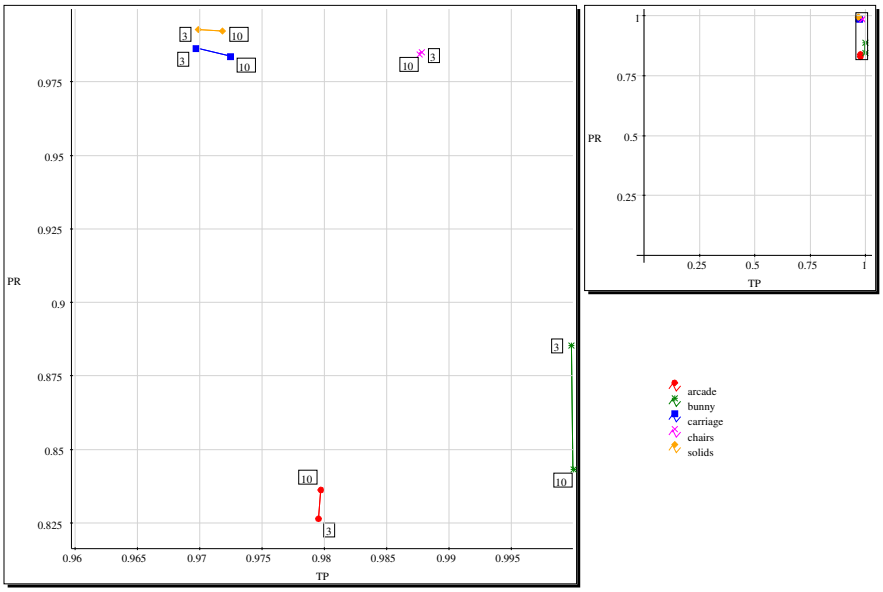


Figure 5.8: *Vote range tests (reconstruction views)*.

Reconstruction using a range of 3 votes (top) versus a range of 10 votes (bottom)



arcade

dtDir	fp	tp	fn	tn	EP	TP	PR
10	*13473*	**68761**	**1420**	*59110*	**82234**	0.980	**0.836**
3	**14020**	*66689*	*1391*	**60664**	*80709*	0.980	*0.826*

bunny

dtDir	fp	tp	fn	tn	EP	TP	PR
10	**13167**	**70799**	*2*	*17594*	**83966**	1.000	*0.843*
3	*8658*	*66856*	**10**	**26038**	*75514*	1.000	**0.885**

carriage

dtDir	fp	tp	fn	tn	EP	TP	PR
10	**1299**	**77849**	*2205*	*32280*	**79148**	**0.972**	*0.984*
3	*1071*	*77221*	**2411**	**32930**	*78292*	*0.970*	**0.986**

chairs

dtDir	fp	tp	fn	tn	EP	TP	PR
10	**699**	*44129*	**552**	**14393**	*44828*	0.988	*0.984*
3	*671*	**44287**	*546*	*14269*	**44958**	0.988	**0.985**

solids

dtDir	fp	tp	fn	tn	EP	TP	PR
10	**74**	**9459**	*274*	*4718*	**9533**	**0.972**	*0.992*
3	*69*	*9294*	**288**	**4874**	*9363*	*0.970*	**0.993**

Figure 5.9: *Vote range tests (data chart).*

5.3.4 Cross-check threshold

The crossCheckThreshold parameter sets a level of confidence in the estimates (Section 3.4.1). An estimated point is projected in cameras similar to the reference camera; if the number of votes in the projected position is below this threshold in at least one of the similar cameras, the point is rejected as invalid. This rejection is based on the assumption that a point lying on a contour on the reference camera should also lie on (a close neighborhood of) a contour in similar cameras. This is expected to eliminate much of the cross-voting problem. This threshold can only vary within the vote range (in this case set to 3), and therefore it was tested with only values between 0 (more liberal) and 3 (more conservative).

Figure 5.10 shows the points reconstructed for two scenes using the different cross-check threshold values. The effect of this parameters is quite noticeable in the arcade

scene. With a value of 0 (top), a large number of incorrect points are accepted as valid, cluttering the whole reconstruction. This is caused by a number of factors. The high density of the contours in the columns and the fact that the columns are aligned with each other in a number of views, leads to view rays intersecting a large number of contours in some images and results in cross voting. Such effect extends to points on the roof as some of the cross-voting occurs also with cameras aiming up from the bottom. Using the value 1 eliminates a significant number of these incorrect points. Increasing to 2 eliminates a few more incorrect points, but the rejection of correct points starts to be noticeable, with the columns and parts of the roof becoming sparser (yet still acceptable). At the limit of 3, the sparseness is evident over all the scene.

In the carriage scene, the effects are not so evident, since the amount of cross voting is significantly smaller. But it is possible to notice some outliers in the images with lower threshold that are not present in the images with higher threshold, namely inside the wheels and close to the rooftop front edge. Some denser areas, like the long bars in front of the carriage, become sparser as the threshold increases. Again, in the limit of 3, the sparseness becomes evident. The remaining scenes present a similar behavior to the variation of this threshold.

The chart in Figure 5.11 shows that in general, for threshold values between 2 and 3, the loss in recall is significant, with very small variations in precision. On the other hand, for values between 0 and 2, the variations in precision are more significant, reaching up to 0.08 in the case of the bunny, with small changes in recall.

There is a noticeable decrease in precision in the case of the arcade when increasing the threshold value from 1 to 2. This is justified by the extreme contour density occurring at the columns, and the consequent strong cross voting. While a correct point e.g. on the roof may be rejected in a stricter cross-checking due to a missing or displaced contour on a similar camera, many points lying in the neighborhood of the columns' surfaces (e.g. inside them) still get accepted, since there are so many close contours in that region that a match in the cross-checking stage is highly likely. This fact, associated with the much higher number of points in the columns compared with the rest of the scene, causes that the proportion of true positives that are lost is higher than the proportion of false positives. The values in the table show that the number of true positives approximately halves from 66689 to 33119, while the number of false positives is only reduced of one third, from 14020 to 8906. This results in a relatively small decrease in precision, below 0.04.

Still, the use of cross checking can be quite effective in scenes with strong cross voting, but the threshold should not be set too strict (high), otherwise very few points will remain in the reconstruction.

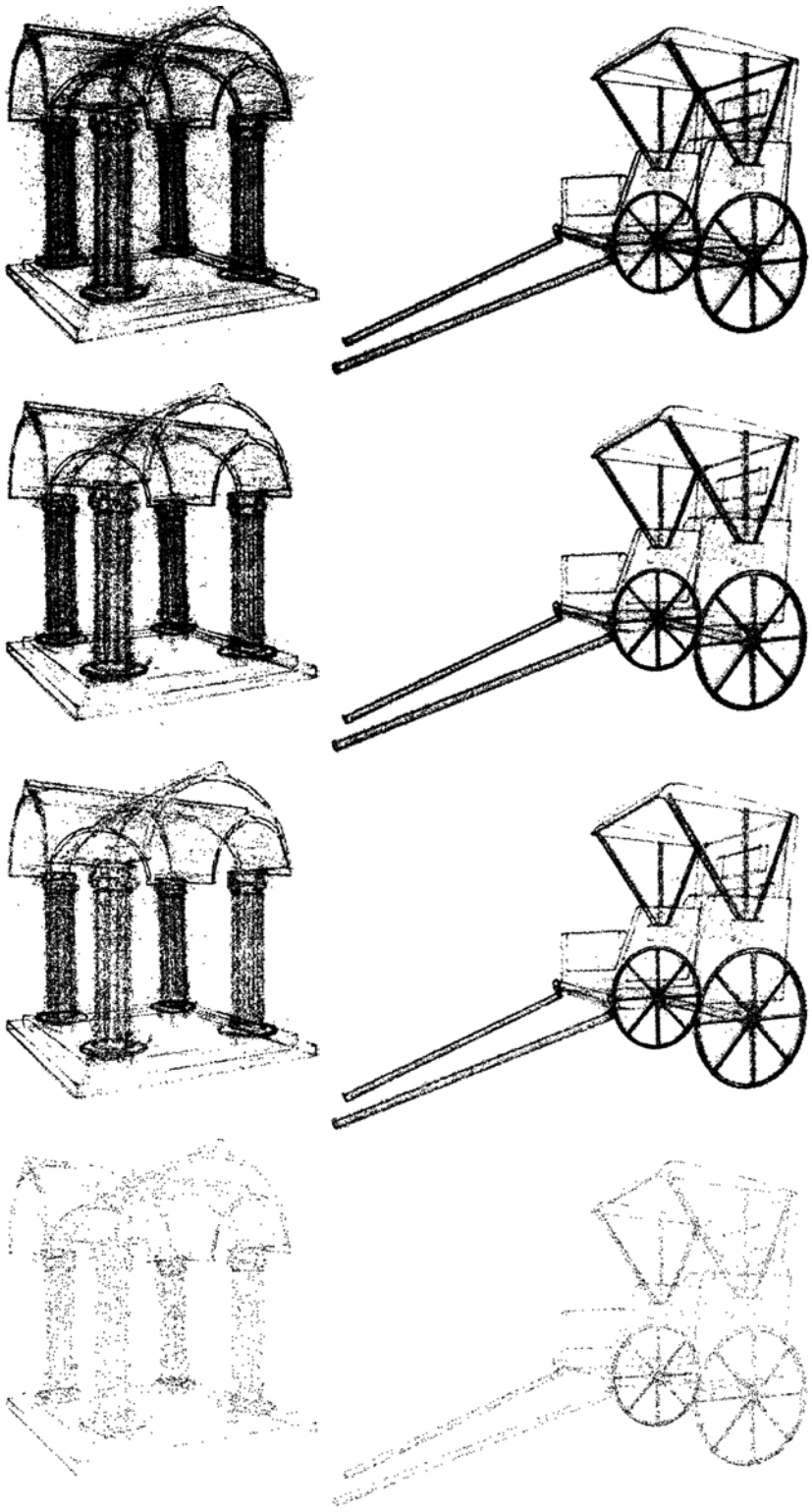
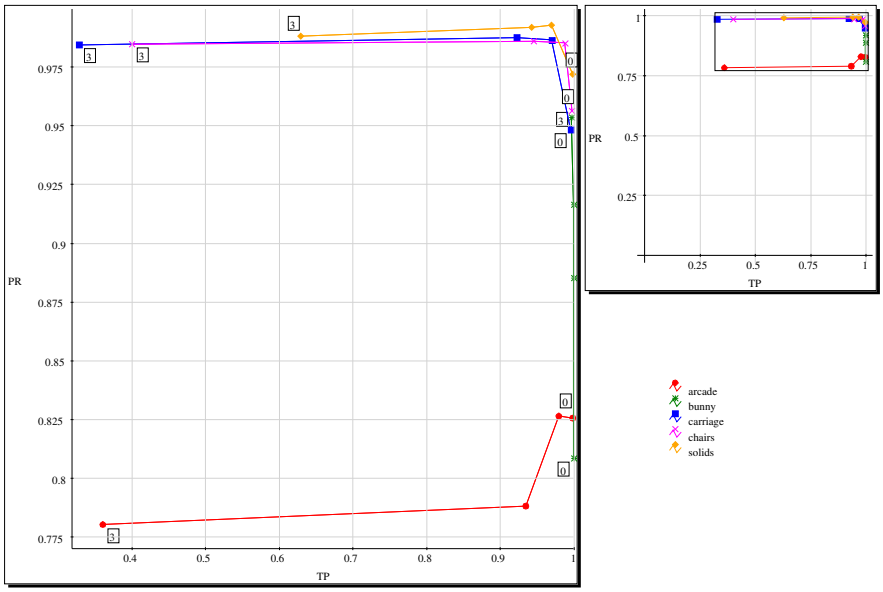


Figure 5.10: *Cross check threshold tests (reconstruction views).*
Reconstruction using cross check thresholds of 0, 1, 2, and 3 (from top to bottom)



arcade							
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR
0	**24357**	**115215**	*140*	*3052*	**139572**	**0.999**	0.825
1	14020	66689	1391	60664	80709	0.980	**0.826**
2	8906	33119	2297	98442	42025	0.935	0.788
3	*540*	*1916*	**3392**	**136916**	*2456*	*0.361*	*0.780*

bunny							
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR
0	**18298**	**77219**	*10*	*6035*	**95517**	**1.000**	*0.808*
1	8658	66856	*10*	26038	75514	**1.000**	0.885
2	5274	57792	11	38485	63066	**1.000**	0.916
3	*1113*	*22819*	**71**	**77559**	*23932*	*0.997*	**0.953**

carriage							
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR
0	**5605**	**102674**	*412*	*4942*	**108279**	**0.996**	*0.948*
1	1071	77221	2411	32930	78292	0.970	0.986
2	638	49960	4199	58836	50598	0.922	**0.987**
3	*52*	*3282*	**6695**	**103604**	*3334*	*0.329*	0.984

chairs							
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR
0	**2511**	**54920**	*147*	*2195*	**57431**	**0.997**	*0.956*
1	671	44287	546	14269	44958	0.988	0.985
2	393	27577	1590	30213	27970	0.945	**0.986**
3	*34*	*2170*	**3248**	**54321**	*2204*	*0.401*	0.985

solids							
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR
0	**399**	**13813**	*20*	*293*	**14212**	**0.999**	*0.972*
1	69	9294	288	4874	9363	0.970	**0.993**
2	60	7102	428	6935	7162	0.943	0.992
3	*18*	*1488*	**876**	**12143**	*1506*	*0.629*	0.988

Figure 5.11: *Cross check threshold tests (data chart).*

5.3.5 Similar camera angle

Camera similarity affects the number of cameras to be used when performing cross-checking (Section 3.4.1).

In the tests presented here, camera similarity is assessed based on the similarity of the viewing angles, according to the similarCameraAngle parameter. More complex measures for similarity can incorporate the comparison of the viewing frusta for both cameras, however this simple measure is more intuitive and hence more appropriate to analyze the results.

A point has to pass the cross-checking test in *all* similar cameras to avoid being deemed invalid. If the angle is set too wide, the number of cameras will be high and therefore the cross-checking will be too strict. This in principle leads to more confidence in the results, but a lower number of valid points in the end. The expected consequence is an increase in precision and decrease in recall.

Figure 5.12 shows the effects of varying this parameter, in the reconstructions of the chairs and carriage scenes. On the top, the angle is lower (10 degrees), meaning that there are less similar cameras, thus less constraints. As the angle raises, the constraints increase, and the point filtering begins to show. Incorrect point clouds that are present in the top images are filtered out in the lower ones. In the chairs scenes, the more visible clouds are on the seats and back, and around its borders and legs. In the carriage scene, they are on the wheels, the front of the rooftop and between the bars supporting the rooftop.

The chart in Figure 5.13 shows the expected trade-off between precision and number of points. Again, the arcade scene shows some minor fluctuations in precision (below 0.04), due to the very high density of contours in the columns.

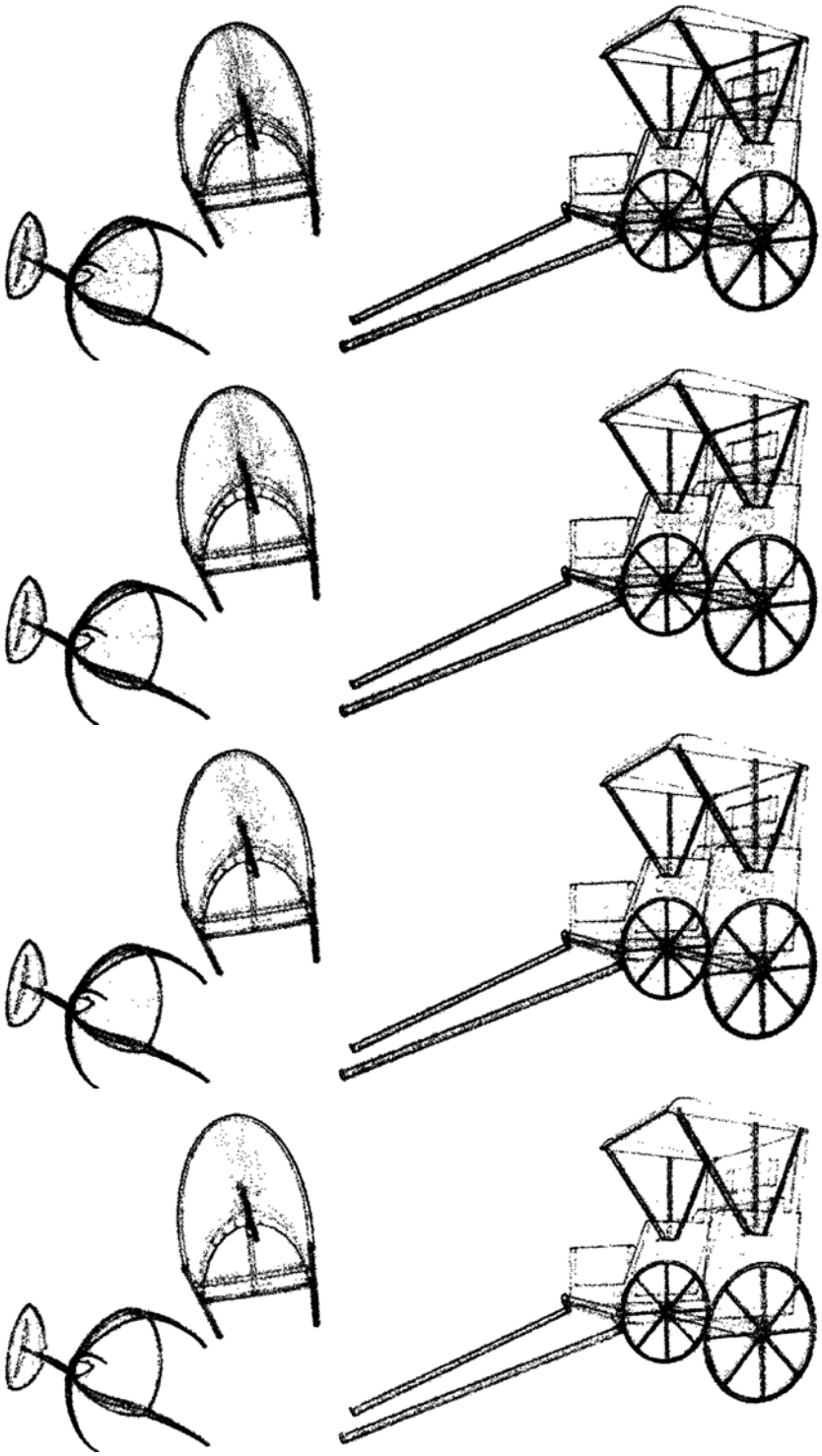
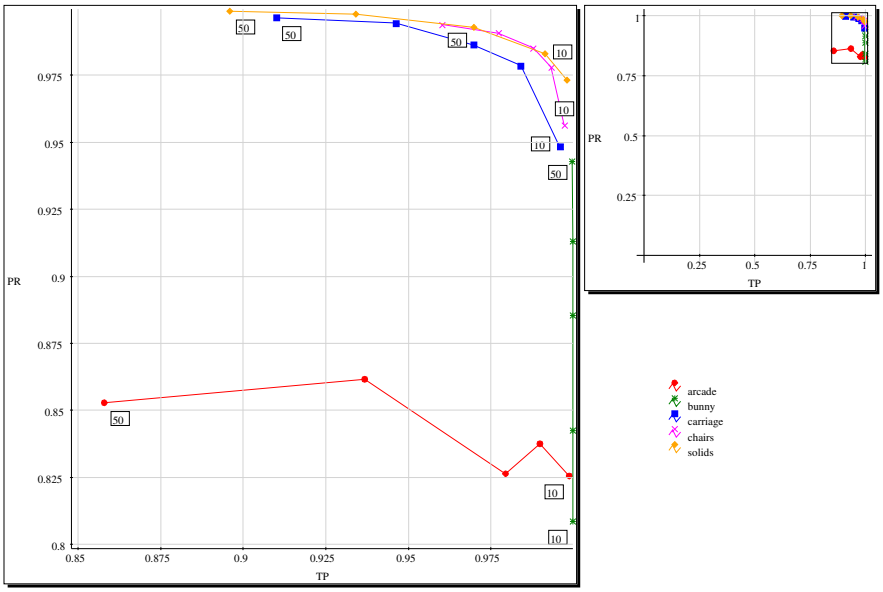


Figure 5.12: *Similar camera angle tests (reconstruction views).*
Reconstruction using similar camera angles of 10,20,30 and 40 degrees (from top to bottom)



arcade							
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR
10	**24357**	**115215**	*140*	*3052*	**139572**	**0.999**	*0.825*
20	17305	89146	903	35410	106451	0.990	0.837
30	14020	66689	1391	60664	80709	0.980	0.826
40	5195	32320	2179	103070	37515	0.937	**0.862**
50	*2785*	*16116*	**2667**	**121196**	*18901*	*0.858*	0.853

bunny							
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR
10	**18298**	**77219**	*10*	*6035*	**95517**	1.000	*0.808*
20	14063	75156	*10*	12333	89219	1.000	0.842
30	8658	66856	*10*	26038	75514	1.000	0.885
40	5336	56016	*10*	40200	61352	1.000	0.913
50	*2423*	*39958*	**17**	**59164**	*42381*	1.000	**0.943**

carriage							
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR
10	**5605**	**102674**	*412*	*4942*	**108279**	**0.996**	*0.948*
20	2010	91020	1471	19132	93030	0.984	0.978
30	1071	77221	2411	32930	78292	0.970	0.986
40	329	57636	3274	52394	57965	0.946	0.994
50	*150*	*41876*	**4140**	**67467**	*42026*	*0.910*	**0.996**

chairs							
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR
10	**2511**	**54920**	*147*	*2195*	**57431**	**0.997**	*0.956*
20	1121	49675	336	8641	50796	0.993	0.978
30	671	44287	546	14269	44958	0.988	0.985
40	348	36662	849	21914	37010	0.977	0.991
50	*183*	*29307*	**1214**	**29069**	*29490*	*0.960*	**0.994**

solids							
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR
10	**375**	**13565**	*27*	*558*	**13940**	**0.998**	*0.973*
20	209	11926	103	2287	12135	0.991	0.983
30	69	9294	288	4874	9363	0.970	0.993
40	14	6150	433	7928	6164	0.934	0.998
50	*5*	*4471*	**519**	**9530**	*4476*	*0.896*	**0.999**

Figure 5.13: *Similar camera angle tests (data chart).*

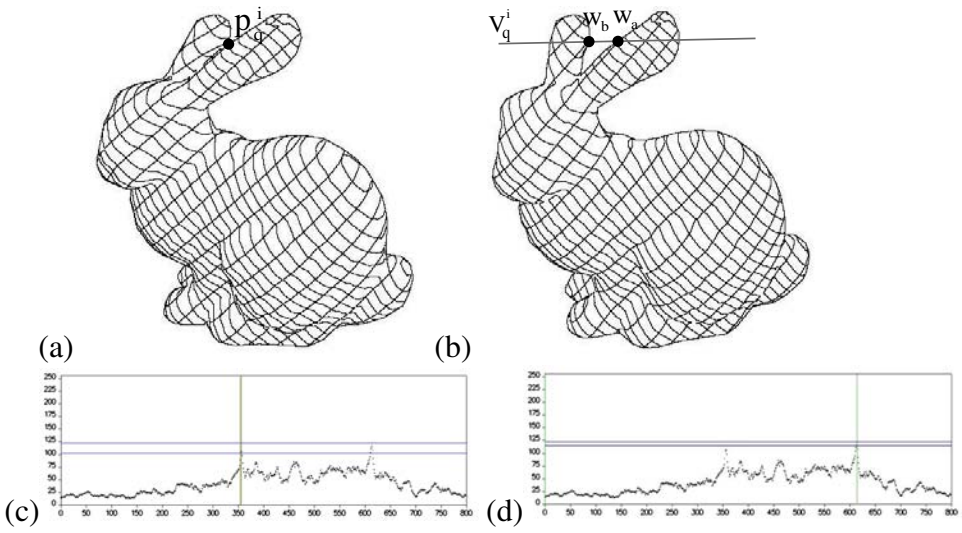


Figure 5.14: *Vote percent threshold example*.

(a) Reference view and selected point

(b) A view showing the view ray and the points at the ears (the point on the right, $\mathbf{w}_a$, is the correct one)

(c) Graph of the votes as a function of the candidates, using a value of 85%

(d) Same but using a value of 95%

5.3.6 Vote percent threshold

In some cases, two or more candidates for a given point may gather a similar number of votes. This happens for instance for points that are on opposite sides of a same object, or when an object occludes another. In some of these cases, due to various error sources, it may happen that the correct point closest to the reference camera gathers slightly less votes than one of the other high-voted candidates. The parameter votePercentThreshold was introduced to account for these situations (Section 3.3.5). Instead of considering only the global maximum of the candidates, the local maxima that have votes close to the global maximum are also considered. The votePercentThreshold parameter defines how close that margin is.

Figure 5.14 shows an example of this situation. It refers to the votes gathered for candidates from a view ray that crosses the bunny, intersecting it in both ears. Candidates on those two surfaces gather a high number of votes. However, the candidate on the back ear gathers some more votes, being the global maximum. If only the global maximum is considered, the point is either rejected if cross checking is in effect, or (incorrectly) accepted as valid. However, considering also the high-valued local maxima that may exist within a close range of the number of votes of the global maximum, the correct candidate may be included and accepted as the winner (assuming it passes the cross-checking).

This parameter should not be set too low, as it increases the probability of adding too many incorrect local maxima. Figure 5.15 shows the results of varying this parameter in the bunny and arcade scenes. Lower percentages lead to clouds of incorrect

points, more noticeable in the bunny's ears and paws, and in the interior and borders of the arcade. As the percentage is increased, outliers are filtered out, but some sparseness also arises (e.g. in the bunny's paws and the middle of the rightmost column of the arcade).

The precision/recall chart in Figure 5.16 shows that for the solids and the chairs scenes, there are only very small variations in precision and recall. This is justified by the reduced number of contours, that lead to a low number of cross-voting occurrences. In this situation, the points have in general strong global maxima, and therefore the variation of the percentage does not influence the results much.

However, in cases with higher contour density and contour alignment, such as the arcade, the carriage and the bunny scenes, the variations are more noticeable. In general, lowering the threshold below 85% results in a decrease of precision, compensated by an increase in recall. Increasing the threshold above 90% produced a decrease in precision in the bunny scene (of up to 0.027) and in the arcade scene (up to 0.036), and a very small variation of precision in the carriage scene (in the range of 0.002). In these last two cases there was also a decrease in recall.

As expected, values too close to 100% lead to a loss in precision in scenes where there is a high density of contours. This shows that there is a significant number of points where there are close votings between two or more candidates. Values below 80% also lead to a precision loss, since there are many incorrect candidates being considered, which may be accepted as valid in the cases of extreme contour density, as in the arcade scene. This suggests that values outside the range of 85% to 95% should be avoided.

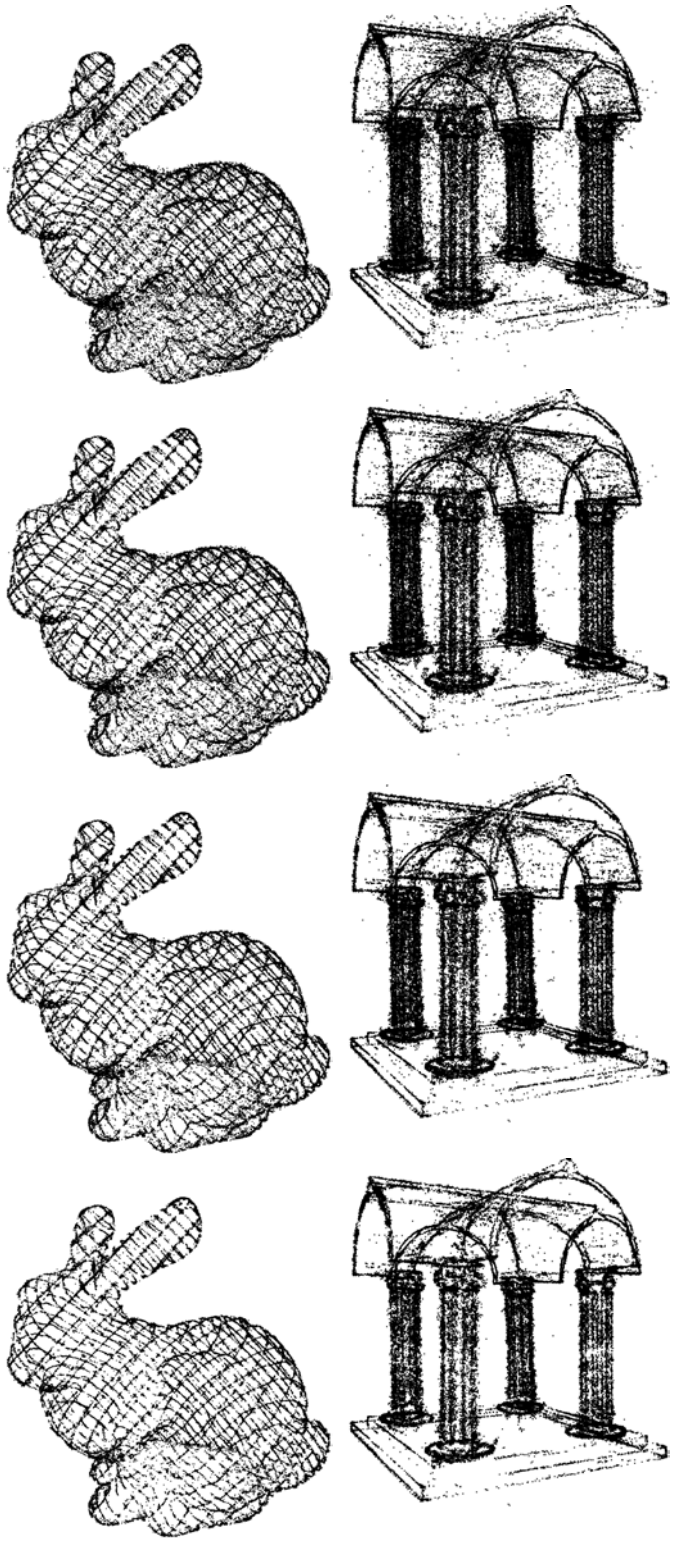
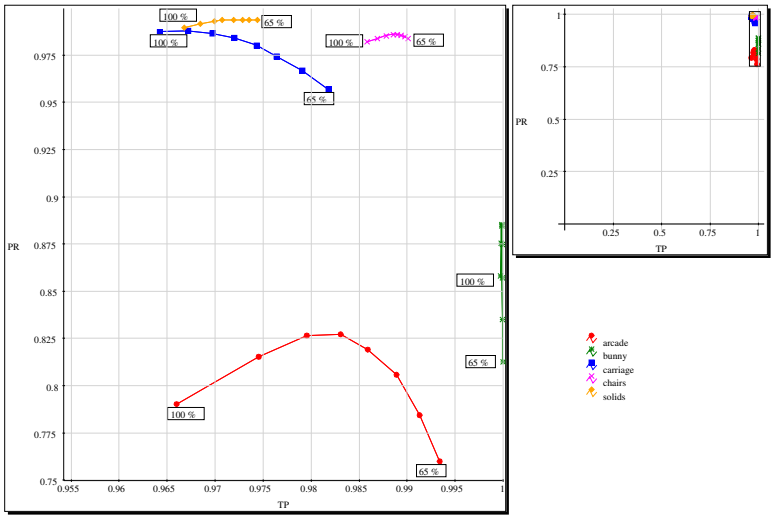


Figure 5.15: *Vote percent threshold tests (reconstruction views).*
Reconstruction using vote percentages of 70%, 80%, 90% and 100% (from top to bottom)



arcade							
errorPercentInterval	fp	tp	fn	tn	EP	TP	PR
65 %	**22493**	71136	*469*	*48666*	**93629**	**0.993**	*0.760*
70 %	19838	72065	631	50230	91903	0.991	0.784
75 %	17486	**72421**	812	52045	89907	0.989	0.806
80 %	15866	71792	1024	54082	87658	0.986	0.819
85 %	14667	70081	1205	56811	84748	0.983	**0.827**
90 %	14020	66689	1391	60664	80709	0.980	0.826
95 %	13827	60964	1589	66384	74791	0.975	0.815
100 %	*13820*	*52032*	**1830**	**75082**	*65852*	*0.966*	0.790

bunny							
errorPercentInterval	fp	tp	fn	tn	EP	TP	PR
65 %	**16025**	69539	2	*15996*	**85564**	1.000	*0.813*
70 %	13941	70484	4	17133	84425	1.000	0.835
75 %	11862	**71139**	5	18556	83001	1.000	0.857
80 %	10187	71025	6	20344	81212	1.000	0.875
85 %	9085	69697	9	22771	78782	1.000	**0.885**
90 %	*8658*	66856	10	26038	75514	1.000	**0.885**
95 %	8812	61910	12	30828	70722	1.000	0.875
100 %	9069	*54812*	**13**	**37668**	*63881*	1.000	0.858

carriage							
errorPercentInterval	fp	tp	fn	tn	EP	TP	PR
65 %	**3604**	79703	*1478*	*28848*	**83307**	**0.982**	*0.957*
70 %	2752	**79926**	1707	29248	82678	0.979	0.967
75 %	2118	79783	1925	29807	81901	0.976	0.974
80 %	1621	79324	2089	30599	80945	0.974	0.980
85 %	1286	78482	2260	31605	79768	0.972	0.984
90 %	1071	77221	2411	32930	78292	0.970	0.986
95 %	949	75642	2567	34475	76591	0.967	**0.988**
100 %	*947*	*73351*	**2721**	**36614**	*74298*	*0.964*	0.987

chairs							
errorPercentInterval	fp	tp	fn	tn	EP	TP	PR
65 %	764	**45967**	*460*	*12582*	**46731**	**0.990**	0.984
70 %	716	45833	473	12751	46549	**0.990**	0.985
75 %	681	45599	490	13003	46280	0.989	0.985
80 %	649	45284	504	13336	45933	0.989	**0.986**
85 %	*648*	44864	518	13743	45512	0.989	**0.986**
90 %	671	44287	546	14269	44958	0.988	0.985
95 %	722	43507	576	14968	44229	0.987	0.984
100 %	**777**	*42401*	**609**	**15986**	*43178*	*0.986*	*0.982*

solids							
errorPercentInterval	fp	tp	fn	tn	EP	TP	PR
65 %	65	**9686**	*254*	*4520*	**9751**	**0.974**	**0.993**
70 %	64	9649	262	4550	9713	**0.974**	**0.993**
75 %	64	9592	267	4602	9656	0.973	**0.993**
80 %	64	9528	275	4658	9592	0.972	**0.993**
85 %	*63*	9425	284	4753	9488	0.971	**0.993**
90 %	69	9294	288	4874	9363	0.970	**0.993**
95 %	80	9121	297	5027	9201	0.968	0.991
100 %	**97**	*8889*	**305**	**5234**	*8986*	*0.967*	*0.989*

Figure 5.16: *Vote percent threshold tests (data chart).*

5.3.7 Number of candidates

A discussion of the choice of the candidate set was presented in Section 3.3.2. The cardinality of this set has implications on the final result as can be seen in Figure 5.17. The smaller set (Figure 5.17(a)) provides less definition, appearing slightly blurred when compared to the sets in figures (b) (c) and (d). With more than 600 candidates the differences become too subtle to judge.

This is to be expected since more candidates imply more resolution in the reconstruction. Hence for a correctly reconstructed candidate one may expect lower errors. Figure 5.19 shows the error distribution for the reconstructed points with errors up to 1% when considering different cardinality for the sets of candidates. It can be seen that the error distribution curves get steeper as the number of candidates increases, i.e. in average the error of these reconstructed points decreases.

Therefore, in principle increasing the number of candidates should lead to the rejection of fewer points by cross-checking. As the resolution increases, and the reconstruction errors get smaller, the candidates get more accurately reconstructed, and therefore are more likely to pass the cross-checking tests.

However increasing the number of candidates may have also a downside in some situations, as it may increase the number of plateaus. Consider a candidate x that is identified as a local maximum in a small set of candidates. If the set of candidates is e.g. doubled, the new neighbors of x may have higher, lower or equal voting as x. In the first two cases, what happens is that the local maximum is shifted accordingly (or kept on x, if both neighborhoods are lower). But if the voting is equal (and this effect is extended to a significant neighborhood), then the local maxima becomes a plateau and the point will be rejected. As mentioned in Section 3.3.4 this plateau effect occurs more often when there is an alignment of contour lines with epipolars, due to camera arrangement.

The tables in Figure 5.18 shows that, in the general case, the number of points that are rejected for failing the cross-check test (in the columns labelled 'RB_CC') decrease as the number of candidates increases. It is also clear that, as the number of candidates per point increases, so does the number of rejected points due to the formation of plateaus (or the absence of local maxima), represented in the columns labelled 'RB_PLATEAU'. In the bunny scene there is a significant decrease in the number of valid points, with repercussions on the number of true and false positives. The same decrease may be observed in some settings with other scenes but to a much lesser extent.

The bunny scene registers the lowest decrease in the number of points rejected by cross-checking, and one of the strongest increases of points rejected by plateaus. Figure 5.20 shows that the points rejected by cross-checking in the bunny scene are in general located either on apparent contours, therefore not meant to be reconstructed, or in recondite areas of the model, hence not seen even from similar cameras with the test setup. In both cases, increasing the number of candidates does not help to solve the problem.

Regarding the increase in the number of points rejected by plateaus, the effect is

stronger in the bunny scene due to a strong degree of alignment of contours with the displacement in some cameras (relative to the reference camera). The same problem can be visualized in the arcade scene, where rejections by plateaus occur when increasing the number of candidates from 200 to 800. These new rejections are mostly located at an edge in the lower right corner of the image. Looking at the points rejected by plateaus in the carriage scene also reveals another edge that has accumulated a significant number of plateaus. The images show that the points rejected by plateaus for the bunny scene are placed on contours that have roughly the same alignment than the edges in the carriage and arcade scenes where rejections by plateaus appeared due to the increase on the number of candidates.

As a conclusion, sampling further candidates does improve the error of the reconstructed points, but represents a compromise between rejection by cross-checking and rejection by plateaus. The number of rejections due to cross-checking tends to decrease as a consequence, due to the more accurate 3D reconstruction, and therefore more accurate reprojection on similar camera images. The potential downside is an increase on the number of plateaus, and as a consequence the elimination of local maxima that we're present when considering fewer candidates

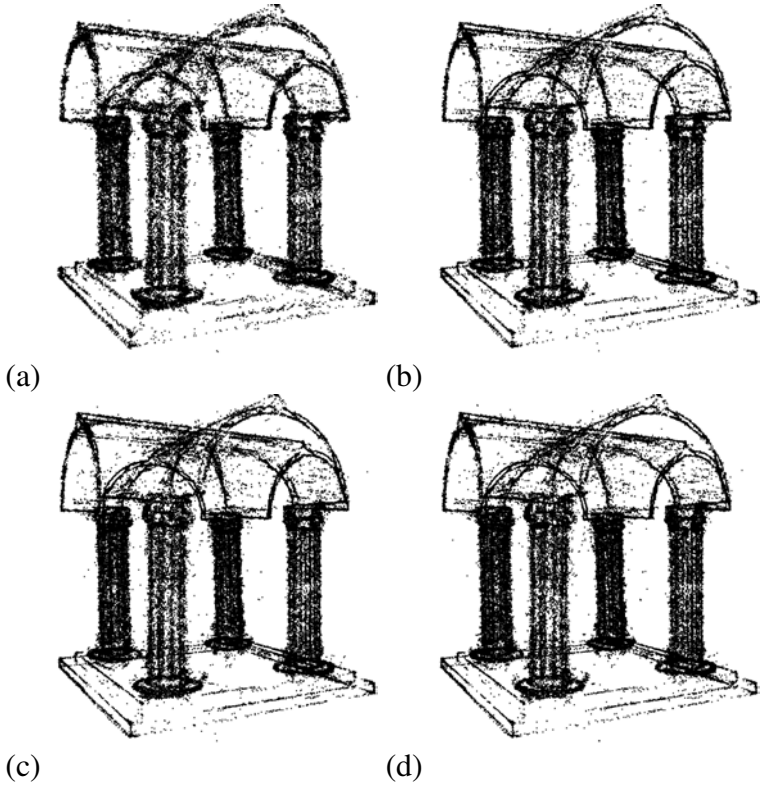
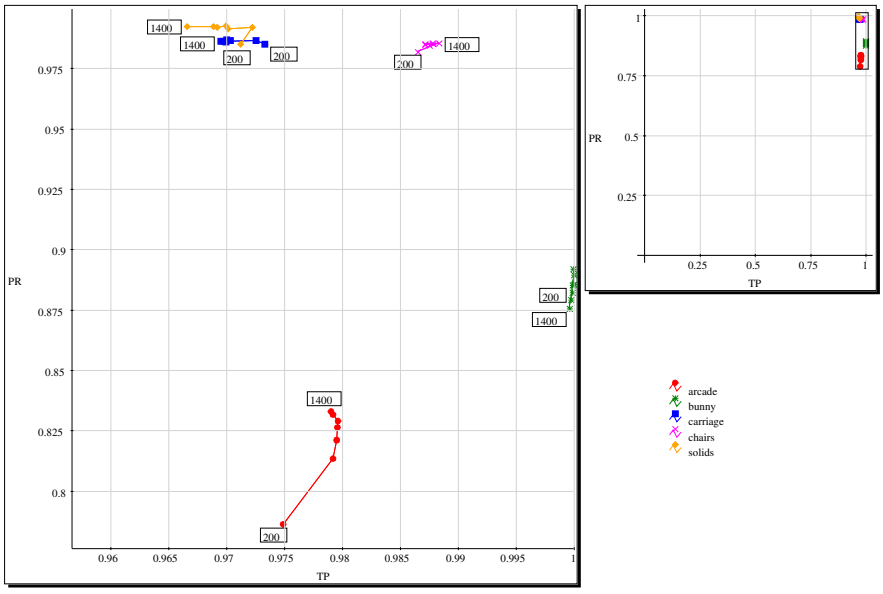


Figure 5.17: *Number of candidates tests (reconstruction views).*
Reconstruction using different numbers of candidates, namely 200 (a), 600 (b), 1000 (c) and 1400 (d)



arcade									
lineWindowSize	fp	tp	fn	tn	EP	TP	PR	RB_PLATEAU	RB_CC
200	**14628**	*53787*	1384	**72965**	*68415*	*0.975*	*0.786*	3866	**70483**
400	14288	62232	*1322*	64922	76520	0.979	0.813	*2980*	63264
600	14208	65168	1363	62025	79376	**0.980**	0.821	3044	60344
800	14020	66689	1391	60664	80709	**0.980**	0.826	3192	58863
1000	13937	67486	1404	59937	81423	**0.980**	0.829	3576	57765
1200	13767	67931	1444	59622	81698	0.979	0.831	4045	57021
1400	*13695*	**68227**	**1461**	*59381*	**81922**	0.979	**0.833**	**4509**	*56333*

bunny									
lineWindowSize	fp	tp	fn	tn	EP	TP	PR	RB_PLATEAU	RB_CC
200	**9083**	70227	2	*22250*	**79310**	1.000	0.885	*1347*	**20905**
400	*8525*	**70234**	3	22800	78759	1.000	**0.892**	2726	20077
600	8535	68600	*0*	24427	77135	1.000	0.889	4403	20024
800	8658	66856	10	26038	75514	1.000	0.885	6045	20003
1000	8748	65504	9	27301	74252	1.000	0.882	7446	*19864*
1200	8816	64180	17	28549	72996	1.000	0.879	8643	19923
1400	8959	*63007*	**24**	**29572**	*71966*	1.000	*0.876*	**9691**	19905

carriage									
lineWindowSize	fp	tp	fn	tn	EP	TP	PR	RB_PLATEAU	RB_CC
200	**1137**	*75611*	*2076*	**34809**	*76748*	**0.973**	*0.985*	*1462*	**35423**
400	1056	**77952**	2203	*32422*	**79008**	**0.973**	**0.987**	3119	31506
600	1087	77404	2401	32741	78491	0.970	0.986	4416	30726
800	1071	77221	2411	32930	78292	0.970	0.986	5354	29987
1000	1045	77079	2360	33149	78124	0.970	**0.987**	6283	29226
1200	*1030*	76976	2380	33247	78006	0.970	**0.987**	6918	28709
1400	1053	76759	**2418**	33403	77812	*0.969*	0.986	**7370**	*28451*

chairs									
lineWindowSize	fp	tp	fn	tn	EP	TP	PR	RB_PLATEAU	RB_CC
200	**785**	*42564*	**582**	**15842**	*43349*	*0.987*	*0.982*	*716*	**15708**
400	*650*	44212	*522*	14389	44862	**0.988**	**0.986**	1450	13461
600	699	44141	557	14376	44840	**0.988**	0.984	1981	12952
800	671	44287	546	14269	44958	**0.988**	0.985	2342	12473
1000	674	44360	579	14160	45034	*0.987*	0.985	2653	12086
1200	680	**44367**	579	*14147*	**45047**	*0.987*	0.985	2967	11759
1400	652	44344	550	14227	44996	**0.988**	**0.986**	**3205**	*11572*

solids									
lineWindowSize	fp	tp	fn	tn	EP	TP	PR	RB_PLATEAU	RB_CC
200	**127**	*8306*	*246*	**5846**	*8433*	0.971	*0.985*	*82*	**6010**
400	76	9281	265	4903	9357	**0.972**	0.992	110	5058
600	80	9293	286	*4866*	**9373**	0.970	0.991	219	4933
800	*69*	**9294**	288	4874	9363	0.970	**0.993**	313	4849
1000	74	9226	293	4932	9300	0.969	0.992	420	4805
1200	71	9188	295	4971	9259	0.969	0.992	521	4745
1400	71	9095	**314**	5045	9166	*0.967*	0.992	**632**	*4727*

Figure 5.18: *Number of candidates tests (data chart).*

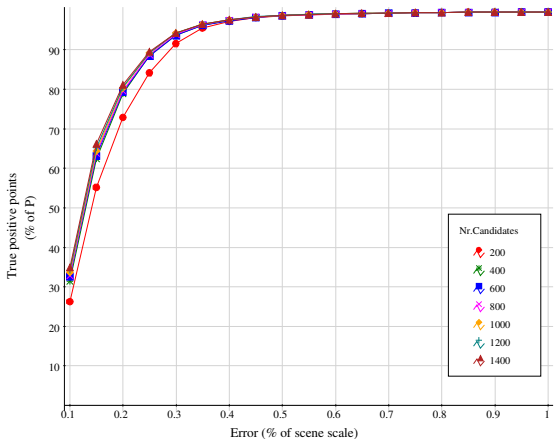


Figure 5.19: *Error distributions for different error thresholds*.
Each point in the chart indicates the percentage of valid points that have a real error below the corresponding error threshold. These results are shown for different candidate sets for the same scene (carriage)

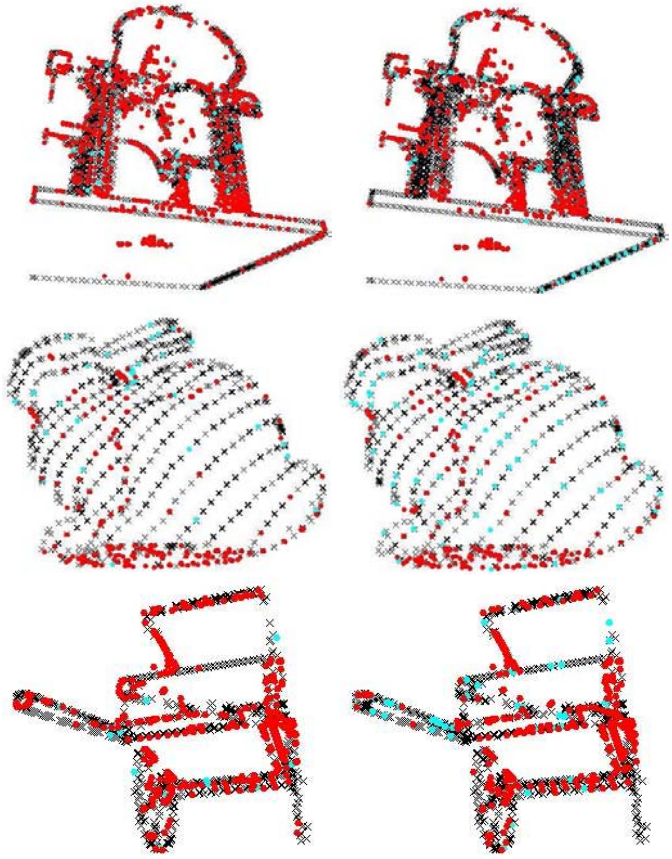


Figure 5.20: *Distribution of point rejection* .
Images on the left show the points rejected by plateaus (cyan) and cross-checking (red) over the points accepted as valid (black) for a set of 200 candidates, for a single reference image. The images on the right show the same for a set of 800 candidates.

5.3.8 Merging

Merging establishes correspondences between points reconstructed from different reference images (Section 3.4.2). In this process, points that are not merged with any other point are rejected. Thus, the final point set is simplified both by the merging of points and the rejection of non-merged points.

The rejection of non-merged points is grounded on the assumption that a correct point is likely to be reconstructed from various reference images, and those reconstructions should overlap or be very close in world space. Therefore, if a reconstructed point is isolated, the probability of being incorrectly reconstructed is high. The rejection of non-merged point adds an additional layer of robustness to the reconstruction.

Merging is governed by two parameters: merge3DThreshold and merge2DThreshold. Tests were performed to each of the parameters individually.

Computation of geometric error for merged points

Merged points are the result of multiple reconstructed points and are not associated to a single reference image. This raises an issue in the computation of the geometric error as described in Section 5.2.1. In order to compare the results of point merging with the results of the core reconstruction algorithm, each merged point $\mathbf{m}$ corresponding to the centroid of a set of n r-points $\mathbf{r}_r$ – *source points* – is considered as a set of n valid points $\mathbf{r}_{m_i}$ $(i = 1..n)$ with the same coordinates as $\mathbf{m}$.

As referred to in Section 5.2.1, care should be taken in determining which w-point $\mathbf{w}$ is to be compared with a reconstructed point. In the context of the core reconstruction algorithm, the real world point was found based on the reference camera from which the point was originally reconstructed. This reduces the chance of considering points that are close to a surface that is distant from the correct surface (recall the bunny's ears example, Figure 5.1).

In the case of a merged point, there are n reference cameras involved. This allows a better choice of the real world point to associate to the merged point, comparing to the use of a small neighborhood in a single reference image (as used in Section 5.2.1). By using the different n views, the problem of offsets at occluding contours is decreased, as the probability of not including the correct surface in the n neighborhoods also decreases. The advantages of relating the choice to cameras from where the point was reconstructed – i.e. reducing the probability of using points that are not on the correct surface, by choosing the surfaces that are closest to the reference cameras – are still retained.

Having a set of at most n possible w-points that $\mathbf{m}$ is meant to represent, a choice has to be made, in order to compute the reconstruction error. An intuitive solution would be to compute the individual distances between the merged point and each of the n possible world points and average those distances to obtain the error. However, such an error value would not reflect the distance at which the merged point is from the closest object of the scene, considering the reference cameras. Furthermore, it would

be even possible that there is no world point at the average distance, which would not be in accordance with the idea of the error corresponding to a distance to an existing surface. Instead, it is proposed that the error of a merged point corresponds to its distance to the closest real world point that is associated to the n source points.

If $\mathbf{m}_m$ is the result of merging n reconstructed points $\mathbf{r}^1_{p_1}$, $\mathbf{r}^2_{p_2}$, ..., $\mathbf{r}^n_{p_n}$ coming from the respective n reference cameras $\mathbf{C}^1..\mathbf{C}^n$, the reconstruction error of $\mathbf{m}_m$ is defined based on the real world points $\hat{\mathbf{w}}^i_{p_i}$ associated to the individual reconstructed points $\mathbf{r}^i_{p_i}$, as presented in Definition 5.3:

Definition 5.3 *Reconstruction error of a merged point*

$$e_m(\mathbf{m}_m) = \min_{i=1..n} ||\mathbf{m}_m - \hat{\mathbf{w}}^i_{p_i}||.$$

□

3D merge threshold

The 3D Merge Threshold parameter is the maximum 3D distance between two points for them to be considered for merging, defined as a percentage of the scene's reference dimension (Section 3.4.2).

The topmost row of Figure 5.21 shows the reconstruction of the bunny and carriage scenes using a low threshold of 0.05% of the scenes' dimensions. As referred to in Section 3.4.2, using a too strict (low) value of the merging threshold leads to excessive rejection of non-merged points. This justifies the sparseness found in that first row.

With higher values of the threshold, more points are merged together, resulting in sparseness by excessive merging. This effect can be seen on the bottom row of Figure 5.21 (corresponding to a threshold value of 1%), in places such as the foremost ear of the bunny and parts of the wheels of the carriage.

These effects are also reflected in the precision/recall data. In Figure 5.22, there are high increases in precision when reducing the threshold from 1% to 0.5% or to 0.1%, but further reductions on the threshold result in very little gain in precision, with a high loss in recall. The column labeled TM indicates the total number of valid merged points (a point from TM represents the merging of multiple points of EP). This column further evidences the balance between excessive rejection and excessive merging.

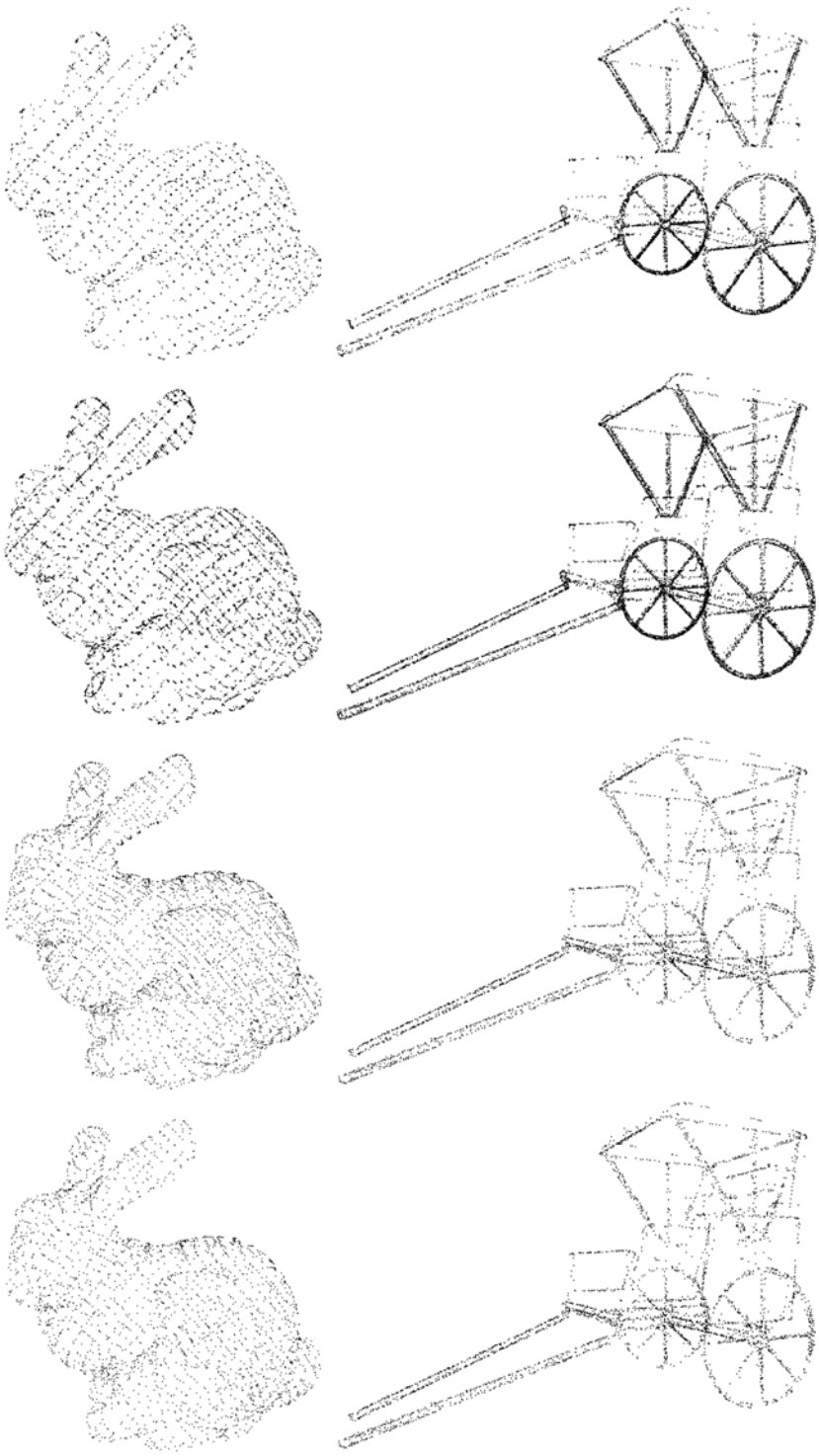
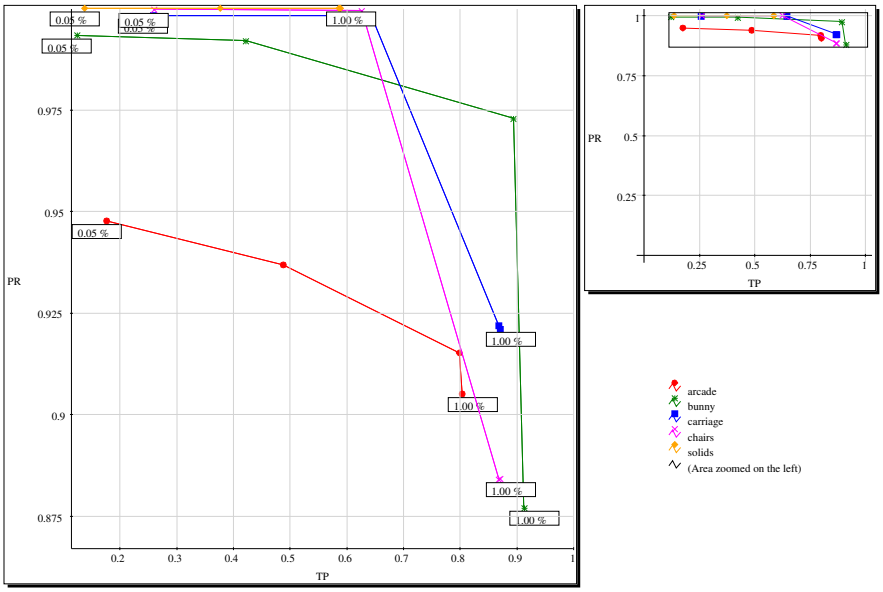


Figure 5.21: *Merging threshold 3D tests (reconstruction views).*
Results of merging using a 3D threshold of 0.05%, 0.1%, 0.5% and 1% of the scenes' dimensions (from top to bottom)



arcade								
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
0.05 %	*669*	*12107*	**56208**	**73780**	*12776*	*0.177*	**0.948**	*5468*
0.10 %	2278	33773	35401	71312	36051	0.488	0.937	**10442**
0.50 %	5290	**57005**	14294	66175	62295	0.800	0.915	8098
1.00 %	**5979**	57001	*13893*	*65891*	**62980**	**0.804**	*0.905*	7894

bunny								
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
0.05 %	*8315*	*8315*	**58655**	**34536**	*8371*	*0.124*	**0.993**	3675
0.10 %	231	28516	39016	33799	28747	0.422	0.992	**8033**
0.50 %	1760	**63002**	7441	29359	64762	0.894	0.973	4270
1.00 %	**8272**	59008	*5618*	*28664*	**67280**	**0.913**	*0.877*	*3004*

carriage								
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
0.05 %	*39*	*20625*	**59036**	**33933**	*20664*	*0.259*	**0.998**	8345
0.10 %	93	51367	28376	33797	51460	0.644	**0.998**	**10983**
0.50 %	5494	64841	9806	33492	70335	0.869	0.922	3787
1.00 %	**5566**	**64956**	*9655*	*33456*	**70522**	**0.871**	*0.921*	*3721*

chairs								
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
0.05 %	*4*	*11674*	**33183**	**14912**	*11678*	*0.260*	**1.000**	4693
0.10 %	18	28169	16776	14810	28187	0.627	0.999	**6340**
0.50 %	4623	35259	5331	14560	39882	0.869	*0.884*	2331
1.00 %	**4633**	**35341**	*5249*	*14550*	**39974**	**0.871**	*0.884*	*2273*

solids								
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
0.05 %	0	*1327*	**8255**	**4943**	*1327*	*0.138*	1.000	*557*
0.10 %	0	3614	5973	4938	3614	0.377	1.000	**1120**
0.50 %	0	5623	3969	*4933*	5623	0.586	1.000	1082
1.00 %	0	**5652**	*3940*	*4933*	**5652**	**0.589**	1.000	1069

Figure 5.22: *Merging threshold 3D tests (data chart).*

2D merge threshold

The 2D Merge Threshold parameter is the maximum 2D distance in pixels between two points' projections for them to be considered for merging (Section 3.4.2). For two points to be merged both their 3D distance and their projections' 2D distance must be below the corresponding thresholds.

While the 3D threshold parameter can be seen as a global parameter, the 2D merge threshold is more of a local parameter, useful e.g. when there are cameras closer to some objects of the scene. Two points that are close in 3D (according to the value of the 3D merge threshold) may be quite distant in 2D, in images from one or more *similar cameras* close to objects. One can assume that images taken at close range imply a desire for additional detail. This means that points that have a relatively large distance at that range should be considered as independent points, and not two reconstructions of the same point. Therefore, in these cases, such points should not be merged.

Figure 5.23 shows the reconstructed points for the arcade and chairs scenes using different values for the 2D merging threshold. The first row represents the more conservative value of one pixel, and the bottom the more liberal value of three pixels.

The bottom row exhibits sparseness caused by excessive merging, typical of liberal values for merging. Between the two top rows, the differences are not very significant. Still, in the arcade scene, it is noticeable that e.g. the back columns are denser on the 2-pixel version than on the 1-pixel. The 2-pixel version of the chairs scene also exhibits a bit more scatter and outliers, namely on the legs of the chair on the left and on the back of the chair on the right, than the 1-pixel version.

The precision-recall data presented in Figure 5.24 show the effects of varying the 2D merging threshold. As it can be seen, for values of the threshold below 3, there is a gain in precision on all scenes, when compared with the values before merging. In particular, the scenes that have higher contour density and therefore more chances of cross-voting situations – the arcade and the bunny – present the highest gains in precision. The additional precision has the cost of decreasing recall, especially if the more conservative value of one pixel is used. The number of valid merged points TM is in general higher for the value of 1, as less points get merged into one and hence, merged points result of more, smaller groups of points. This finer granularity leads then to the better precision. The exception in this case is the arcade scene, as the original reconstructed points are more scattered and some excessive rejection occurs at this low threshold of 1.

Merging can therefore be a valuable alternative when precision is more important than the number of points (compared to the results obtained without merging), and is particularly effective in cases of high contour density.

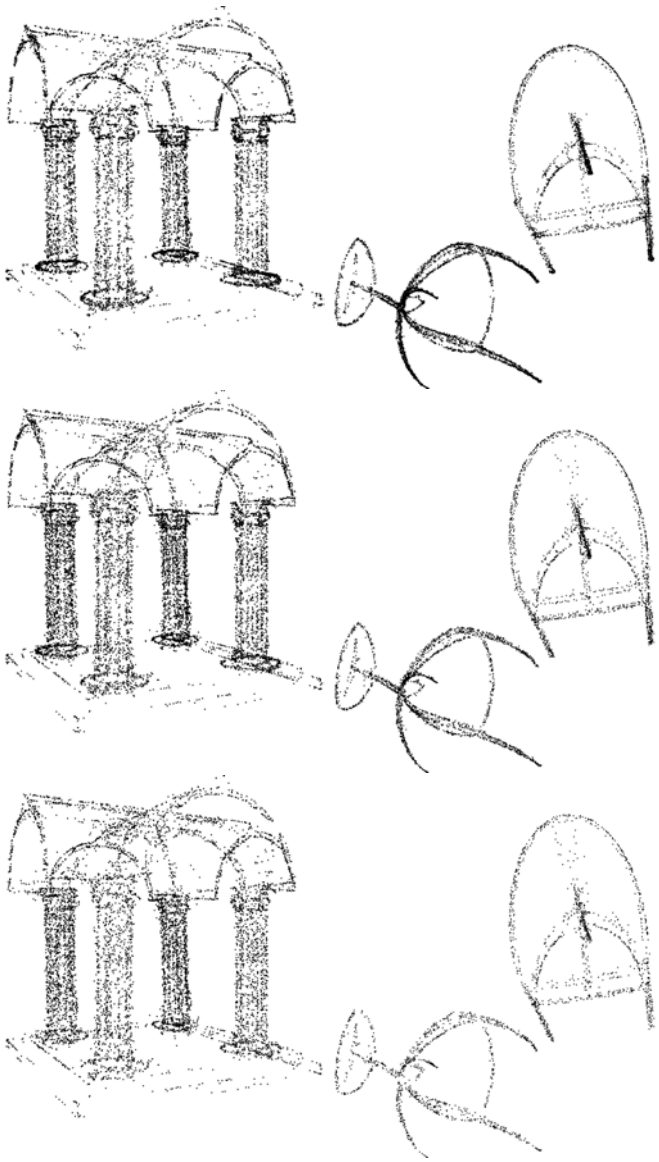
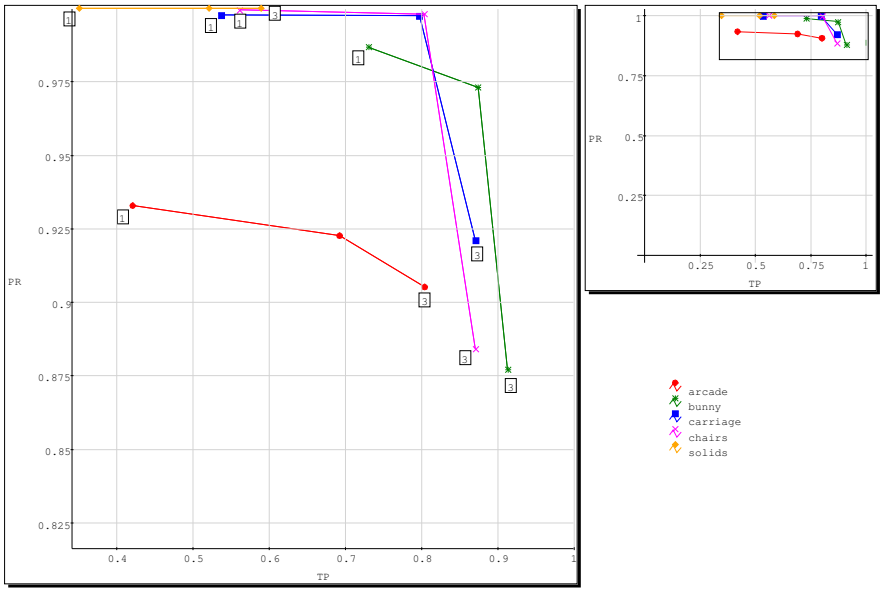


Figure 5.23: *Merging threshold 2D tests (reconstruction views).*
Results of merging using a 2D threshold of 1, 2 and 3 pixels (from top to bottom)



arcade								
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
1	*2099*	*29090*	**40028**	**71547**	*31189*	*0.421*	**0.933**	**10706**
2	4105	48875	21705	68079	52980	0.692	0.923	10473
3	**5979**	**57001**	*13893*	*65891*	**62980**	**0.804**	*0.905*	*7894*

bunny								
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
1	*680*	*50660*	**18712**	**31510**	*51340*	*0.730*	**0.987**	**7263**
2	1706	**61465**	8844	29547	63171	0.874	0.973	4474
3	**8272**	59008	*5618*	*28664*	**67280**	**0.913**	*0.877*	*3004*

carriage								
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
1	*102*	*42843*	**36908**	**33780**	*42945*	*0.537*	**0.998**	**12475**
2	172	63695	16196	33570	63867	0.797	0.997	7208
3	**5566**	**64956**	*9655*	*33456*	**70522**	**0.871**	*0.921*	*3721*

chairs								
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
1	*18*	*25235*	**19718**	**14802**	*25253*	*0.561*	**0.999**	**7000**
2	75	**36189**	8858	14651	36264	0.803	0.998	4193
3	**4633**	35341	*5249*	*14550*	**39974**	**0.871**	*0.884*	*2273*

solids								
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TM
1	0	*3364*	**6224**	**4937**	*3364*	*0.351*	1.000	*1062*
2	0	4994	4595	4936	4994	0.521	1.000	**1173**
3	0	**5652**	*3940*	*4933*	**5652**	**0.589**	1.000	1069

Figure 5.24: *Merging threshold 2D tests (data chart).*

5.4 Effects of camera errors on synthetic scenes

In real situations, it is common to have errors in the calibration of the extrinsic parameters of the cameras. Such errors lead to problems in the reconstruction, as geometric principles such as the epipolar constraint cannot be directly applied, as referred in Section 2.1.4. Therefore, camera errors should be accounted for in some way. The method presented in this thesis accomplishes that by providing a controlled tolerance to offsets due to the use of distance-based vote maps, and integrating multiple images to gather more robust information that allows to cope with those and other types of errors. The tests presented in this section were aimed at testing the robustness of the reconstruction algorithm to different levels of camera calibration errors.

There are no widely accepted values for what can be a reasonable error in camera calibration, due to the multitude of camera calibration variants, the scenarios in which they are applicable and the quality of the inputs. Average reprojection errors for calibration techniques based on Tsai's [Tsai 87] methods and others [Bart 03, Roth 03, Roth 02, Mart 03, Tayl 95] are often reported in terms of to be in the order of sub-pixel up to 1 % or 2 % of the images' dimensions.

Therefore, to analyze the sensitivity of the algorithm to extrinsic camera calibration errors, different levels of noise were introduced in the cameras parameters. Each level represents an average offset of contours in the order of 0.5%, 1% and 2% of the images' dimensions (about 2.5, 5 and 10 pixels, respectively).

Figure 5.25 shows images of the contours, overlapped over the objects as viewed from cameras with errors in their parameters, for the two higher error values. This gives an idea of the type of offsets used in the tests.

The reconstructions of the scenes using those incorrectly calibrated cameras are shown in Figure 5.26, and keeping the basis set of parameters specified in Section 5.2.3. Figure 5.27 shows the precision/recall chart of the variations in camera errors, for each scene. As expected, there is a general decrease in precision and recall, as the errors increase.

Some of the most significant tests without errors in camera calibration, presented in the previous sections, were repeated with the average error of 1% to assess the sensitivity of some parameters to noise situations. The parameters tested were vote range, cross check threshold, similar camera angle and 3D and 2D merge thresholds.

The results are presented next, with each table augmented with the recall and precision values of the corresponding tests without camera errors (marked with an asterisk), and the corresponding losses (ΔTP and ΔPR).

In the vote range test (Figure 5.28), for each given scene, the losses in recall and precision are very similar, when comparing the vote ranges of 3 and 10 (e.g. in the carriage scene, there is a drop in precision of ΔPR=0.032 for the vote range of 10, and the corresponding loss for a vote range of 3 is of ΔPR=0.033). This suggests that the conclusions drawn in Section 5.3.3 still hold i.e., in scenes that contain a significant number of contours that are close to each other, the range of the vote maps should not by set too high.

It is also noticeable that some scenes were in general more sensitive to camera errors than others. There is a more evident loss in recall in the solids scene (over 0.12) resulting from a significant number of points being rejected in the case of camera errors. The most likely reason for this is the conjunction of the significant number of straight and parallel lines and the lower number of points: in the case of a camera error whose offset is normal to a significant number of parallel contour lines, most points of those lines will be poorly reconstructed. This effect can be seen in the last row, center image of Figure 5.26, where it is noticeable that the higher point losses are at the lines, while the corners of the solids present a significant point density. The arcade scene suffered the most significant losses in precision, in the order of 0.08, due to the cluttering in the columns area, that produces even stronger cross-voting when the contours are offset due to camera errors. These relative loss tendencies are scene-dependent, and therefore hold for the remaining tests.

The cross-check threshold test (Figure 5.29) shows that the higher losses in recall are associated to the more conservative cross-check threshold value tested here (2), topping at ΔTP=0.298 for the solids scene. The overly conservative values lead not only to the rejection of points previously labeled as false positives, but also to a rejection of a high number of points previously labeled as true positives. This was expected, due to the offsets caused by the camera errors, which cause cross-checking to fail in a high number of points when such a conservative value is used. On the other hand, precision losses are in general higher for more liberal values (e.g. 0) as the number of false positives can be significantly high in those cases. The arcade was again an exception in this case, for the same reasons that had already been observed in Section 5.3.4.

The similar camera angle test (Figure 5.30) presents tendencies similar to those observed with the cross-check threshold tests: more conservative values (higher angle) lead to higher losses in recall, and less conservative values lead to higher losses in precision (with the exception of the arcade scene, which is globally worse in terms of precision).

Finally, the merging tests (Figure 5.31 and Figure 5.32) show the lower losses in recall corresponding to the more liberal values of the parameters – 1% for the 3D threshold and 3 pixels for the 2D threshold. A similar effect occurs in terms of precision, but with two noteworthy singularities. First, in some cases, the loss is negative i.e. the precision actually increased. However, the number of positives has significantly decreased, when compared with the corresponding tests without camera errors (Section 5.3.8). Second, the variation in precision for the solids scene is 0, as the number of false positives was already zero in the exact camera calibration setting, implying the maximum precision. Also in this case, the number of positives was significantly reduced. In terms of the number of valid merged points, the tendencies were also similar to the ones observed in the case of exact camera calibration, with the solids scene suffering more loss with conservative values than other scenes, this time also in the case of the merge3Dthreshold. The arcade scene also was slightly more sensitive to excessive rejection caused by the merge3Dthreshold.

As a conclusion, the introduction of camera errors carried in general a natural but graceful reduction in precision and recall, as well as in the number of positives (points actually reconstructed). The behavior of the algorithm and the sensitivity of the pa-

rameters was similar to the case of exact camera calibration, providing the graceful degradation sought.

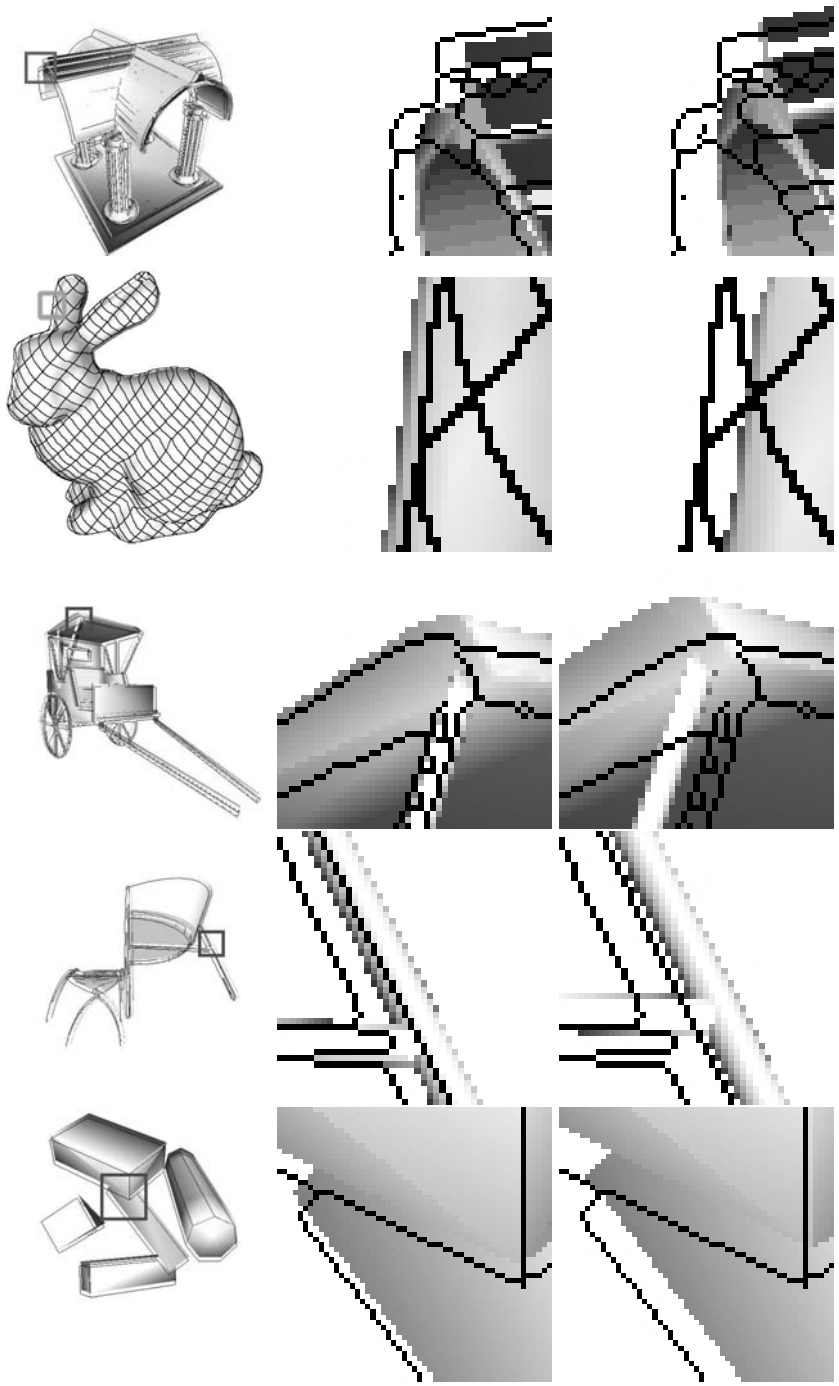


Figure 5.25: *Examples of the offsets introduced by the camera errors.*

Left: Selected area highlighted in the scene
Center: View of the selected area with camera errors in the order of 1%
Right: View of the selected area with camera errors in the order of 2%

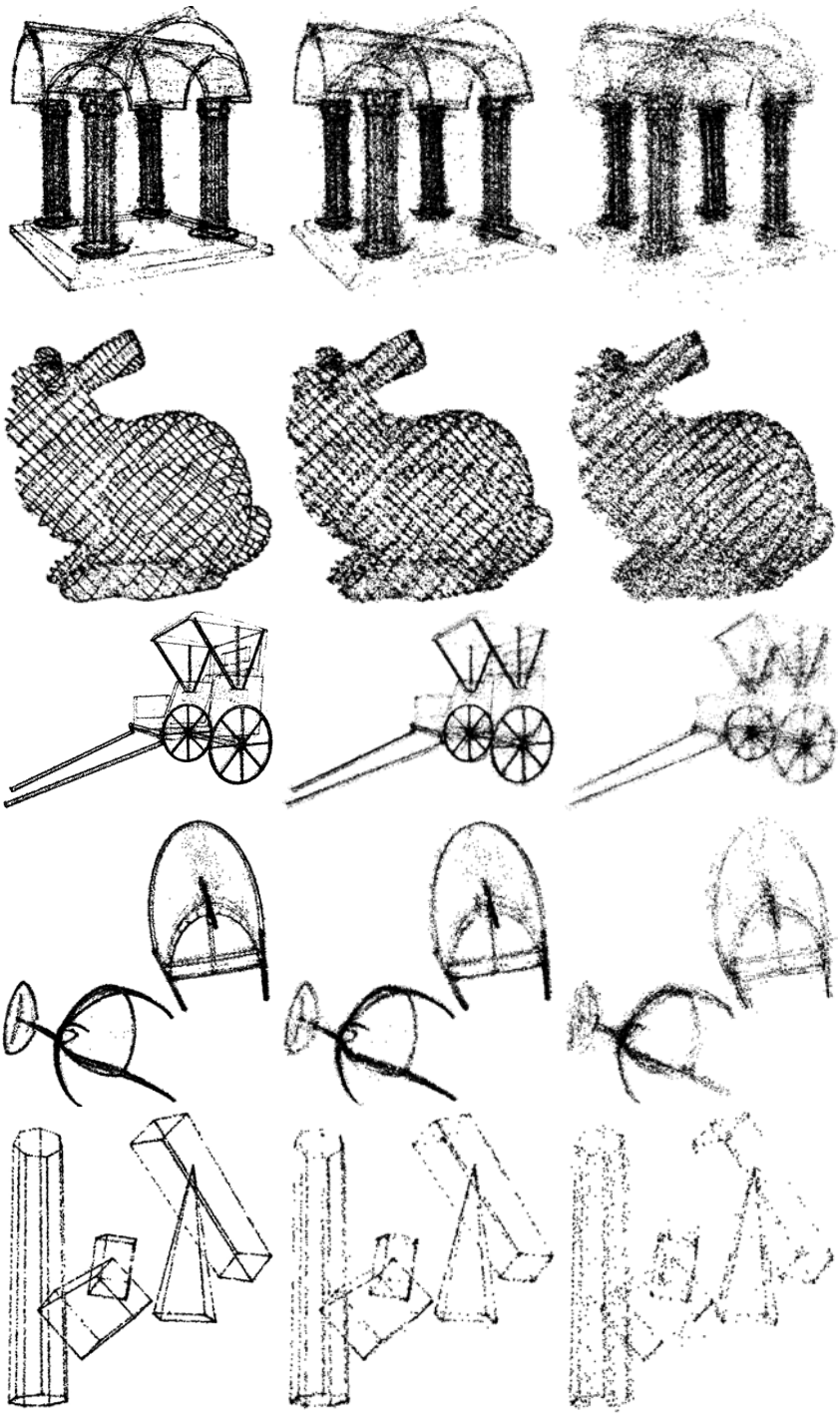
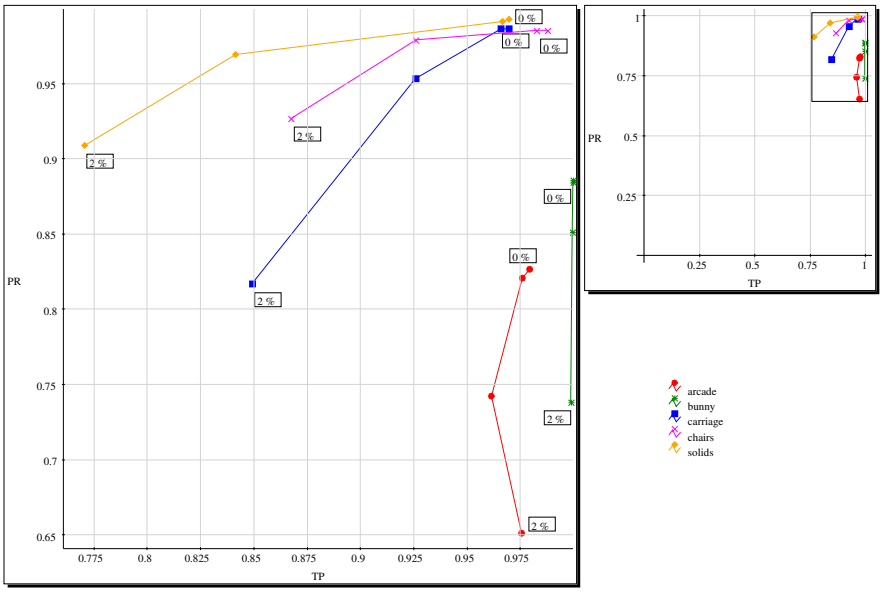


Figure 5.26: *Reconstructions with camera errors.*

Left: Without camera errors
Center: Camera errors in the order of 1 %
Right: Camera errors in the order of 2 %



arcade							
cameraFile	fp	tp	fn	tn	EP	TP	PR
0 %	*14020*	**66689**	1391	*60664*	**80709**	**0.980**	**0.826**
0.5 %	14089	64307	1572	62796	78396	0.976	0.820
1 %	15262	43881	**1757**	81864	59143	*0.962*	0.742
2 %	**16222**	*30240*	*750*	**95552**	*46462*	0.976	*0.651*

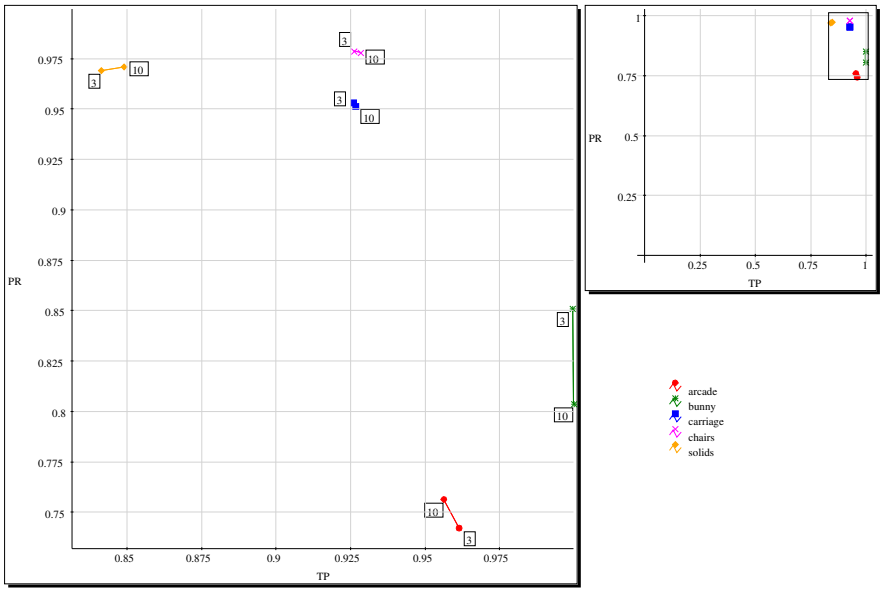
bunny							
cameraFile	fp	tp	fn	tn	EP	TP	PR
0 %	*8658*	**66856**	10	*26038*	**75514**	**1.000**	**0.885**
0.5 %	8732	66575	*5*	26250	75307	**1.000**	0.884
1 %	10441	59540	24	31557	69981	**1.000**	0.851
2 %	**13204**	*37189*	**42**	**51127**	*50393*	*0.999*	*0.738*

carriage							
cameraFile	fp	tp	fn	tn	EP	TP	PR
0 %	1071	**77221**	*2411*	*32930*	**78292**	**0.970**	**0.986**
0.5 %	*1045*	74675	2631	35282	75720	0.966	**0.986**
1 %	1921	39042	**3123**	69547	40963	0.926	0.953
2 %	**3770**	*16781*	2979	**90103**	*20551*	*0.849*	*0.817*

chairs							
cameraFile	fp	tp	fn	tn	EP	TP	PR
0 %	671	**44287**	*546*	*14269*	**44958**	**0.988**	**0.985**
0.5 %	645	42613	739	15776	43258	0.983	**0.985**
1 %	*505*	23144	**1845**	34279	23649	0.926	0.979
2 %	**825**	*10404*	1586	**46958**	*11229*	*0.868*	*0.927*

solids							
cameraFile	fp	tp	fn	tn	EP	TP	PR
0 %	*69*	**9294**	*288*	*4874*	**9363**	**0.970**	**0.993**
0.5 %	80	8999	310	5136	9079	0.967	0.991
1 %	107	3336	**628**	10454	3443	0.842	0.969
2 %	**186**	*1848*	550	**11941**	*2034*	*0.771*	*0.909*

Figure 5.27: *Camera error tests (using default settings).*



arcade											
dtDir	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
10	*14531*	**45097**	**2052**	*81084*	**59628**	*0.956*	**0.756**	0.980	**0.836**	**0.024**	*0.080*
3	**15262**	*43881*	*1757*	**81864**	*59143*	**0.962**	*0.742*	0.980	*0.826*	*0.018*	**0.084**

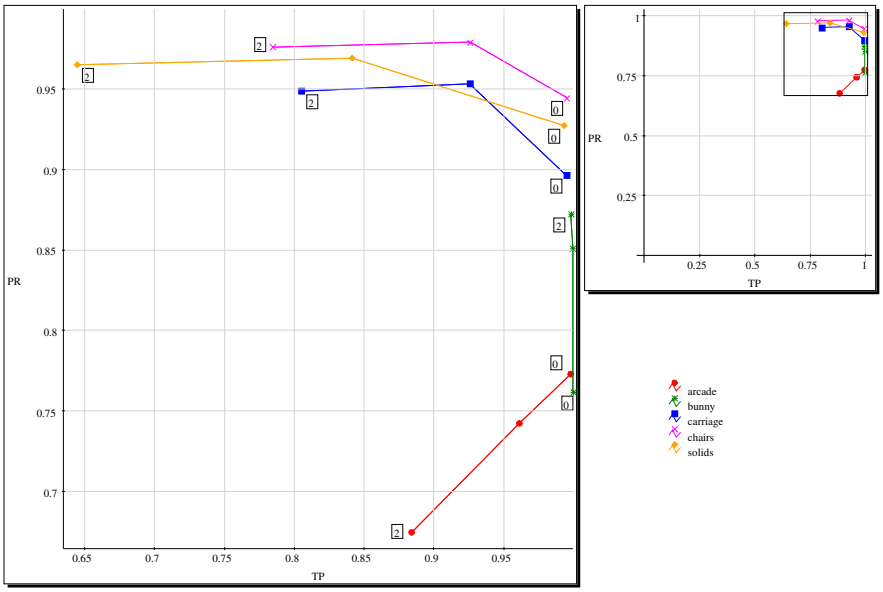
bunny											
dtDir	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
10	**15392**	**62948**	*4*	*23218*	**78340**	1.000	*0.804*	1.000	*0.843*	0.000	**0.039**
3	*10441*	*59540*	**24**	**31557**	*69981*	1.000	**0.851**	1.000	**0.885**	0.000	*0.034*

carriage											
dtDir	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
10	**2004**	**39200**	*3101*	*69328*	**41204**	**0.927**	*0.951*	**0.972**	*0.984*	**0.045**	0.033
3	*1921*	*39042*	**3123**	**69547**	*40963*	*0.926*	**0.953**	*0.970*	**0.986**	*0.044*	0.033

chairs											
dtDir	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
10	**525**	**23252**	*1795*	*34201*	**23777**	**0.928**	*0.978*	0.988	*0.984*	*0.060*	0.006
3	*505*	*23144*	**1845**	**34279**	*23649*	*0.926*	**0.979**	0.988	**0.985**	**0.062**	0.006

solids											
dtDir	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
10	*103*	**3419**	*607*	*10396*	**3522**	**0.849**	**0.971**	**0.972**	*0.992*	*0.123*	*0.021*
3	**107**	*3336*	**628**	**10454**	*3443*	*0.842*	*0.969*	*0.970*	**0.993**	**0.128**	**0.024**

Figure 5.28: *Vote range tests (with camera errors).*Compare with Figure 5.9



arcade											
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
0	**31544**	**107124**	*190*	*3906*	**138668**	**0.998**	**0.773**	**0.999**	0.825	*0.001*	*0.052*
1	15262	43881	1757	81864	59143	0.962	0.742	0.980	**0.826**	0.018	0.084
2	*8021*	*16626*	**2175**	**115942**	*24647*	*0.884*	*0.675*	*0.935*	*0.788*	**0.051**	**0.113**

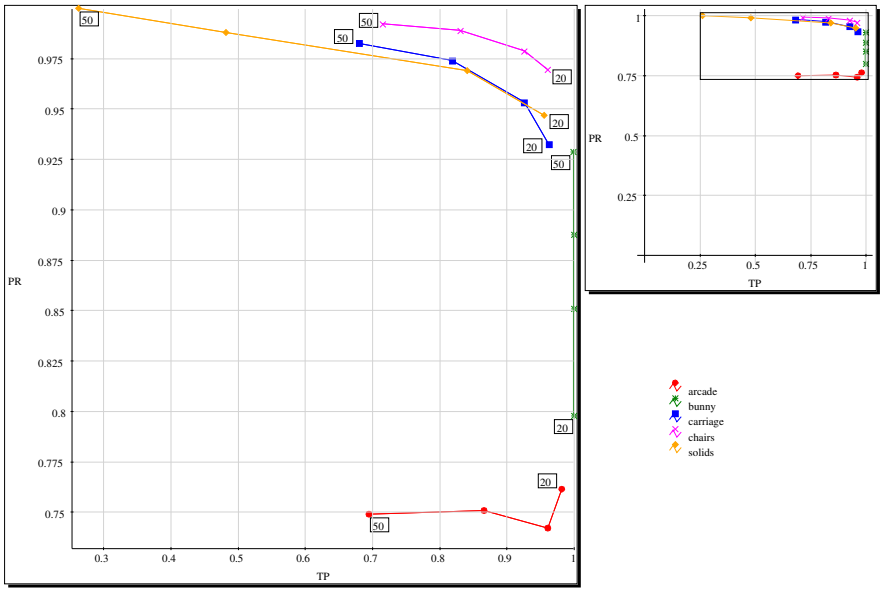
bunny											
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
0	**22735**	**72534**	*18*	*6275*	**95269**	**1.000**	*0.761*	1.000	*0.808*	*0.000*	**0.047**
1	10441	59540	24	31557	69981	**1.000**	0.851	1.000	0.885	*0.000*	*0.034*
2	*5453*	*37163*	**67**	**58879**	*42616*	*0.998*	**0.872**	1.000	**0.916**	**0.002**	0.044

carriage											
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
0	**11088**	**95721**	*471*	*6353*	**106809**	**0.995**	*0.896*	**0.996**	*0.948*	*0.001*	**0.052**
1	1921	39042	3123	69547	40963	0.926	**0.953**	0.970	0.986	0.044	*0.033*
2	*931*	*17125*	**4140**	**91437**	*18056*	*0.805*	0.948	*0.922*	**0.987**	**0.117**	0.039

chairs											
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
0	**3104**	**52425**	*238*	*4006*	**55529**	**0.995**	*0.944*	**0.997**	*0.956*	*0.002*	**0.012**
1	505	23144	1845	34279	23649	0.926	**0.979**	0.988	0.985	0.062	*0.006*
2	*221*	*8890*	**2435**	**48227**	*9111*	*0.785*	0.976	*0.945*	**0.986**	**0.160**	0.010

solids											
crossCheckThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
0	**991**	**12568**	*80*	*886*	**13559**	**0.994**	*0.927*	**0.999**	*0.972*	*0.005*	**0.045**
1	107	3336	628	10454	3443	0.842	**0.969**	0.970	**0.993**	0.128	*0.024*
2	*52*	*1423*	**784**	**12266**	*1475*	*0.645*	0.965	*0.943*	0.992	**0.298**	0.027

Figure 5.29: *Cross-check threshold tests (with camera errors).*Compare with Figure 5.11



arcade											
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
20	**21116**	**67376**	*1234*	*53038*	**88492**	**0.982**	**0.761**	**0.990**	0.837	*0.008*	*0.076*
30	15262	43881	1757	81864	59143	0.962	*0.742*	0.980	*0.826*	0.018	0.084
40	4717	14200	2192	121655	18917	0.866	0.751	0.937	**0.862**	0.071	**0.111**
50	*1779*	*5307*	**2325**	**133353**	*7086*	*0.695*	0.749	*0.858*	0.853	**0.163**	0.104

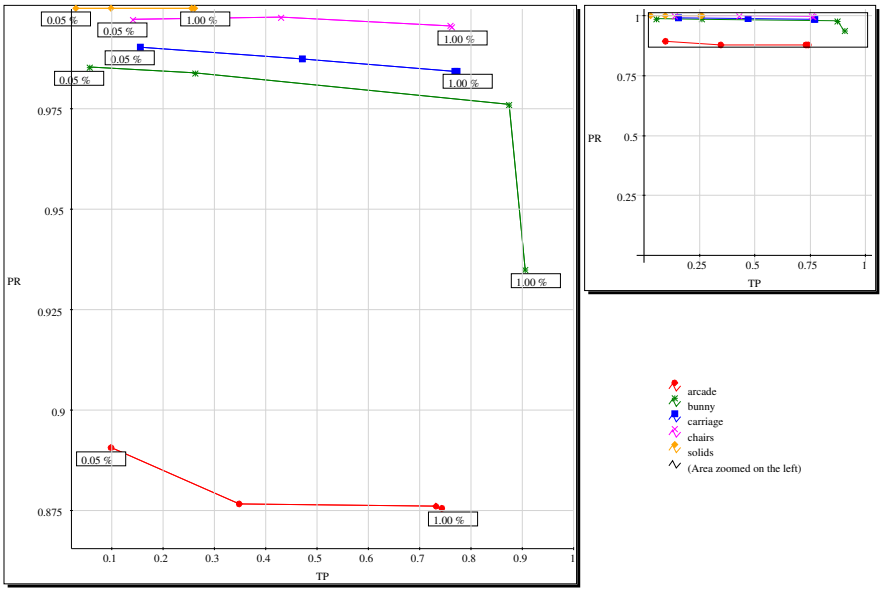
bunny											
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
20	**17639**	**69523**	*19*	*14381*	**87162**	**1.000**	*0.798*	1.000	*0.842*	*0.000*	**0.044**
30	10441	59540	24	31557	69981	**1.000**	0.851	1.000	0.885	*0.000*	0.034
40	6218	49189	26	46129	55407	*0.999*	0.888	1.000	0.913	**0.001**	0.025
50	*2581*	*33603*	**33**	**65345**	*36184*	*0.999*	**0.929**	1.000	**0.943**	**0.001**	*0.014*

carriage											
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
20	**4258**	**58831**	*2267*	*48277*	**63089**	**0.963**	*0.933*	**0.984**	*0.978*	*0.021*	**0.045**
30	1921	39042	3123	69547	40963	0.926	0.953	0.970	0.986	0.044	0.033
40	488	18160	4029	90956	18648	0.818	0.974	0.946	0.994	0.128	0.020
50	*168*	*9382*	**4399**	**99684**	*9550*	*0.681*	**0.982**	*0.910*	**0.996**	**0.229**	*0.014*

chairs											
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
20	**1036**	**33046**	*1360*	*24331*	**34082**	**0.960**	*0.970*	**0.993**	*0.978*	*0.033*	**0.008**
30	505	23144	1845	34279	23649	0.926	0.979	0.988	0.985	0.062	0.006
40	133	11665	2378	45597	11798	0.831	0.989	0.977	0.991	0.146	*0.002*
50	*52*	*6486*	**2583**	**50652**	*6538*	*0.715*	**0.992**	*0.960*	**0.994**	**0.245**	*0.002*

solids											
similarCameraAngle	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR
20	**441**	**7842**	*358*	*5884*	**8283**	**0.956**	*0.947*	**0.991**	*0.983*	*0.035*	**0.036**
30	107	3336	628	10454	3443	0.842	0.969	0.970	0.993	0.128	0.024
40	9	739	792	12985	748	0.483	0.988	0.934	0.998	0.451	0.010
50	*0*	*294*	**825**	**13406**	*294*	*0.263*	**1.000**	*0.896*	**0.999**	**0.633**	*-0.001*

Figure 5.30: *Similar camera angle tests (with camera errors).*Compare with Figure 5.13



arcade												
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
0.05 %	*557*	*4533*	**41332**	**96342**	*5090*	*0.099*	**0.891**	*0.177*	**0.948**	0.078	0.057	*2349*
0.10 %	2288	16254	30434	93788	18542	0.348	0.877	0.488	0.937	**0.140**	**0.060**	6708
0.50 %	5149	36363	13277	87975	41512	0.733	*0.876*	0.800	0.915	0.067	0.039	**6997**
1.00 %	**5275**	**37113**	*12770*	*87606*	**42388**	**0.744**	*0.876*	**0.804**	*0.905*	*0.060*	*0.029*	6843

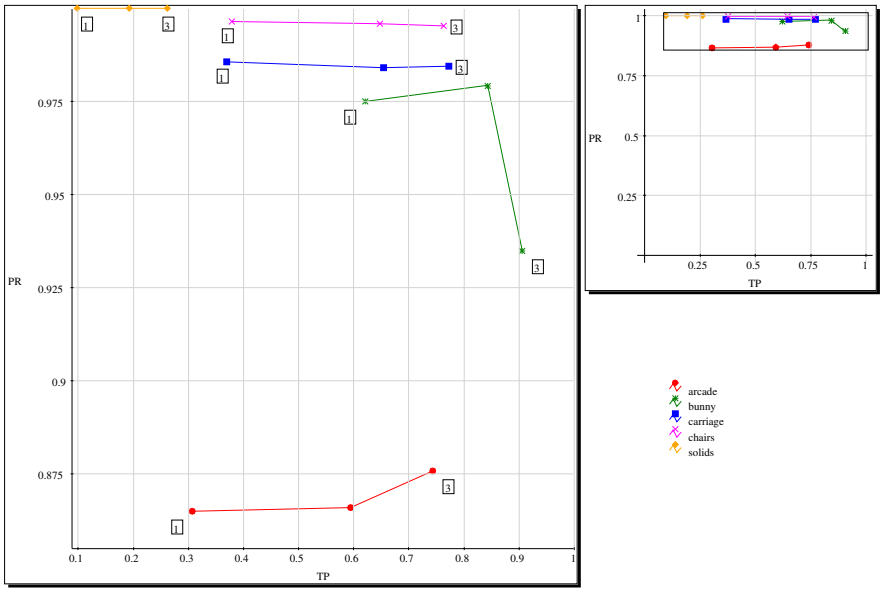
bunny												
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
0.05 %	*50*	*3335*	**56313**	**41864**	*3385*	*0.056*	**0.985**	*0.124*	**0.993**	0.068	**0.008**	*1627*
0.10 %	259	15766	44346	41191	16025	0.262	0.984	0.422	0.992	**0.160**	**0.008**	**6088**
0.50 %	1383	56204	8062	35913	57587	0.875	0.976	0.894	0.973	0.019	-0.003	4479
1.00 %	**3961**	**56811**	*5900*	*34890*	**60772**	**0.906**	*0.935*	**0.913**	*0.877*	*0.007*	*-0.058*	3281

carriage												
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
0.05 %	*64*	*6533*	**35653**	**71383**	*6597*	*0.155*	**0.990**	*0.259*	**0.998**	0.104	0.008	*2874*
0.10 %	254	19901	22354	71124	20155	0.471	0.987	0.644	**0.998**	**0.173**	**0.011**	**5457**
0.50 %	521	32626	9854	70632	33147	0.768	*0.984*	0.869	0.922	0.101	-0.062	3067
1.00 %	**522**	**32880**	*9667*	*70564*	**33402**	**0.773**	*0.984*	**0.871**	*0.921*	*0.098*	*-0.063*	3010

chairs												
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
0.05 %	*10*	*3512*	**21493**	**34758**	*3522*	*0.140*	0.997	*0.260*	**1.000**	0.120	**0.003**	*1549*
0.10 %	25	10771	14275	34702	10796	0.430	**0.998**	0.627	0.999	**0.197**	0.001	**3185**
0.50 %	85	19104	6057	34527	19189	0.759	0.996	0.869	*0.884*	0.110	*-0.112*	1687
1.00 %	**93**	**19206**	*5957*	*34517*	**19299**	**0.763**	*0.995*	**0.871**	*0.884*	*0.108*	-0.111	1662

solids												
merge3DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
0.05 %	0	*122*	**3842**	**10561**	*122*	*0.031*	1.000	*0.138*	1.000	*0.107*	0.000	*58*
0.10 %	0	390	3576	10559	390	0.098	1.000	0.377	1.000	0.279	0.000	158
0.50 %	0	1015	2958	10552	1015	0.255	1.000	0.586	1.000	**0.331**	0.000	283
1.00 %	0	**1041**	*2933*	*10551*	**1041**	**0.262**	1.000	**0.589**	1.000	0.327	0.000	**287**

Figure 5.31: *Merging threshold 3D tests (with camera errors).*Compare with Figure 5.22



arcade												
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
1	*2237*	*14310*	**32259**	**93958**	*16547*	*0.307*	*0.865*	*0.421*	**0.933**	**0.114**	**0.068**	*6522*
2	4447	28667	19502	90148	33114	0.595	0.866	0.692	0.923	0.097	0.057	**8502**
3	**5275**	**37113**	*12770*	*87606*	**42388**	**0.744**	**0.876**	**0.804**	*0.905*	*0.060*	*0.029*	6843

bunny												
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
1	*988*	*38384*	**23390**	**38800**	*39372*	*0.621*	0.975	*0.730*	**0.987**	**0.109**	**0.012**	**7640**
2	1154	54147	10059	36202	55301	0.843	**0.979**	0.874	0.973	0.031	-0.006	4777
3	**3961**	**56811**	*5900*	*34890*	**60772**	**0.906**	*0.935*	**0.913**	*0.877*	*0.007*	*-0.058*	*3281*

carriage												
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
1	*228*	*15587*	**26675**	**71143**	*15815*	*0.369*	**0.986**	*0.537*	**0.998**	**0.168**	0.012	**5645**
2	451	27701	14684	70797	28152	0.654	*0.984*	0.797	0.997	0.143	**0.013**	4959
3	**522**	**32880**	*9667*	*70564*	**33402**	**0.773**	*0.984*	**0.871**	*0.921*	*0.098*	*-0.063*	*3010*

chairs												
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
1	*34*	*9472*	**15568**	**34699**	*9506*	*0.378*	**0.996**	*0.561*	**0.999**	**0.183**	**0.003**	**3291**
2	68	16246	8864	34595	16314	0.647	**0.996**	0.803	0.998	0.156	0.002	2678
3	**93**	**19206**	*5957*	*34517*	**19299**	**0.763**	*0.995*	**0.871**	*0.884*	*0.108*	*-0.111*	*1662*

solids												
merge2DThreshold	fp	tp	fn	tn	EP	TP	PR	TP(*)	PR(*)	Δ TP	ΔPR	TM
1	0	*392*	**3576**	**10557**	*392*	*0.099*	1.000	*0.351*	1.000	*0.252*	0.000	*161*
2	0	768	3204	10553	768	0.193	1.000	0.521	1.000	**0.328**	0.000	241
3	0	**1041**	*2933*	*10551*	**1041**	**0.262**	1.000	**0.589**	1.000	0.327	0.000	**287**

Figure 5.32: *Merging threshold 2D tests (with camera errors).*Compare with Figure 5.24

5.5 Tests with real scenes

To assess the applicability of the algorithm, several tests were performed with real scenes. These types of scenes inherently have inaccurate camera calibration and faulty contour detection, and the reconstruction quality is not trivial to judge, since there is no accurate ground truth data available.

Therefore, the results presented in this section for each of the scenes will be supported by images from different views of the reconstructed scenes, in some cases superimposed with the original images, and an analysis of those results.

Since camera calibration is outside the scope of this work, a camera calibration pattern was introduced in the scene, and the cameras were calibrated by manually indicating some correspondences in a commercial camera calibration package. This pattern was not used or explored in any particular way in any stage of the reconstruction other than camera calibration.

5.5.1 Audio speaker

This scene consists of 30 images of 768x1024 pixels of an audio loudspeaker. The speaker is composed of the following parts (Figure 5.33):

- A black plastic part (P) with:
  - A flat front face containing:
    - A small colored and reflective label (L), in the upper part;
    - Two small speakers, each consisting of a metallic circle (membrane) (M), surrounded by a black ring, and with a black vertical "bridge" (b) in front of it, protruding from the flat front face;
    - An engraved horizontal line at the middle of the face;
  - A rounded back (B) with some engraved letters slightly above its middle;
- A silver-colored metallic stand (S), with curved base and sides. The plastic part is fixed within this stand. Centered in the lower part of the back of the stand there are two small vertical stripes;
- A black wire (W), of slightly irregular shape, connecting to the back of the black part, and extending outside the scene.

Under the speaker, there is a camera calibration pattern, containing an array of small crosses. As mentioned before, calibration was performed manually using a commercial software package [EOS].

Several aspects of this scene and its reconstruction deserve attention. The speaker shows highlights in parts such as the stand and the two round membranes and the label of the front face (detailed in Figure 5.34).

There are some apparent lighting variations due to the fluorescent lamps and camera's automatic adjustments. Some self-occlusions are also present in the images, such

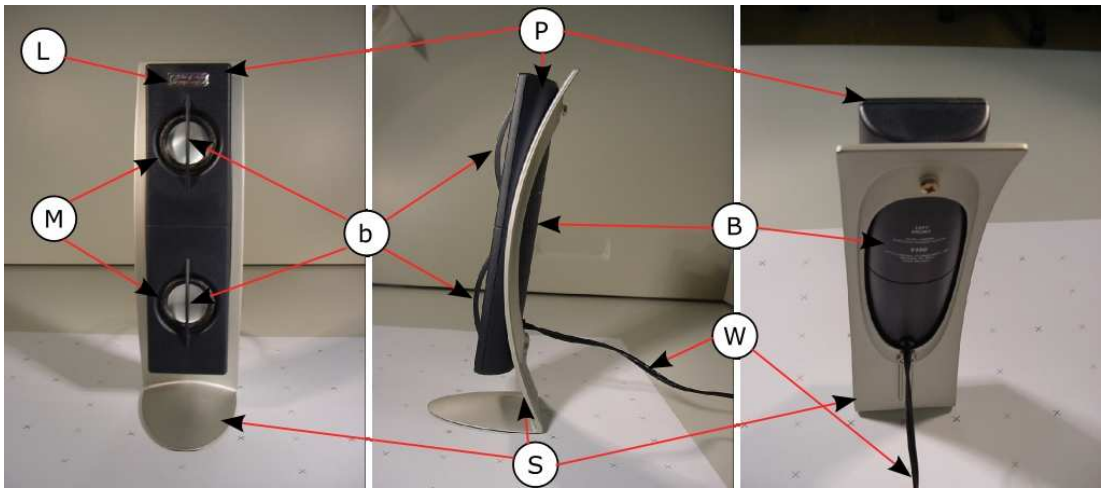


Figure 5.33: *Three views of the speaker.*
The main parts are labeled as follows (refer to the text for more details): (P) Plastic part; (L) Label; (M) Metallic membranes and rings; (b) "bridges"; (B) Back face; (S) Stand; (W) Wire



Figure 5.34: *Variations in appearance caused by lighting and reflection.*
Notice how the colors and intensities of the label and the membrane change from one view to another

as the ones caused by the "bridges". The back of the plastic part has very few contours. Some are apparent, others correspond to the engraved letters.

The images used for reconstruction were taken from various positions and angles around the speaker, at irregular distances, as shown in Figure 5.35, and the set of parameters used is presented in Table 5.3.

Figure 5.36 shows various views of the reconstructed points, without merging, including a detailed profile view. In this view, some of the parts identified in Figure 5.33 are pointed out.

The reflectivity and contents of the label (L) give rise to dense contours in that region, which results in some outliers in that area, although within a close range of the expected surface. The same happens with the metallic membranes, which despite the density of contours are in general contained in the correct region (some outliers are still visible, though).

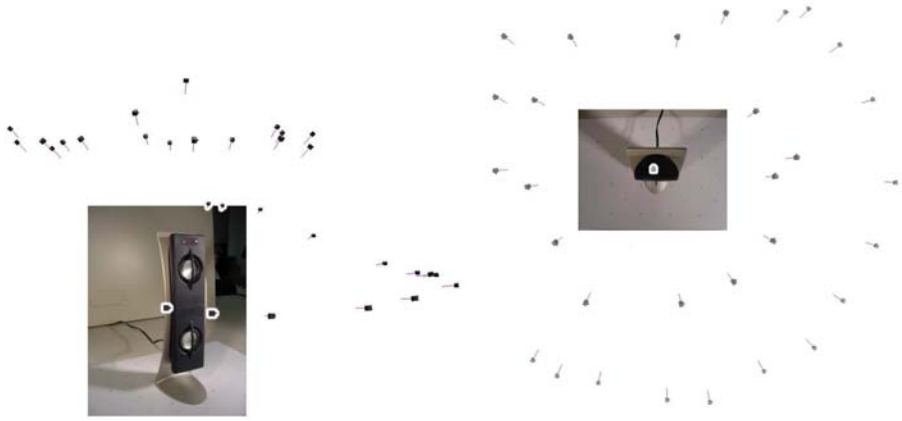


Figure 5.35: *Camera placement for the speaker scene*.
The speaker's images shown are meant to provide a context of the camera placement relative to the speaker

Parameter	**Default Value**
Set of points	features (1.2)
Vote Range	3 votes
Cross-Check Threshold	1 vote
Similar Camera Angle	30 deg.
Vote Percent Threshold	90%
Nr. of Candidates	800 cands.
2D Merge Threshold	3 pixels
3D Merge Threshold	0.5% of scene scale

Table 5.3: *Parameters values used on the speaker scene*.

The "bridges" (b) are interestingly reconstructed: despite being a small detail, they are captured from sufficient views and give rise to significant contours. Their curved shape can be clearly identified in the image. The majority of the reconstructed points in the black back part correspond to the engraved letters (l) and the non-apparent contours. This was expected, since most of this part does not give rise to contours. The ones that do, are in general reconstructed with good localization.

The wire (W) is also nicely reconstructed, as well as the stand borders and the small crosses in the camera calibration pattern. The co-planarity of the points of the camera calibration pattern can be seen in the middle view of Figure 5.36. Given that there is not any regularization step involved in the point reconstruction, that co-planarity is by itself a good indication of the reconstruction quality.

Figure 5.37 shows three views of the reconstruction after merging. The point set is inherently sparser, but it is noticeable that a significant part of the outliers have been removed.

To obtain an approximate idea of the reconstruction quality, measurements were taken in the real object and the reconstruction, and compared up to a scale. The ratio between the segments measured in the reconstruction and the real scene should ideally be equal for all segments. Given errors in the process of measuring the real scene and

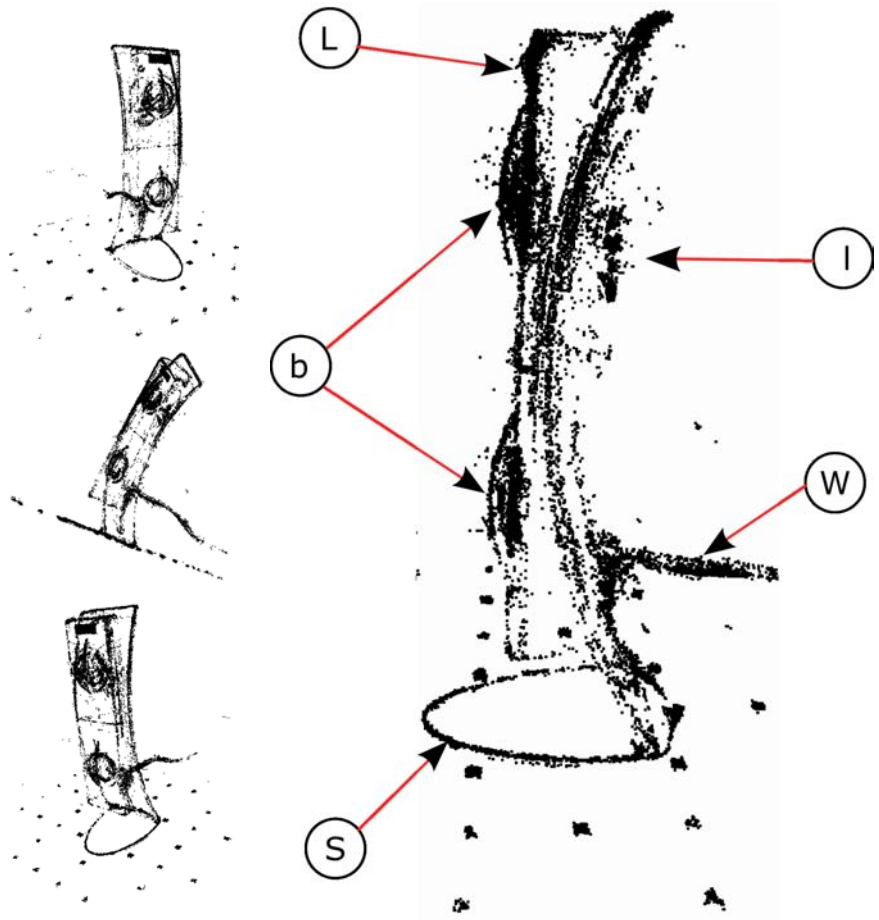


Figure 5.36: *Point reconstruction of the speaker scene, before merging*.
Some of the apparent cluttering is due to the overlapping of close and distant points.

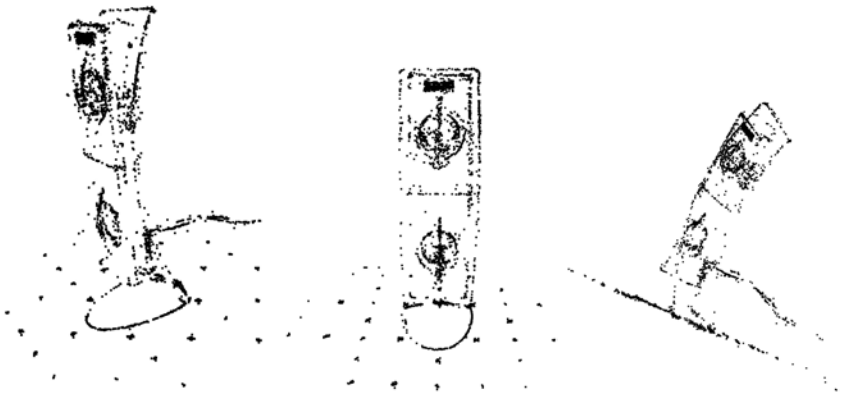


Figure 5.37: *Point reconstruction of the speaker scene, after merging*.

in the reconstruction, some differences should be observed.

Figure 5.38 shows the points between which measures were taken, and Table 5.4 contains the results of the measuring. As the table shows, despite measurement errors, the individual ratios are close to the average value of 1.399. Using this average ratio to convert the measurements of the reconstruction from world units to centimeters, an average error of less than one millimeter was obtained (0.068 cm). Comparing the errors proportionally to the diagonal of the bounding box of the speaker (20 cm), one gets an average error in the order of 0.342% of the scene's dimensions. Note that this bounding box's diagonal is a more strict (tighter) measure than the scene's dimension used for the reconstruction, which was larger and, if used, would yield an even lower error percentage. Note also that there are measurement errors involved, namely due to parallax errors in the measuring process itself, to the low precision of the measuring tools, and also due to possible errors in the choice of each reconstructed point from among a set of points very close to each other. Furthermore, some of the points are located at rounded corners of the speaker, which are sources of apparent contours and may therefore lead to higher errors in the reconstruction. Taking into account these limitations, the results obtained meet the expectations created by the results on synthetic scenes.

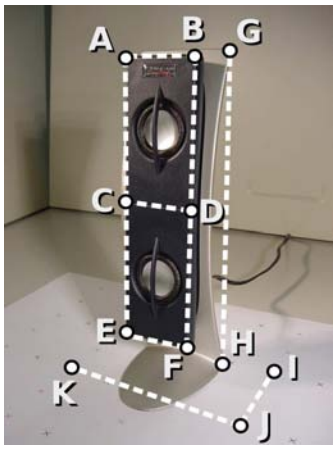


Figure 5.38: *Measurements on the speaker scene*.

Segment	**Recons. Length (world units)**	**Real Length (cm)**	**Ratio (real/recon)**	**Recons. length (cm using avg. ratio)**	**Measurement error (recons. vs real, in cm)**	**Error relative to scene's diagonal (20cm) (%)**
AB	2.635	3.8	1.442	3.686	0.114	0.572%
CD	2.719	3.8	1.397	3.803	0.003	0.017%
EF	2.784	3.8	1.365	3.894	0.094	0.472%
GH	12.837	18.0	1.402	17.956	0.044	0.221%
IJ	3.796	5.3	1.396	5.309	0.009	0.047%
JK	7.766	11.0	1.416	10.863	0.137	0.687%
AC	5.361	7.5	1.399	7.499	*0.001*	*0.004%*
CE	5.411	7.5	1.386	7.568	0.068	0.341%
BD	5.386	7.5	1.393	7.534	0.034	0.168%
DF	5.235	7.5	1.433	7.322	**0.178**	**0.889%**
			Average Ratio		**Average Measurement error (cm)**	**Average error relative to scene's diagonal (%)**
			1.399		0.068	0.342%

Table 5.4: *Measurements of real and reconstructed segments: speaker scene*.



Figure 5.39: *Camera placement for the boat scene*.

5.5.2 Boat

This scene contains a small wooden boat (Figure 5.42(a)), depicted in 39 images of 1024x768 pixels, taken from various positions and angles around the boat (Figure 5.39).

It is a particularly challenging scene, due to the large amount of occlusions between the ropes, the masts and the cabinets on the deck. The contour extraction in the ropes' area may also be the source of some problems, due to contours being very close to each other.

The parts of interest of the boat are the following (Figure 5.40):

- The hull (H), consisting mostly of a green color, with an engraved line in the middle, and a white line in the bottom;
- Two white cabins (C), with small black windows and red roof;
- The masts (M), at four different positions in the boat's length:
  - A protruding mast in the front of the boat;
  - A set of three masts in the first third of the boat;
  - Another set of three masts in the middle (right behind the first cabin);
  - An isolated mast in the back of the boat (behind the second cabin);
- A sail at the rear of the boat (S);
- A large number of ropes (R) connecting the masts, the deck and the cabins;
- Two black finns (F): one in the front and other in the rear.

The two parts of this scene which seem more challenging to reconstruct are the ropes and the masts. Both create a high density of contours, as it can be seen in Figure

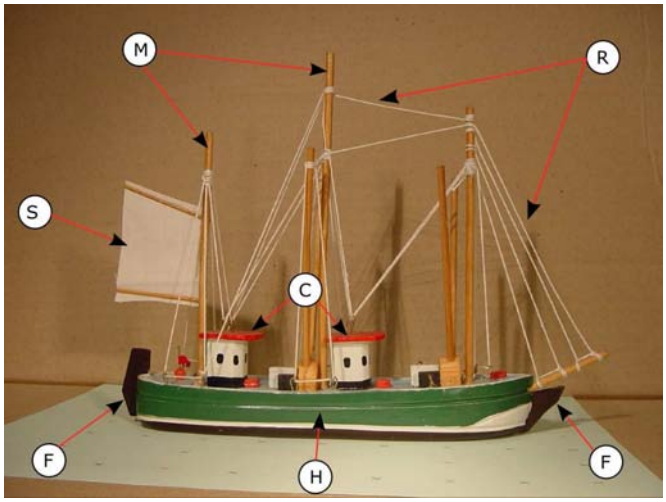


Figure 5.40: *Details of the boat.*

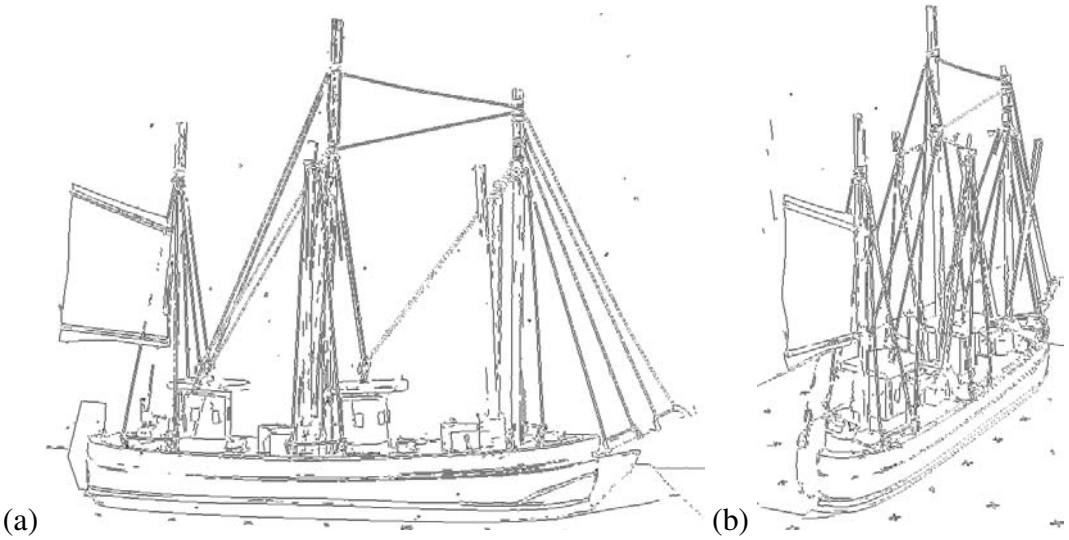


Figure 5.41: *Examples of contour maps of the boat scene.*
Multiple occlusions and missing contours are noticeable in both images. The contour density and intertwining is particularly strong in (b).

5.41. The ropes give rise to several occlusions and strong contour intertwining. The masts also generate multiple occlusions, and their contours are mostly apparent.

Figure 5.42 and Figure 5.43 show the results of a reconstruction of the boat scene, before and after merging, from two views from the camera set (Figure 5.42) and two other arbitrary views (Figure 5.43)

The ropes seem mostly well reconstructed, as can be seen in the images. They are both well-defined and with good localization. The masts are also in general well localized, but are missing some of the surrounding (apparent) contours. Since those apparent contours only appears in very few images (sometimes only one) they do not gather sufficient votes to be considered valid. It is interesting to notice, however, how

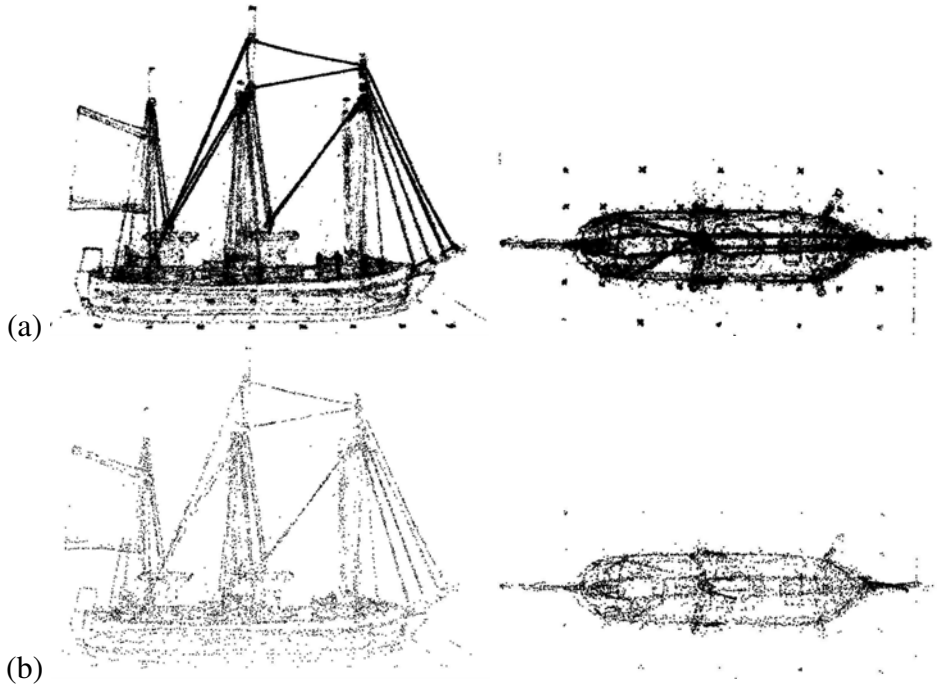


Figure 5.42: *Views of the boat reconstruction taken from existing cameras.*
Two views of the reconstruction of feature points on the boat scene taken from existing cameras, before (a) and after (b) merging

the top of the masts are nicely reconstructed. The cabins' rooftops also seem correctly reconstructed, as well as some of the other elements lying on the deck, but it is difficult to distinguish them in the virtual images, due to the overlapping of points which are farther back in the scene (such as the points from the calibration pattern).

Merging causes a decrease in point density, but also removes most of the outliers that can be seen on the sides of the boat (right images of Figure 5.43).

The results show special details on the thinner structures, such as the ropes and the masts, that are not easy to reconstruct with other techniques. The remaining structures are also in general well localized, but in some cases there seems to be some fuzziness (namely close to the cabinets). This is most likely due to a high density of contours in the area. An example of this contour density is shown in Figure 5.41. Still, the overall line structure of the boat can be easily recognized from arbitrary views.

Again, to have a quantitative assessment of the reconstruction quality, measurements were taken in the real object and the reconstruction, and compared up to a scale. Figure 5.44 shows the points between which measures were taken, and Table 5.5 contains the results of the measuring. Also in this case, the individual ratios are close to the average value, in this case of 1.231. The average error measured is below one millimeter (0.081 cm). Comparing the errors proportionally to the diagonal of the bounding box of the boat (30 cm), one gets an average error in the order of 0.271% of the scene's dimensions. Note that, as with the speaker scene, this bounding box's diagonal is a more strict (tighter) measure than the scene's dimension used for the reconstruction, which was larger.

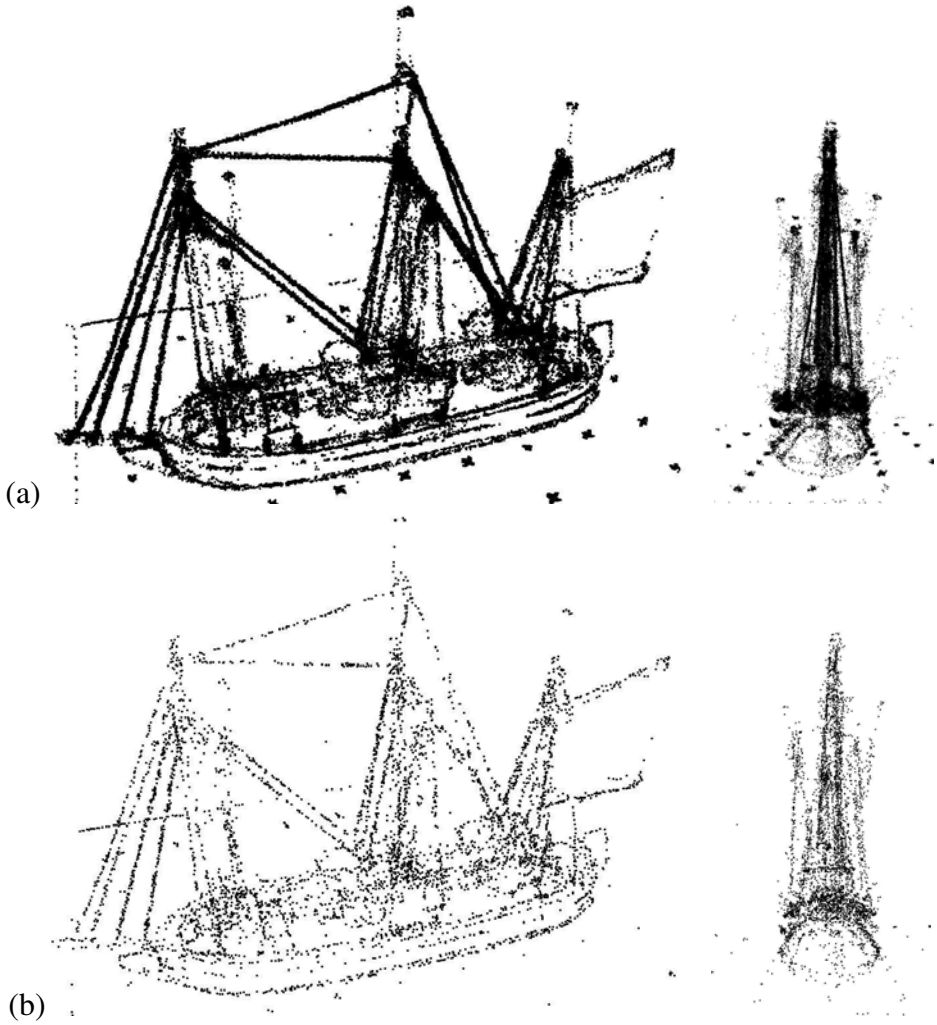


Figure 5.43: *Arbitrary views of the boat reconstruction.*
Two arbitrary views of the reconstruction of feature points on the boat scene before (a) and after (b) merging

These values are again a good indication of the quality of the reconstruction, especially considering that there are errors by excess due to the measuring process.

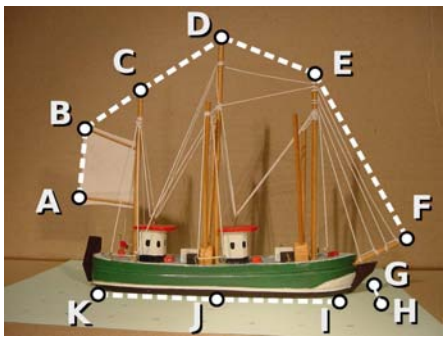


Figure 5.44: *Measurements on the boat scene.*

Segment	**Recons. Length (world units)**	**Real Length (cm)**	**Ratio (real/recon)**	**Recons. length (cm using avg. ratio)**	**Measurement error (recons. vs real, in cm)**	**Error relative to scene's diagonal (30cm) (%)**
AB	4.121	5.2	1.262	5.071	0.129	0.431%
BC	4.162	5.1	1.225	5.121	*0.021*	*0.071%*
CD	5.871	7.0	1.192	7.224	**0.224**	**0.748%**
DE	5.990	7.4	1.235	7.370	0.030	0.098%
EF	10.916	13.5	1.237	13.432	0.068	0.226%
GH	4.339	5.3	1.221	5.340	0.040	0.133%
IJ	6.775	8.3	1.225	8.337	0.037	0.122%
JK	6.662	8.3	1.246	8.198	0.102	0.341%
			Average Ratio		**Average Measurement error (cm)**	**Average error relative to scene's diagonal (%)**
			1.231		0.081	0.271%

Table 5.5: *Measurements of real and reconstructed segments: boat scene*.

5.5.3 Remarks on results with real scenes

The two scenes presented serve as examples of several types of problems that can be found in different scenes, and that cause problems to different reconstruction methods.

The speaker scene has highlights, regions of dense contours and apparent contours, as well as some self-occlusion. Highlights in highly reflective places such as the front label caused some very minor outliers. Still, if one takes into consideration that no regularization was used whatsoever, the results in that area are very good. Correlation-based techniques without regularization are expected to cause strong outliers in such regions, and the color consistency measures of voxel-based or other multiple baseline approaches based on color or intensity would have problems as well, as there are many variations from image to image.

The areas of apparent contours in general did not affect the reconstruction. In parts where apparent contours where coincident with real contours, such as the letters on the back of the speaker, the real contours were reconstructed with good localization.

In the boat scene, problems such as apparent and dense contours, thin structures, highlights and occlusions occur concurrently in different parts. The method has shown good robustness and localization in most cases, with particular detail on the ropes, and on multiple objects on the deck which were occluded in one way or another in most images.

Correlation-based methods would have problems in reconstructing thin structures as the ropes, and in dealing with the multiple occlusions between ropes and masts, not to mention the variations in camera angles and positions. Visual-hull based techniques would also fail, as the extraction of silhouettes would cause several problems, and end up connecting different parts of the boat. The thin rope structures, together with eventual camera calibration errors would also cause some problems in voxel-based approaches, as the color consistency can be easily broken if a small offset is observed and causes a voxel on the rope to be checked against e.g. the background. With the proposed reconstruction method, the use of contours and distance-based voting compensates for that in many cases.

5.6 Performance tests

This section contains the results of tests used to assess the performance of the algorithm and its GPU-based implementation. The performance results are independent of the types of scenes used (real or synthetic). Therefore, all the tests were conducted using the same scene – the Boat scene (described in Section 5.5) – and the 4460 feature points of its first reference image. The number of points is close to the average number of feature points of the other images, and is therefore a representative reference image.

A first test was conducted to compare between single and multiple line buffer methods (described in Section 4.4), to assess how significant can be the differences between both methods.

Performance tests comparing the CPU-based implementation of the algorithm with the GPU-based implementation follow. These implementation variants were submitted to two different tests, one showing the influence of varying the number of candidates, and another to compare performances on systems with different hardware.

5.6.1 Single-line buffer versus multi-line buffer

In Section 4.4, it was suggested that the time-expensive operation of transferring data from the graphics card to the main memory could be optimized by rendering multiple view rays in different lines of a large buffer, followed by a single buffer read.

Tests were performed to assess how this optimization would impact the performance of a reconstruction. The results are presented in Figure 5.45.

The time decreases exponentially with the increase of the buffer size. Results show that a gain of about 43% of the time can be achieved when using buffer sizes above 150 lines, when compared to a single-line buffer, and above those sizes, the values are maintained. For the following tests a buffer size of 400 lines will be used.

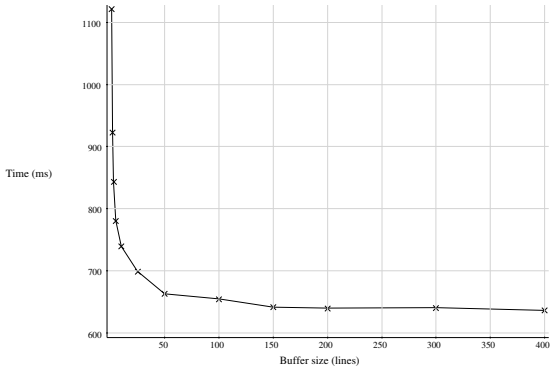
5.6.2 Number of candidates

This test serves to show the influence that the *number of candidates* has in reconstruction time.

In Section 5.3.7, it was seen that varying the number of candidates above a given value does not influence significantly the reconstruction quality (given the threshold of 0.5% used for the error metric). However, the number of candidates plays a major role in the computational load of the reconstruction.

For this reason, reconstruction was performed on the Boat scene (described in Section 5.5), using 400, 800 and 1200 candidates, for 4460 feature points of the first image.

The results in Table 5.6 show the average time in milliseconds required for reconstructing the 4460 feature points of a reference image using the remaining 38 images, evaluating 400, 800 and 1200 candidates, using the two implementation variants (CPU



Buffer Size (lines)	GPU time (ms)
1	**1121.33**
2	922.83
3	843.16
5	780.51
10	739.33
25	698.83
50	662.83
100	654.33
150	641.16
200	639.5
300	640.16
400	*636.33*

Figure 5.45: *Performance results for optimizing the number of lines rendered per buffer reading*.
Results of the reconstruction of 4460 feature points, by evaluating 800 candidates for each point in 38 images, using the single-line buffer, and multi-line buffers off different sizes

and GPU). The test was performed on System A, with an AMD Athlon XP 2400+ processor, 512 MB RAM and a NVidia GeForce4 Ti 4600 graphics card.

Nr. of Candidates	**CPU time (ms)**	**GPU time (ms)**	**CPU/GPU**
400	*17 736*	*565*	*31.39*
800	34 317	636	53.96
1200	**50 179**	**915**	**54.84**

Table 5.6: *Performance results for varying numbers of candidates*. Results for reconstructing 4460 feature points, evaluating 400, 800 and 1200 candidates in 38 different images (time in milliseconds)

There is an overhead when the system runs for the first time due to the initial setup of cameras and textures and loading images. The overhead for the first run is very similar for both the GPU and CPU times and varied between 3 and 6 seconds in the tests performed for this section. Subsequent runs are substantially faster and hence the values presented in Table 5.6 represent the average time per reference image, considering a set of 39 iterations.

The number of cross-checking and other conditional operations performed on the CPU, which vary with different candidate sets, cause variations in the order of some milliseconds. These variations have a higher relative impact in the GPU times (which total under one second), than in the CPU times. As a result, while varying the number of candidates increases the CPU times of about 17 seconds per 400 candidates (close

to linear), in the GPU case, the increase is not linear.

The gains over the CPU implementation are quite significant, starting from a 31-fold speed increase for the smaller candidate set, and reaching almost a 55-fold increase for the largest set, almost doubling the advantage of using projective texturing versus the CPU implementation, when compared to the smallest set. An important result is that using the smaller set of candidates, a reconstruction can be achieved for a reference image in about 565 ms (or less than 23 seconds using all of the 39 images as reference). This confirms the expectations on the advantages of using the graphics pipeline, which is highly optimized for the type of geometric computations performed by the algorithm.

5.6.3 Different hardware

This final test compares the performance under different hardware settings. It shows that even with less powerful hardware there is still an advantage in using projective texturing.

Two hardware systems were used: *System A* is the same system used in the previous test. *System B* consists of a computer with a processor of similar performance to the one used on system A – a Pentium IV Celeron running at 2.4 GHz – but with a less powerful graphics card – a NVidia GeForce MX 440.

The scene and number of points were the same as in the previous tests. Tests were performed in each of the systems, and using both implementation variants. Table 5.7 shows the time results in milliseconds of both implementation variants in the two hardware systems. These times include the reconstruction of the 4460 feature points, by evaluating 800 candidates for each point, using 38 images.

System	CPU time (ms)	GPU time (ms)	CPU/GPU
A	**34 317**	*636*	**53.96**
B	*33 892*	**8 579**	*3.95*

Table 5.7: *Performance results for different hardware systems*. Results of the reconstruction of 4460 feature points, by evaluating 800 candidates for each point using 38 images, in different hardware systems (time in milliseconds)

The results show that, even when using less powerful graphics systems, it is still possible to obtain performance gains with the GPU-based implementation, although to a lesser extent.

5.7 Summary and conclusions

This chapter presented a series of experiments and tests performed using the reconstruction method described in Chapter 3, and its hardware-based implementation proposed in Chapter 4.

These tests were done with different goals in mind, namely:

- to measure the quality of the reconstructions;
- to analyze the influence of some parameters of the algorithm in the resulting reconstructions;
- to assess the robustness of the algorithm in adverse conditions;
- to verify its applicability to real scenes;
- to evaluate the relative performance of the hardware-based implementation.

It is not trivial to measure and assess quality and robustness, especially in real scenes. Therefore, tests were first performed with synthetic scenes for which ground truth data was available. Even in this case, some ambiguities existed on how to measure errors e.g. for points that were incorrectly reconstructed in a surface different from the one were they were first selected. An error metric taking this and other issues into account was proposed. This error metric was used, on one hand, to evaluate the quality of the estimates computed by the method. On the other hand, the same error metric was also used to judge the method's ability to accurately detect and reject incorrect estimates.

To this end, well-known tools from the medical and data mining community for evaluating the performance of classifiers were used, namely precision/recall charts that graphically shows the trade-off of a classifier between, on one hand, accurately filtering incorrect estimates – Precision – and, on the other hand, not rejecting correct estimates – Recall.

Reconstruction was then performed testing variations of one parameter at a time, in five synthetic scenes. Precision/recall charts were generated to analyze the influence of the parameter variation in the quality of the estimates. This, together with a visual assessment of the reconstructed point sets, allowed to draw conclusions on how a parameter affects the quality of the reconstruction, and suggest how it could be set in particular types of scenes, if required or beneficial.

The tests with synthetic scenes were performed with both exact and error-added camera calibration data.

With exact camera calibration, very high values of precision and recall were achieved, and it was possible to isolate in most cases the influence of each parameter in the results. The set of points, the vote range and the number of candidates (above the required resolution) in general do not affect the quality of results as significantly as the other parameters.

The choice of values for cross-check threshold, similar camera angle or vote percent threshold, have a much stronger influence in the quality of the results. Both cross-check threshold and similar camera angle rule the cross-checking procedure, and it was concluded that too conservative values of these parameters, although providing high precision, lead to an excessive rejection of points. On the other hand, too liberal parameters result in an excess of false positives, thus pointing to a choice of intermediate values that provide a good compromise.

It was also concluded that the vote percent threshold, should neither be too liberal, as it would introduce too many low maxima, nor too strict, as it would have no effect.

The tests on merging parameters provided an insight of the excessive merging and rejection effects. Acceptable values for these parameters were also chosen, based on a compromise between the aforementioned effects.

In general the parameters behaved as expected, and it was possible to identify safe default settings or settings' ranges for the tested parameters. With errors in camera calibration, the values of precision and recall lowered as expected, but the reconstructed point sets have shown a graceful degradation, as desired, very satisfactory regarding the magnitude of contour offsets applied – between 0.5% and 2% of the image's size. The behavior was in general consistent throughout different scenes and error conditions, and the safe default setting or ranges can be used.

The tests with real scenes followed, to show the method's applicability to real scenarios of camera calibration errors, image noise, faulty contour detection and occlusions. The results were quite satisfactory: occlusion, highlights, lighting variations, apparent contours and other issues were robustly dealt with in most cases, and the reconstructed point sets have shown very good localization, and a very reduced number of outliers. It is important to notice that the good results obtained in troublesome areas as the label of the speaker, or the lines on the hull of the boat; or in planar areas as the calibration pattern, where obtained without any regularization whatsoever.

The overall visual and perceptual results were in general quite good, both in synthetic and real scenes. This can be assessed by the multiple images and models resulting from the reconstructions, and provided in this thesis. Such an assessment is just as important as the quantitative results provided by the proposed metric, as on one hand it is difficult to determine or define some perceptual errors quantitatively (despite the efforts to do so), and on the other hand, some quantitative errors may pass unnoticed perceptually. The good results in both assessments represent an important confirmation of the expectations created by the results with synthetic scenes.

To assess performance, three types of tests were carried out. The first served to assess the advantages of rendering multiple lines in the same buffer, to reduce the amount of data transfer between the GC and the main memory, and showed gains of 43% in the reconstruction time. The other two tests compare the hardware-based implementation to an equivalent CPU-based. On one test, the number of candidates per reconstructed point was varied, to show the gains of processing large volume of data on the GC versus the CPU. In this case, results have shown that a gain of up to 54 times in terms of processing time is possible when large sets of candidates are used, and that, when smaller sets are used, it is possible e.g. in the boat scene to perform

feature point reconstruction using all 39 images as reference in less than half a minute. In the last test, both implementations were compared between the test system used in the other tests, and a system with an older GC – both with consumer-grade CPU's and GC's graphics cards. It was concluded that the hardware-accelerated implementation could run unmodified even in older graphic cards, and still provide a speed-up of up to 4 times, when compared to the CPU version.

These results have shown that it is possible to robustly reconstruct points based on contours at interactive rates.

Chapter 6

Line reconstruction

6.1 Introduction

Previous chapters were devoted to the reconstruction of points based on contour lines. Robust results have been achieved in this context, but improvements are possible in terms of scene representation. A natural extension to point reconstruction based on contour lines is line reconstruction. In terms of scene representation, several advantages arise from line reconstruction:

- It is a higher level representation, that better describes the continuity of lines in the scene;
- It is a more compact representation, providing benefits in storage, transmission and rendering;
- It can be used to infer and complete some missing information (e.g. gaps in contour lines).

This chapter proposes a line reconstruction algorithm that uses as inputs the robust point information obtained using the point reconstruction method presented in Chapter 3, and the connectivity information between reconstructed points provided by contours.

This two-stage approach to line reconstruction – first robustly reconstructing points, and then reconstructing lines based on the points and their connectivity – has some important properties. By robustly reconstructing points in the first place, line reconstruction will be initialized by a set of points with good quality, thus enforcing a degree of line quality. The degrees of freedom in reconstructing a line are constrained by the reconstructed points and their connectivity: only lines connecting reconstructed points are considered. Another benefit from a two-stage approach is the ability to identify and deal with problems by tuning the point reconstruction to improve line reconstruction. If lines were reconstructed directly, it could be more difficult to identify and deal with any problems of incorrectly reconstructed lines.

The line reconstruction algorithm itself relies on the same metric, and on a similar voting strategy – voting based on closeness – as the ones used on the point reconstruction algorithm. By sharing the same concepts, it inherits benefits such as robustness and hardware acceleration capability. The use of the same entity-independent metric also allows the re-use of the same vote maps that are pre-computed for point reconstruction.

Two variants for line reconstruction based on reconstructed points will be presented in this chapter. Variant A produces candidate lines by extracting connectivity information between the projections of merged points in contour maps. This variant requires *a priori* point merging. Variant B is incremental. Each iteration uses one image as the reference for reconstructing a set of points. Candidate 3D lines are generated from the connections between the projections of the reconstructed points. The lines are evaluated, and those that are classified as valid are added to the set of valid lines from previous iterations. In the following iteration, that set of valid lines is used to reduce the number of points to reconstruct from the new reference image. As the process iterates, the set of reconstructed lines increases in number and coverage of the scene, thus decreasing the number of points and lines left to reconstruct. In both variants, each line is either accepted or rejected based on the votes gathered by points on the line and those votes' distribution over the line. The voting procedure applied to points of the line is similar to the one used for candidates in point reconstruction, and uses the same vote maps generated for point reconstruction.

The two variants will be described in detail in Section 6.2 and Section 6.3. Some results of tests performed with implementations of both variants are presented in Section 6.4, and an overview of differences between the presented method and related line reconstruction methods is presented in Section 6.5. The chapter is concluded in Section 6.6 with some final remarks on the method.

6.2 Variant A: Connecting merged points

Contours provide connectivity information between points, which in turn can be used to define lines between reconstructed points. However, if connectivity is extracted independently from different reference images and later joined together, bundles of lines representing the same real line will exist. Incorrect lines resulting from incorrect connectivity observed only in a small number of images may also occur.

The problem of bundles of lines is greatly reduced by merging reconstructed points and extracting connectivity only between merged points, as the lines are implicitly merged through their merged end points. The process of extracting connectivity between merged points can be roughly described as for every image, projecting the merged points in the corresponding contour map, and determine which pairs of merged points are connected by contour lines (Figure 6.1). This would provide a graph containing every connection that was observed in any given image.

In this process, some points that are not actually connected in the scene may be connected by contours in an image due to e.g. coincidental alignment of surfaces in that image. This leads to the establishment of incorrect connections, that should be

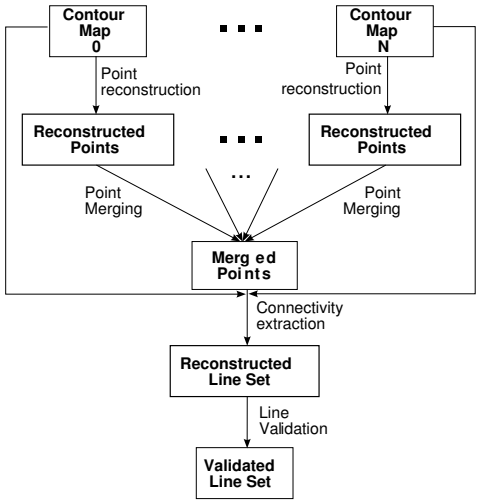


Figure 6.1: *Flow chart of variant A: line reconstruction based on merged points.*

identified and discarded. Robustness to incorrect connectivity can be improved by applying to the points of a candidate line a voting procedure similar to the one used in the point reconstruction. Analyzing the vote distribution along the points of the candidate lines, it is possible to identify lines that gather sufficient evidence of their existence from other images, and reject other lines as invalid.

The process briefly described here is shown in Figure 6.2, and analyzed in detail next.

Project merged points in a contour map

Assume that points were reconstructed from all the images, and that point merging also took place (Figure 6.2(a)).

To extract connectivity information, it is first necessary to project the merged points into contour maps. This determines which contour points will serve as starting points for the extraction of connectivity.

Each merged point that is projected on a contour map (Figure 6.2(b)) may be projected outside a contour, due to small offsets caused by merging, camera calibration or errors in the contours. This leads to lost connectivity.

To account for these small offsets, each projected merged point is associated to the closest contour point – *assigned point* – within a small neighborhood (e.g. 3x3). If there is no contour point in that range, the merged point is considered isolated i.e. it has no connectivity in that contour map. An example of such a situation can be seen in the leftmost contour map of Figure 6.2(c), where the lower left corner of the square is isolated. This does not imply, however, that it will not have connections in other maps: despite of a point being isolated on one image due to contour offsets or occlusion, it may be visible and on the correct contour in other images.

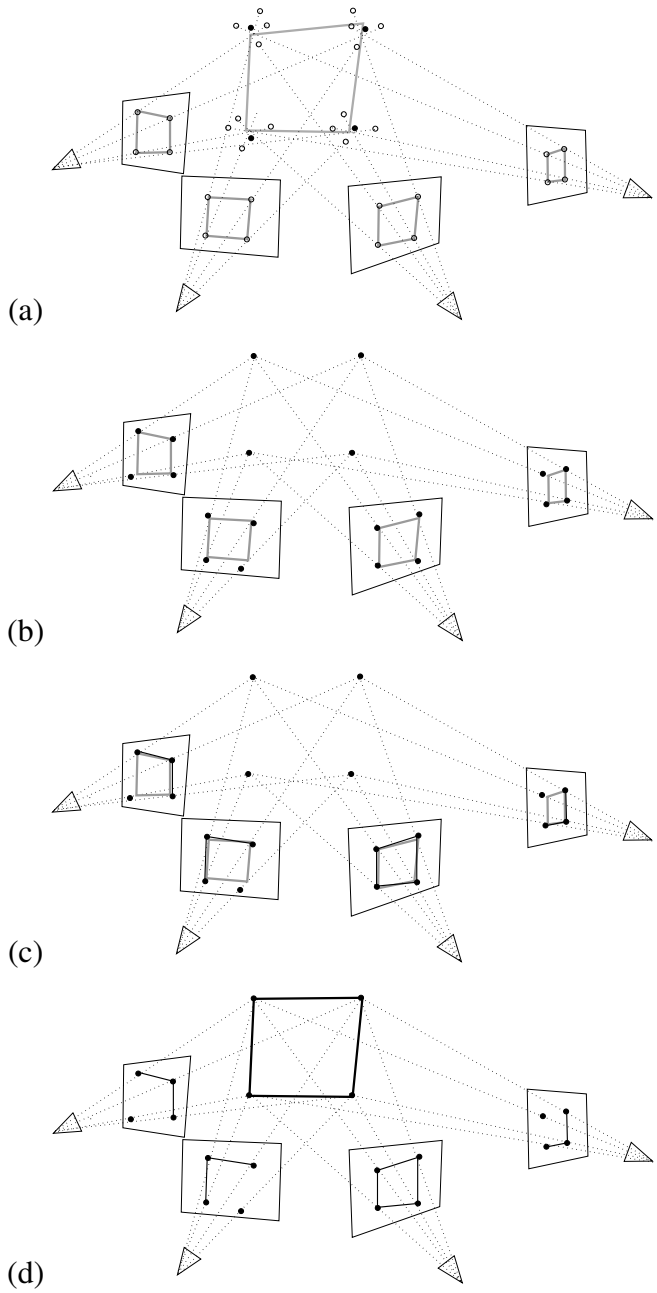


Figure 6.2: *Steps for connecting merged points*.

(a) Reconstructed points (white dots) are merged (black dots) (original object in grey)

(b) Merged points are reprojected in the contour maps (contour lines marked as grey)

(c) Connections (black lines) are established between projected points, using contour information

(d) Connections established between projected points are associated to the corresponding merged points, generating the candidate 3D lines

A merged point may also project on the wrong contour point in an image, resulting in incorrect connectivity. In this case, one relies on a line validation procedure (described later) to identify and reject such incorrect connections

Extract connectivity information

The connectivity between merged points that have been assigned can be extracted by contour following, in a similar way to the one described in Section 3.5.2 for line simplification.

In the process of feature extraction by line simplification, described in Section 3.5.2, topological features are identified, and the paths connecting them are extracted by following contour lines until the next topological feature is found. Those paths are simplified by extracting corners and dividing the path into straight line segments, delimited by feature points (junctions, extremities or corners).

In the case of merged points, the connections are established between merged points, instead of topological features. In this case, however, it may happen that two merged points are connected by a curved contour path. Representing such a connection by a straight line may be a very coarse approximation, thus yielding an incorrect connection. Since at this stage the only points that can be used are the merged points, simplifying that path does not help, as there will be no other merged points in the possible corners of that path. Therefore, straight-line connections that have at least one point whose distance to a contour is above a threshold (equivalent to the threshold used in line simplification) are discarded and not considered for line validation.

Each connection between two merged points $\mathbf{m}_a$ and $\mathbf{m}_b$ that is consistent with a straight contour is stored in a *connectivity graph*. After extracting connectivity from all images, the connectivity graph will contain a connection for each pair of merged points that were connected by a contour line in at least one image (Figure 6.2(d)). Each connection represents a candidate 3D line. The number of images in which a given connection is observed i.e., in which the projections of the same pair of merged points are directly connected by a contour line, is also stored for validation (as described next).

Validate lines

Line validation is performed using a voting procedure similar to the one used for point reconstruction.

Each line from the graph is voted using the vote maps from all images (the same used for point reconstruction). While in point reconstruction the vote maps were projected over a segment of a view ray, in line reconstruction they are projected over the candidate line. The result is an accumulation of votes distributed along the line (Figure 6.3), representing how close the points of the candidate line project to contours in the various images.

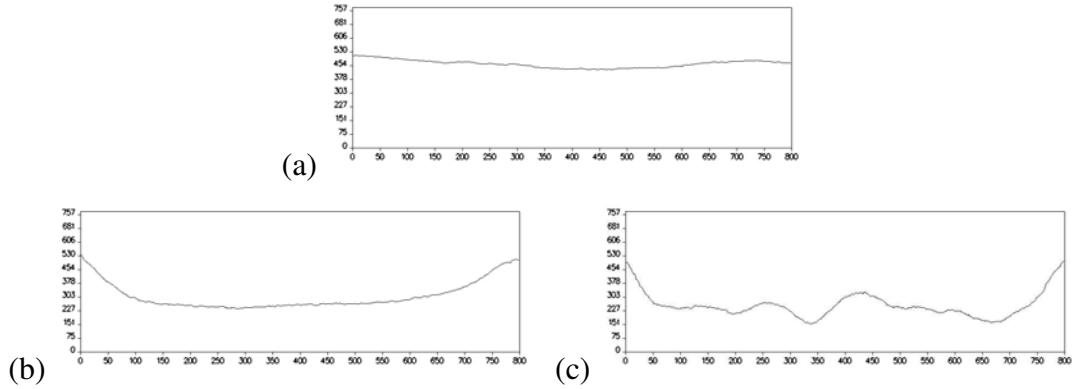


Figure 6.3: *Accumulation of votes on a candidate 3D line*.

(a) A correct line has a consistent high voting

(b) An incorrect line has in general high voting close to its endings (the original reconstructed points) and low voting in the remaining points

(c) In some cases, parts of an incorrect line may have some local maxima, albeit low, due to the intersection of the projection of the incorrect line with contours in some images

Some heuristics are used to determine if the line is valid or not, including:

Minimum image count Given the number of m-points, it is possible that pairs of points on opposing surfaces are incorrectly connected on a given image. In scenes with high contour density, these lines can collect a high number of votes due to cross-voting. Experiments have shown that requiring connections to be observed in a minimum number of images t_{cnt} efficiently filters this type of incorrect lines.

Minimum average number of votes per sample The vote average t_{avg} is an indication of the existence of the line. Lines that overlap contours in most images will have a very high vote average. Lines that are badly estimated are expected to have low vote average, since a large number of their points will be distant from contours in the majority of the images, and therefore receive few votes. Therefore, lines with a vote average below a threshold t_{avg} should be rejected.

Maximum standard deviation of votes The maximum standard deviation of votes t_{std} serves to guarantee the quality of the connectivity. Using only the average would allow that a line with an actual gap would still be accepted as valid, if the majority of the votes was high, and only a small part of the line had very low voting. By imposing a maximum in the standard deviation, together with the minimum average, such cases can be identified and the corresponding lines rejected.

The process is described in Algorithm 6.1.

In the case of a contour line missing only a few contour points, most parts of the line will still accumulate votes in an image. If the gap only occurs in some

- For each line L_m in the connectivity graph, with an image count greater or equal to t_{cnt}
  - Sample L_m as a set of n 3D points $l_{m,k}$
  - For each image I^i
    - For each $l_{m,k}$ collect the votes λ_{m_k}
  - Determine the line's validity
    - $\overline{\lambda_m} = \frac{\sum \lambda_{m_k}}{n}$
    - If $\overline{\lambda_m} < t_{avg}$ or $\sqrt{\frac{\sum \left(\lambda_{m_k} - \overline{\lambda_m}\right)^2}{n}} > t_{std}$ then reject L_m

Algorithm 6.1: Line validation using distance transforms

images, it should not influence the standard deviation significantly, as votes from other images will compensate for it. This means a reduced sensitivity to small sets of missing contour points.

On the other hand, if the real line is not straight (and therefore most of the associated contour lines are not, either) then the standard deviation of votes will be high. This is usually the case when a 3D curved or piecewise straight line projects as a single straight line.

Line validation provides a high level of confidence in the final line set. It reduces both the number of rejections due to small gaps in contours, and the number of connections incorrectly established as straight lines.

The use of the same vote maps and voting procedure in both point and line reconstruction provides a performance benefit. Although line reconstruction requires the extraction of connectivity information for the merged points, there is no need to recompute contour or vote maps, or transformation matrices for projective texturing. All of these were already computed for the point reconstruction procedure.

The tests and results obtained with this variant will be presented in Section 6.4.3.

6.3 Variant B: Incremental line reconstruction

The method presented in the previous section requires reconstruction and merging of a representative set of points from all images, and may introduce problems of connectivity loss, when trying to extract connectivity for merged points that are projected outside contours.

In this section, a different approach is proposed – *incremental line reconstruction*. Instead of requiring the merging of points reconstructed from multiple reference images, it is proposed to perform the line reconstruction iteratively. The main idea behind it is to process one reference image at a time, performing feature point extraction and reconstruction as described in Chapter 3. Each 3D line that is reconstructed and validated in that reference image, it is "removed" from the contours of other reference images, so that it is not reconstructed again (Figure 6.4).

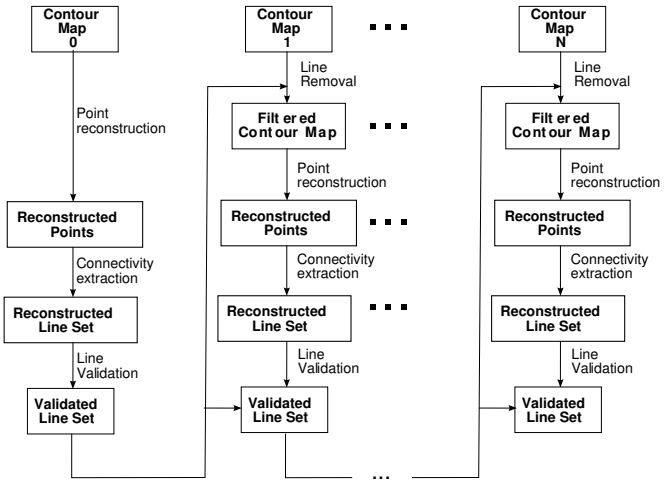


Figure 6.4: *Flow chart of variant B: incremental line reconstruction.*

The removal of reconstructed lines reduces the chance of a line being reconstructed multiple times, with different small errors. Without removal, the reconstruction would contain bundles of lines that would clutter the results and require merging.

Line removal is also a means of reducing the number of point and line reconstructions required. At each iteration, more valid 3D lines are added, and therefore more contours are "removed" in the following iteration, meaning less points and lines to reconstruct. In the limit, after some iterations, all lines will have been reconstructed, and therefore no contours remain in the contour maps.

Two of the components of this variant were already discussed for variant A:

- The generation of 3D line candidates is based on the contour information, similarly to variant A. However, in this case the problem is simplified: while in variant A there were problems projecting back merged points, the points being connected in variant B were extracted directly from the contour map, so their correct position in the contour map is known. Furthermore, connectivity information is implicit in the extraction of feature points, and there is no need to do it a second time.

- The process and heuristics used for evaluating the candidate lines is very similar to the one used in variant A. The only difference is that the threshold on the number of images t_{cnt} is not used, as in this case the points and their connections are all extracted from the same image. The benefit of using the same vote maps and transformation matrices as point reconstruction, observed in variant A, still applies.

The other key process of this variant is the "removal", between each iteration, of previously reconstructed lines from the contour map used for feature point and connectivity extraction of the next reference image. The reconstructed lines are projected on the contour map, and the contours overlapped by them are filtered out. The result is a *filtered contour map*, that contains only non-reconstructed contours. It is from this

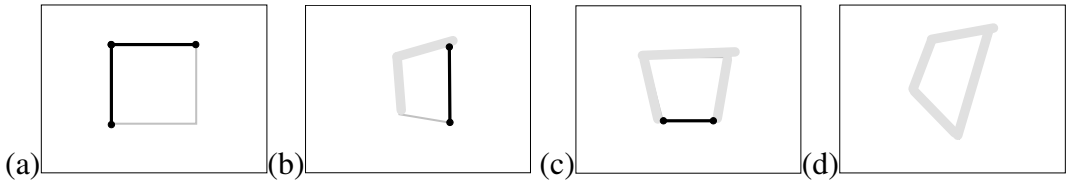


Figure 6.5: *Incremental line reconstruction*.

(a) Two lines are reconstructed (black) from the four contour lines (thin gray lines) possible; other lines missed due to e.g. incorrect reconstruction of the missing corner

(b) The two reconstructed lines are projected in the second image (thick gray), to remove corresponding contour lines; From the remaining contour lines (thin gray) only one was reconstructed (black)

(c) The three previously reconstructed lines are projected in the third image; this time the bottom line was reconstructed

(d) Lines previously reconstructed cover all contours, leaving no points or lines to reconstruct in this iteration

map that feature points and their connectivity will be extracted. This process is now described in detail.

Assume that a set S of lines has been reconstructed from previous iterations of the algorithm. To avoid reconstructing these lines again, they should be removed from the contour map. In this way, the connectivity between the corresponding end points of a line will not be present in the contour map, and therefore the line will not be reconstructed again (Figure 6.5).

Removing such lines can be performed by projecting the 3D line set over the contour map, marking the corresponding points as empty (not a contour). As it happens in other processes presented in this thesis, one has to account for possible offsets caused by errors in contour detection or camera calibration. For this reason, not only the points corresponding to the line projection are removed, but also points that are in a close neighborhood. In practice, this is achieved by rendering the lines in the map with a width corresponding to the size of the neighborhood that should be removed.

The process of line reconstruction may then continue as in variant A. After producing a set of valid lines for this reference camera, those lines should be added to S, so that they are included in the filtering of the contour map of the next reference camera.

When all cameras are processed, S corresponds to the final set of lines.

In order to avoid accepting incorrectly reconstructed lines from the beginning, which would prevent the correct lines to be reconstructed from other cameras, one can use conservative values for the parameters of line validation. That enforces that only the best lines survive and are used for filtering. If the parameters are too strict such that some lines are not reconstructed after using all images as reference, a second pass may be performed with more relaxed parameters, and starting with the reconstructed line set obtained in the first pass.

The incremental line reconstruction offers increased performance over variant

- $S = \emptyset$
- For each reference image I^i
  - Project lines of S on contour map for removing contours reconstructed before
  - Extract feature points on remaining contours as described in Section 3.5.2, and keep connectivity information
  - Reconstruct extracted feature points using e.g. Algorithm 3.4
  - Create a connectivity graph with the connections between valid reconstructed points
  - Validate lines in the connectivity graph using Algorithm 6.1
  - Add valid lines to S

Algorithm 6.2: Incremental line reconstruction

A. However, there are some issues associated with variant B that deserve mention, namely:

- The possibility of incorrectly removing lines from contour maps, due to proximity to a reconstructed line's projection. If two contour lines are close to each other, removing the projection of a reconstructed line and the points in its neighborhood may result in the removal of the other contour line. In practice, a 3D line is projected in various contour maps, so there is a low probability that the same line will be filtered out from all the images, unless they are close in 3D. If this is the case, the outcome is that only one of the lines is reconstructed, which is comparable to merging the two lines due to their closeness.

- The fragmentation of contour lines due to the removal of previously reconstructed lines. If the projection of a line reconstructed in e.g. the first image intersects a contour line on a second image, the line removal procedure will break the contour line in two pieces, whose reconstruction will be attempted independently. This may result in a significant number of fragmented lines in the final reconstruction.

- The order-dependency of the method. The final reconstructions differ if the order in which reference cameras are chosen is different. If line validation is performed with conservative values as suggested, the differences between different camera ordering should be small. A line validated using conservative values has a high degree of confidence, and is consistently represented by contours in multiple images. Unless all contours corresponding to that line are removed by the projections of other lines, the line should be correctly reconstructed from whichever reference camera where it is identified first. Using conservative values, the probability of using incorrectly reconstructed lines that would remove contours corresponding to other 3D lines is low. Therefore, the overall results should be similar, regardless of the order, as long as line validation is used in a strict way, as suggested.

6.4 Tests

The two variants presented in the previous sections were implemented using the projective texturing technique described in Chapter 4. Tests were performed on the five synthetic scenes used in Chapter 5, with both variants, to try to assess the quality of the reconstructions obtained. Below is described the methodology used for testing and assessing quality, the test setup used, and the results obtained with each of the variants. In the end of this section, both variants are compared side by side, including results obtained with the real scenes (speaker and boat).

6.4.1 Quality assessment for line reconstruction

The assessment of errors in the case of line reconstruction is not trivial to define or implement. In point reconstruction, the major issue was to determine the real point associated to an estimated point, so that an error could be computed (Section 5.2.1). In that context, it was decided to choose the real point closest to the reconstructed point, under the constraint that both projected within a small neighborhood of the reference image(s).

The problem of associating a real line to a reconstructed line is of similar nature, with the added difficulties of:

- Defining an error metric between two lines;
- Determining a methodology for assessing the reconstruction quality from that error metric;
- Relating it to the error assessment in point reconstruction.

Considering these factors, it is proposed that the assessment of line reconstruction is performed by sampling the reconstructed lines as a set of points – *point samples* – and to use the same error metric used in point reconstruction. This eliminates the need to define an error metric between two lines. It also allows for the use of a quality assessment methodology similar to the one used on Chapter 5, e.g. determining the number of true and false positive points and evaluating precision, as well as sharing the same error measuring implementation.

It should be noted that the metric proposed here provides at best an approximate indication of the reconstruction quality, which on one hand is consistent with the visual results obtained, and on the other hand provides a quantitative classification of the errors.

Two related issues subsist:

how to choose a reference view for ground truth data In Chapter 5, one of the issues in determining which real point to associate to a reconstructed point was the choice of a reference view. Simply choosing the real point closest to a reconstructed point could lead to incorrect low errors e.g. for points incorrectly

reconstructed on a surface opposed to the correct one. The same problem holds for the point samples used in the case of lines.

how to sample a line to compute the errors of its points different images will provide different point samples for the same line, leading to different density of the resulting point set. This can influence the number of true positives and negatives, and consequently some measures such as precision.

The choice of a reference view is more complex for variant A than for variant B. In variant B, a line and its end points are all extracted from the same image, taken from a given view point. In this case, that view point is the natural choice for the extraction of ground truth data.

This is opposed to the case of variant A. As the end points of a line are merged points in variant A, they may be originated from different images, and end up connected only because they project on the same contour on an image. In this case, there is no obvious choice for a reference view.

Different methods for choosing a reference view for the extraction of ground truth data for variant A were analyzed. It seems natural that such a reference view should be one in which the line is completely observed i.e., one in which the projections of the same pair of merged points are directly connected by a contour line. The relation between the line's direction and the viewing direction should also be taken into account, as choosing a camera whose viewing direction is close to the line's direction may lead to strong foreshortening effects and bias in the error measuring. Choosing an image in which the line is facing the camera as much as possible is therefore beneficial.

The distance to the camera should also be taken to account. Using a camera that is very distant from the line results in a small projection of the line in the image plane, and therefore a low number of point samples.

A method to take these different factors into account is to choose, from the images in which the line was observed, the image in which the line has the largest projection. This provides a balance between the line-viewing direction and the distance to the camera.

6.4.2 Test setup

The same set of scenes presented in Chapter 5 was used to test the parameters that rule line reconstruction. The set of values used for the parameters of point reconstruction and merging were the defaults presented in Section 5.2.3. The set of default values used for the line reconstruction parameters are presented in Table 6.1.

Each variant was tested independently. Each test consisted of varying one of the parameters, and using the default values for the remaining parameters. In each run, a number of lines were collected, reconstructed and validated. The lines considered valid were projected in the corresponding reference view, and sampled into points. The error of each point was computed according to Section 5.2.1. Points with error below the threshold of 0.5% of scene scale were considered true positives, and other points were considered as false positives. Given the difficulty in quantifying the universe of

Parameter	Values tested	Default Value
Minimum image count	{1, 2, 3}	3
Minimum average voting	{60, 80, 100, 120}	80
Maximum standard deviation	{5, 10, 15, 20, 25, 30}	20

Table 6.1: *Parameters tested in line reconstruction.*

lines that could be correctly reconstructed, especially in variant B, a measure of recall would be misleading. Therefore, only precision will be considered in the analysis of the quality of the line reconstruction.

In each of the tests with a different parameter set, the data collected for lines consisted of the number of lines considered valid (vl) and the total number of lines evaluated (NL). The valid lines were sampled into points as described above, and the following data was collected for those samples: number of false positive (fp) points, number of true positive (tp) points, total number of estimated positive (EP) points and the precision value (PR), based on those numbers. It should be stressed that these measures based on point samples are provided only as an indication of the quality of the reconstruction. The uncertainty in measuring these points does not allow a definitive conclusion, but serve to show that in principle there is no significant loss of precision between line reconstructions and the corresponding point reconstructions.

The results obtained will be presented and analyzed in the following subsections.

6.4.3 Variant A

Minimum image count

In line reconstruction based on merged points, there is a high probability that points from opposite surfaces appear to be connected by contours in one or two images. This is especially noticeable in cases of high density of contours. By imposing a requirement on the minimum number of images in which the projections of a pair of merged points are directly connected by a contour line, it is possible to eliminate many of the lines corresponding to those spurious connections.

Figure 6.6 shows the effect of filtering lines based on that image count, in the bunny and carriage scenes. It can be seen that the influence in the bunny is very significant, as the high contour density leads to a very high number of crossed connections. Just by requiring a minimum of two images (middle row of Figure 6.6) the filtering is very significant. At a minimum number of three images, the number of lines that cross the bunny interior is already quite low. In the carriage scene, the same effect is visible, but at a smaller scale. The effect is concentrated in areas where the contour density is higher, such as the center of the wheels, the meeting point of the rooftop's supporting bars (above the wheels) and the bars in the front of the carriage.

The values shown in the charts and tables of Figure 6.7 confirm these observations: scenes with higher contour density, namely the arcade and the bunny, obtain a significant increase in precision when the minimum number of images is increased, es-

pecially from the value of 1 to 2. In the other three scenes, the effect in precision is less noticeable, as the number of crossed connections is lower, and in general correspond to relatively small lines.

By accepting only lines that are observed in a minimum of three images, one effectively filters most of the incorrect connections caused by high contour density.

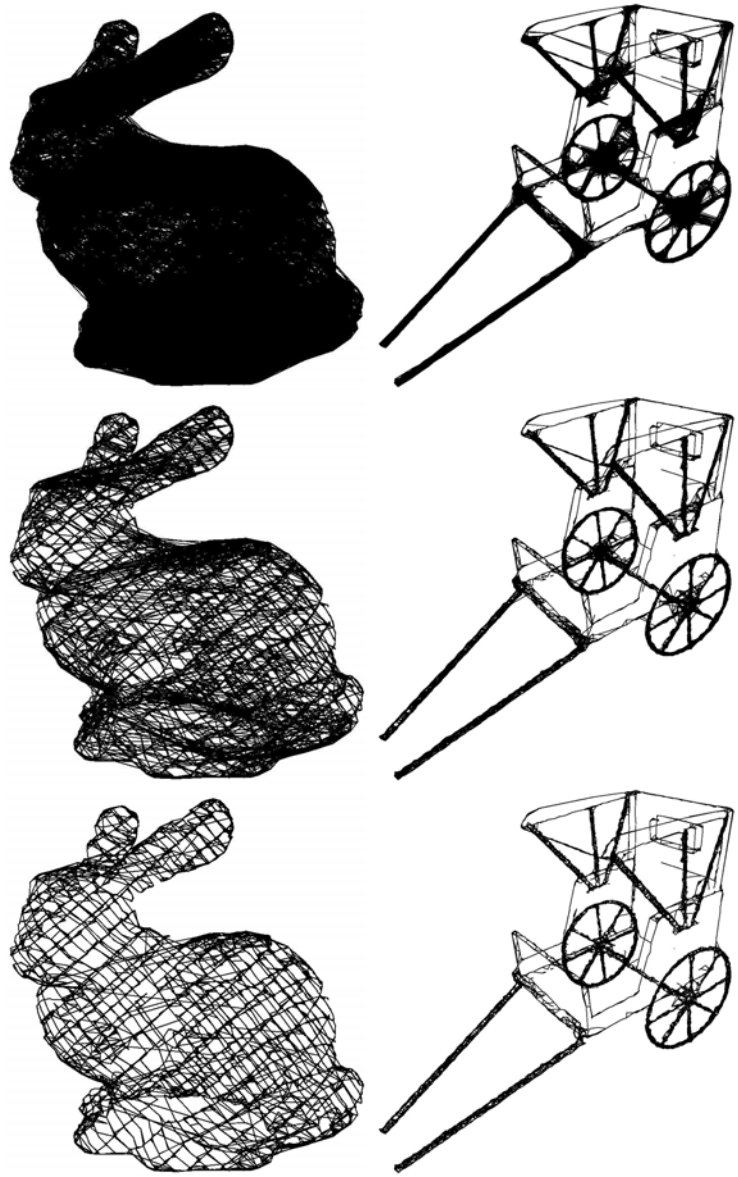
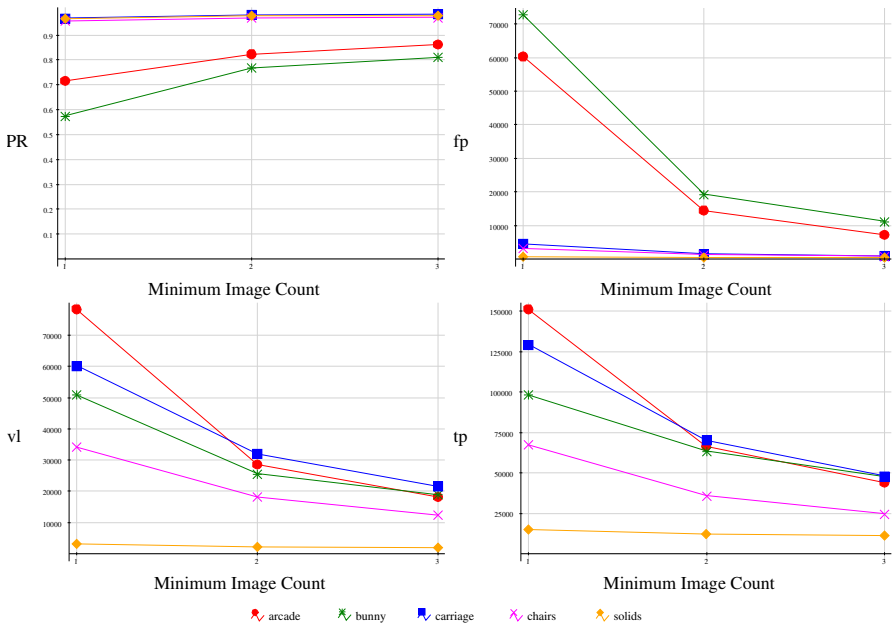


Figure 6.6: *Variant A: Minimum image count tests (reconstruction views).*
Line reconstruction using variant A and a minimum image count of 1, 2 and 3 (top to bottom)



arcade						
Min. Image Count	vl	NL	fp	tp	EP	PR
1	**78242**	112345	**60178**	**150970**	**211148**	*0.715*
2	28434	112345	14377	66257	80634	0.822
3	*18044*	112345	*7084*	*43935*	*51019*	**0.861**

bunny						
Min. Image Count	vl	NL	fp	tp	EP	PR
1	**50821**	99254	**72777**	**97980**	**170757**	*0.574*
2	25514	99254	19196	63238	82434	0.767
3	*18785*	99254	*11115*	*47457*	*58572*	**0.81**

carriage						
Min. Image Count	vl	NL	fp	tp	EP	PR
1	**60183**	81756	**4405**	**129163**	**133568**	*0.967*
2	31917	81756	1403	70091	71494	0.980
3	*21509*	81756	*793*	*47762*	*48555*	**0.984**

chairs						
Min. Image Count	vl	NL	fp	tp	EP	PR
1	**34104**	43609	**3125**	**67192**	**70317**	*0.956*
2	18047	43609	1187	35723	36910	0.968
3	*12308*	43609	*765*	*24358*	*25123*	**0.97**

solids						
Min. Image Count	vl	NL	fp	tp	EP	PR
1	**2944**	7117	**560**	**14765**	**15325**	*0.963*
2	2116	7117	286	12085	12371	**0.977**
3	*1804*	7117	*253*	*10858*	*11111*	**0.977**

Figure 6.7: *Variant A: Minimum image count tests (data charts).*

Minimum average voting

The average of votes on the points of a line is an indication of how that line is projected close to contours. A correct line should have a high average i.e., it should project very close (or over) contours in a significant number of images. Conversely, an incorrect line is likely to be distant of contours in a significant number of images, thus collecting a low vote average. By requiring a minimum average voting to accept a line as valid, one defines a desired level of confidence in the lines. High values for this parameter are more conservative, meaning that setting it too high may lead to excessive rejection of lines.

Figure 6.8 shows the variations in reconstruction results for the arcade and the solids scene. It is visible in the arcade scene that, for the more liberal minimum average voting value of 60 votes, some incorrect lines crossing distant parts of the scene get accepted. As the threshold value is increased, those incorrect lines are rejected, but also some correct lines are removed. These missing correct lines are typically those that are less visible throughout the scene, namely the base of the second step and the pillars that are occluded in images taken from below the arcade, and the upper lines on the rooftop. For very high values of the parameter, only very few lines (the strongest voted) subsist.

In the solids scene, a similar effect occurs: parts that are not visible in many images, such as the corner of the box in the middle (that is occluded in many images by the other solids) or the top of the prism (that is out-of-frustum in some images), are progressively rejected as more conservative values of the parameter are used. In the more liberal values, some incorrect lines are visible in some corners of the solids. In this scene they are less evident, as the contour density is not as high in the arcade, and thus does not promote cross-voting as much as the arcade.

In terms of quantitative results, Figure 6.9 shows that, although the variations in precision are relatively small (at least for values below 120 votes), the variations in the number of lines and points is significant. In general, going from a minimum average of 60 votes to 80 votes causes only a slight decrease in the number of valid lines, with also a small increase in precision in most cases (being the solids the exception, by a very small amount). However, higher values for the minimum average voting cause a strong decrease in the number of lines and points, leading to a significant reduction in the coverage of the scene (as confirmed by the reconstructions in Figure 6.8). This parameter should not be used at very conservative values, especially in cases where the contour density is not high.

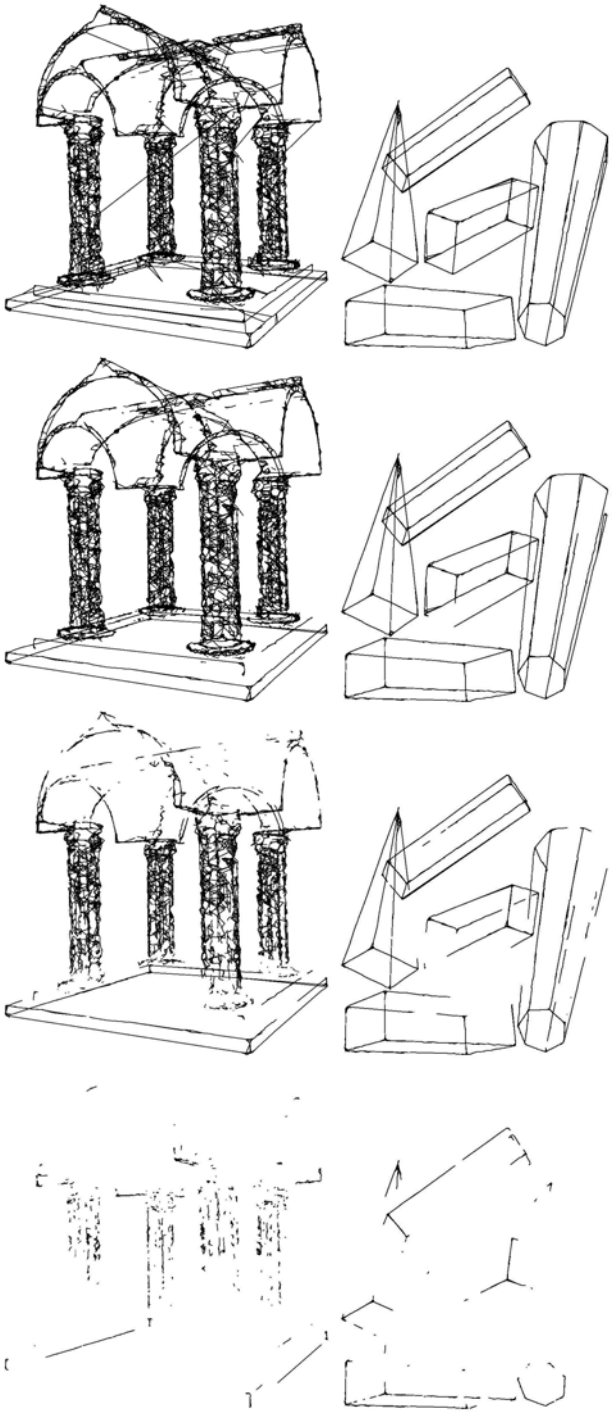
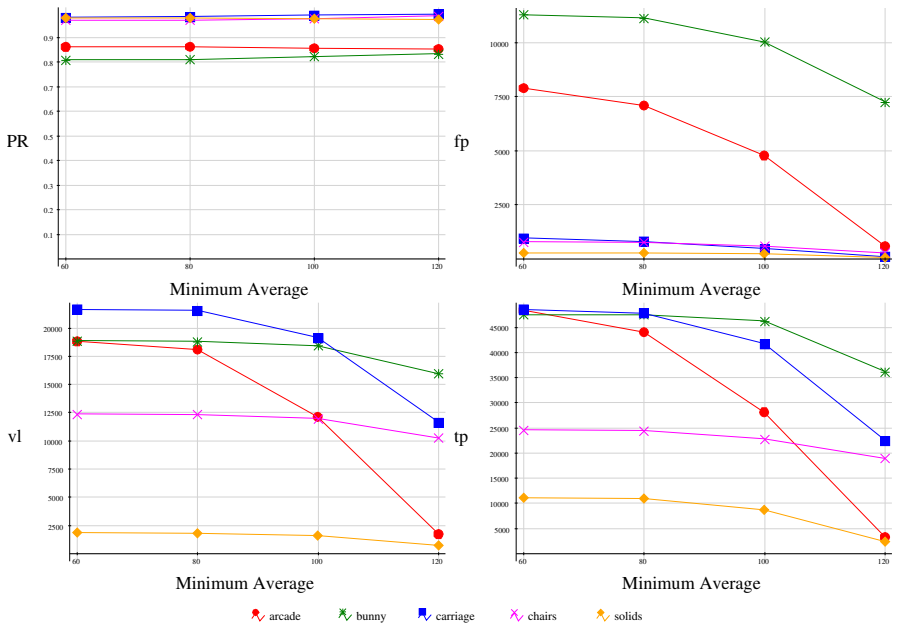


Figure 6.8: *Variant A: Minimum average voting tests (reconstruction views).*
Line reconstruction using variant A and a minimum average threshold of 60, 80, 100 and 120 votes (top to bottom)



arcade						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**18763**	112345	**7889**	**48281**	**56170**	0.86
80.0	18044	112345	7084	43935	51019	**0.861**
100.0	12094	112345	4754	28103	32857	0.855
120.0	*1683*	112345	*562*	*3227*	*3789*	*0.852*

bunny						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**18837**	99254	**11287**	**47492**	**58779**	*0.808*
80.0	18785	99254	11115	47457	58572	0.81
100.0	18401	99254	10018	46188	56206	0.822
120.0	*15930*	99254	*7244*	*36106*	*43350*	**0.833**

carriage						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**21643**	81756	**959**	**48546**	**49505**	*0.981*
80.0	21509	81756	793	47762	48555	0.984
100.0	19106	81756	456	41639	42095	0.989
120.0	*11616*	81756	*102*	*22442*	*22544*	**0.995**

chairs						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**12337**	43609	**777**	**24519**	**25296**	*0.969*
80.0	12308	43609	765	24358	25123	0.97
100.0	11939	43609	578	22759	23337	0.975
120.0	*10244*	43609	*265*	*18891*	*19156*	**0.986**

solids						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**1818**	7117	**265**	**11084**	**11349**	**0.977**
80.0	1804	7117	253	10858	11111	**0.977**
100.0	1591	7117	223	8698	8921	0.975
120.0	*711*	7117	*64*	*2327*	*2391*	*0.973*

Figure 6.9: *Variant A: Minimum average voting tests (data charts).*

Maximum standard deviation

The standard deviation of the votes over a given line provides an indication of the votes' distribution over the line. High standard deviation indicates that there are both very high and very low voting in different parts of the line, i.e. that there are inconsistencies between images for the line. Such lines, even if having a high average, should be considered carefully, as it is possible that an incorrect segment of the line is being compensated by a very good one. Setting a maximum value for the standard deviation allows therefore to filter lines that, despite having an acceptable average voting, may have segments with very low voting.

Figure 6.10 shows the effects of varying the maximum standard deviation threshold on the arcade and the bunny scene. It is expected that relaxing the parameter to allow higher values of standard deviation should increase the number of lines accepted, but at the expense of also increasing the number of incorrect lines. In the arcade scene, that variation is visible when moving between the first and the second level shown, but not so noticeable between the second and the third. In the bunny scene, the differences between the second and the third level are more noticeable.

By analyzing the charts in Figure 6.11, it is noticeable that the highest variation in number of valid lines occurs between the maximum standard deviation values of 5 and 10. This indicates that most lines have a standard deviation below 10, considering the default minimum average voting value of 80 votes. Only in the bunny scene there is a noticeable variation for threshold values above 10.

Those lines that have a standard deviation with values superior to 10 correspond to two types of line sets, as can be seen in the two last bunny reconstructions in Figure 6.10. One is a relatively small number of long lines crossing the bunny, which have a high number of samples with error value high above the threshold. These are bound to have a high concentration of votes close to the extremities, but in their center the voting distribution can be quite irregular, thus leading to the high standard deviation. The other type of line sets corresponds to the high number of small lines connecting close-by checkers on the curved surfaces of the bunny, whose majority of samples are slightly above the error threshold in areas of high curvature. In this case, a similar vote distribution occurs – high voting in extremities, irregular voting in the center – but at a smaller scale, and with a low number of samples, thus also leading to a high standard deviation. The contour density and distribution increases the probability of a high image count, thus contributing to the acceptance of such lines. These effects do not occur as strongly in other scenes, due to the different distribution of contours.

In the case of standard deviation, it is recommended to use conservative values for the maximum standard deviation, for scenes with overall high density of contours. For other scenes, those values may be slightly more relaxed.

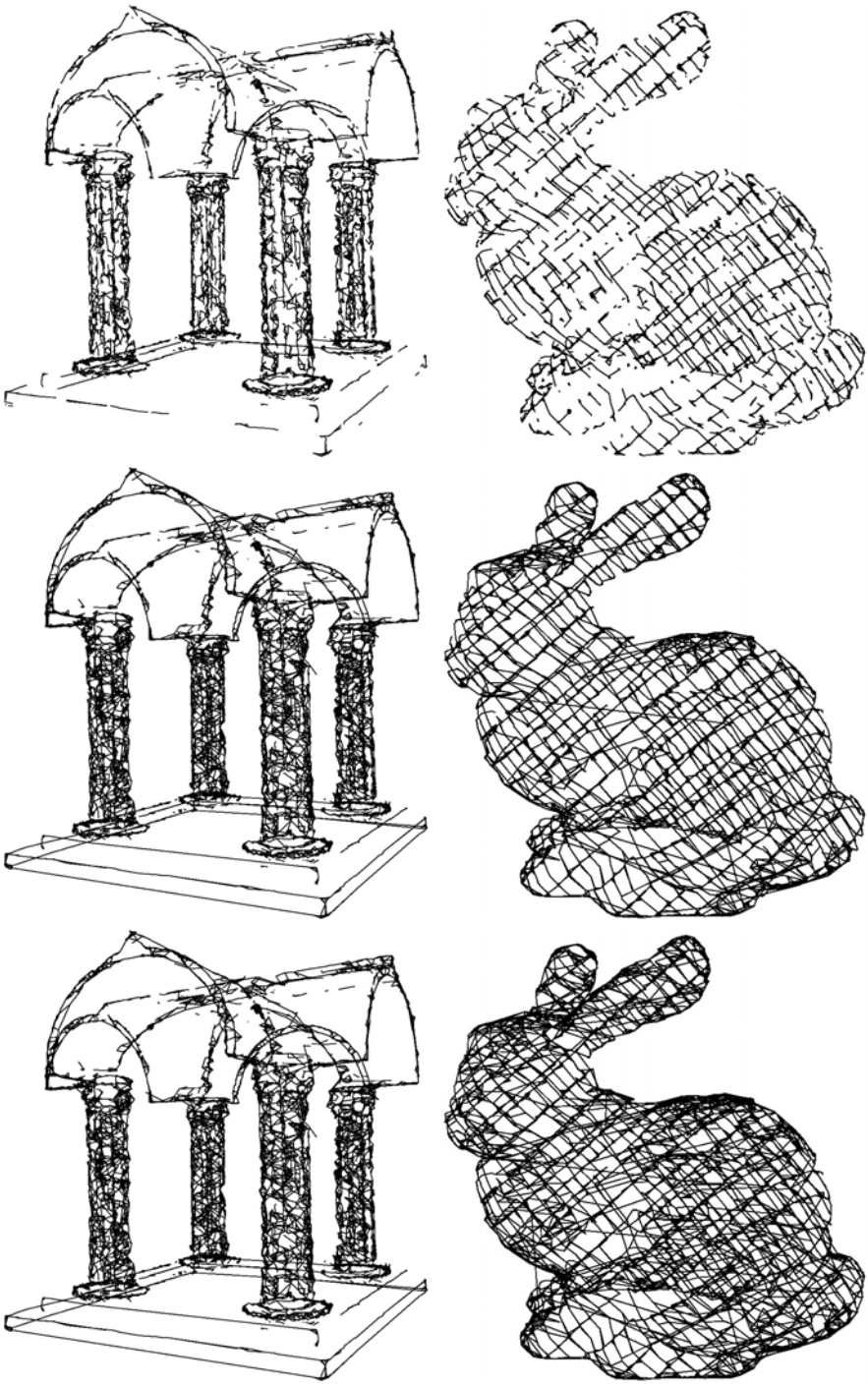
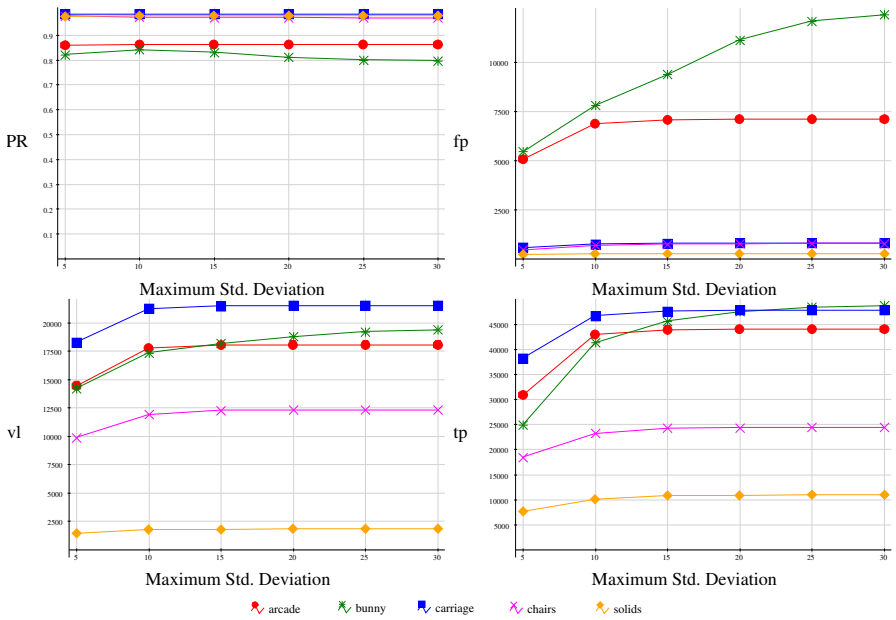


Figure 6.10: *Variant A: Maximum standard deviation tests (reconstruction views).*
Line reconstruction using variant A and a maximum standard deviation of 5, 15 and 25 (top to bottom)



arcade						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	*14456*	112345	*5058*	*30821*	*35879*	*0.859*
10.0	17742	112345	6865	42932	49797	**0.862**
15.0	18027	112345	7056	43876	50932	0.861
20.0	18044	112345	**7084**	43935	51019	0.861
25.0	**18045**	112345	**7084**	**43938**	**51022**	0.861
30.0	**18045**	112345	**7084**	**43938**	**51022**	0.861

bunny						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	*14212*	99254	*5446*	*24956*	*30402*	0.821
10.0	17321	99254	7786	41261	49047	**0.841**
15.0	18174	99254	9368	45606	54974	0.83
20.0	18785	99254	11115	47457	58572	0.81
25.0	19204	99254	12062	48361	60423	0.8
30.0	**19376**	99254	**12406**	**48727**	**61133**	*0.797*

carriage						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	*18275*	81756	*563*	*38224*	*38787*	**0.985**
10.0	21221	81756	735	46685	47420	**0.985**
15.0	21462	81756	775	47571	48346	*0.984*
20.0	21509	81756	793	47762	48555	*0.984*
25.0	21517	81756	**799**	47788	48587	*0.984*
30.0	**21518**	81756	**799**	**47793**	**48592**	*0.984*

chairs						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	*9870*	43609	*456*	*18458*	*18914*	**0.976**
10.0	11911	43609	673	23232	23905	0.972
15.0	12263	43609	744	24238	24982	0.97
20.0	12308	43609	765	24358	25123	0.97
25.0	**12318**	43609	**773**	**24390**	**25163**	*0.969*
30.0	**12318**	43609	**773**	**24390**	**25163**	*0.969*

solids						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	*1425*	7117	*202*	*7550*	*7752*	*0.974*
10.0	1765	7117	246	10098	10344	0.976
15.0	1788	7117	246	10737	10983	**0.978**
20.0	1804	7117	253	10858	11111	0.977
25.0	1810	7117	**255**	10877	11132	0.977
30.0	**1812**	7117	**255**	**10885**	**11140**	0.977

Figure 6.11: *Variant A: Maximum standard deviation tests (data charts).*

6.4.4 Variant B

Variant B presents some differences in the results for the parameters that it has in common with variant A. This is essentially due to the fact that, although the method for evaluating the line is essentially the same (with the exception of the usage of the image count parameter), the method for extracting the lines to evaluate is different. This means that the set of lines that are evaluated is not comparable between variants. The influence of these factors will be shown throughout the section.

Minimum average voting

As in variant A, the use of liberal values for the minimum average voting tests in variant B leads to some noticeably incorrect lines, while too conservative values lead to the excessive rejection of correct lines.

Two examples of these effects can be seen in the reconstruction of the chairs and carriage scenes shown in Figure 6.12. On the top row, one can see some lines connecting the back and the legs of the left chair, and also connecting the two chairs, and in the carriage some spurious lines can also be found. Each new row presents less spurious lines, but also less correct lines.

Similarly to variant A, the charts in Figure 6.13 show that although the variations in precision are relatively small for average values under 120 votes, the variation in the number of lines and points can be quite significant. The number of valid lines decreases with the increase of the minimum average, being the decrease more noticeable between the values of 100 and 120 votes.

There are however some noteworthy differences between variants. In variant A, the number of evaluated links was constant for a given scene, as they were extracted first, and only evaluated after extraction. In variant B, lines are extracted from the first image, evaluated, and valid lines will limit the extraction of the lines on the following frame. Therefore, the complete set of lines that are evaluated depends on the parameters that rule the evaluation of lines. In fact, there is a relation between how conservative the value of a parameter is and the number of evaluated lines. If the parameter is too conservative, only a small number of lines will be accepted as valid on the first image, meaning that many lines will remain to be evaluated in the following image. If a parameter is very liberal, then most lines will be accepted as valid in the first image, resulting in a smaller number of lines left to be reconstructed in the following image. This can be seen in the variation on the total number of lines in Figure 6.13 for a given scene. In almost all cases the number of evaluated lines in variant B is smaller than in variant A, and in most cases the difference is significant, thus bringing some performance benefits. The exceptions occur at the highly conservative value of 120 votes for the minimum average, in which the difference for the arcade is small (112598 to 108218), and in the solids scene variant B even has some more links evaluated than variant A. In both cases the relatively high number of evaluated lines in variant B is justified by the very low number of lines accepted as valid, that cause a very low rate of line removal, and a consequently high number of lines extracted at each image.

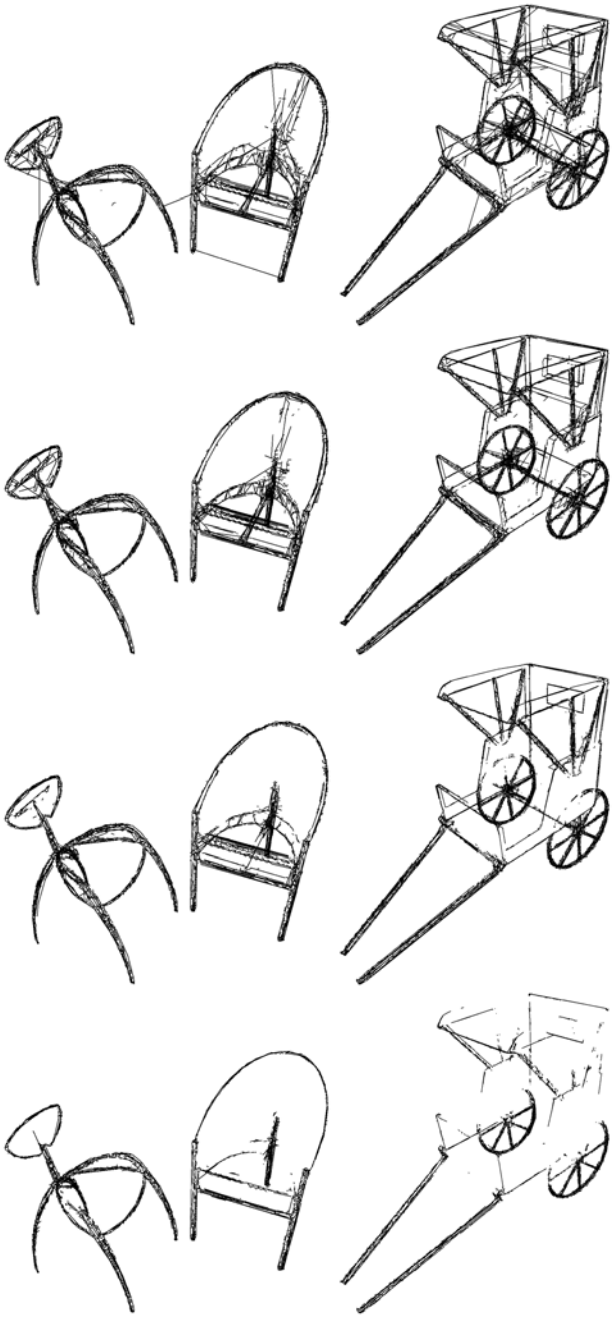
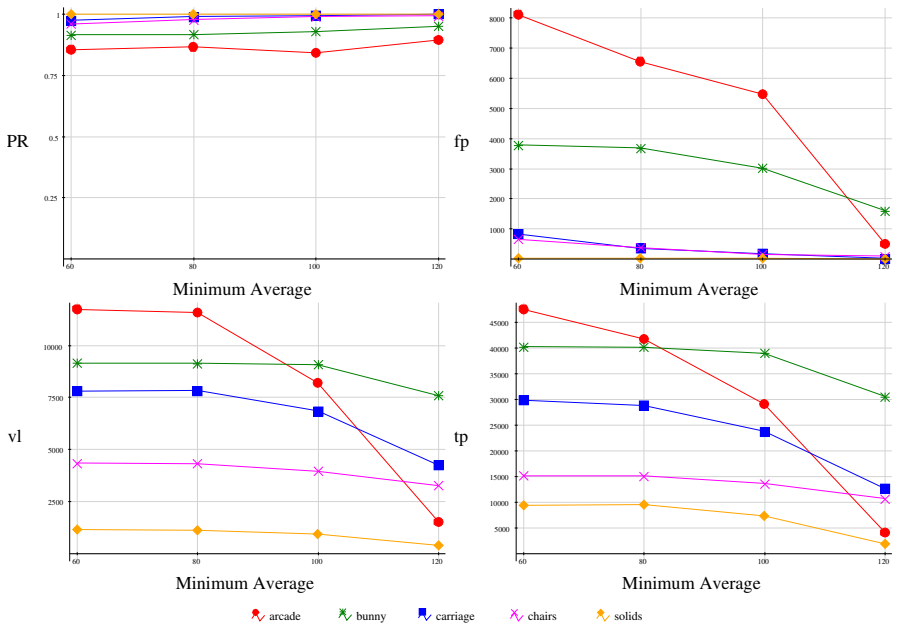


Figure 6.12: *Variant B: Minimum average voting tests (reconstruction views).*
Line reconstruction using variant B and a minimum average threshold of 60, 80, 100 and 120 (top to bottom)



arcade						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**11727**	*23590*	**8110**	**47460**	**55570**	0.854
80.0	11569	29732	6551	41643	48194	0.864
100.0	8180	59475	5467	28983	34450	*0.841*
120.0	*1498*	**108218**	*482*	*4069*	*4551*	**0.894**

bunny						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**9141**	*19671*	**3773**	**40141**	**43914**	*0.914*
80.0	9126	19751	3669	40066	43735	0.916
100.0	9073	21399	3012	38894	41906	0.928
120.0	*7582*	**33162**	*1586*	*30435*	*32021*	**0.95**

carriage						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	7796	*18282*	**818**	**29761**	**30579**	*0.973*
80.0	**7814**	19580	347	28744	29091	0.988
100.0	6843	29355	166	23679	23845	0.993
120.0	*4238*	**56404**	*8*	*12594*	*12602*	**0.999**

chairs						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**4333**	*9116*	**643**	**15087**	**15730**	*0.959*
80.0	4303	9718	372	15040	15412	0.976
100.0	3948	11418	148	13523	13671	0.989
120.0	*3267*	**17267**	*84*	*10597*	*10681*	**0.992**

solids						
Min. Average	vl	NL	fp	tp	EP	PR
60.0	**1135**	3795	**25**	9328	9353	*0.997*
80.0	1107	*3794*	**25**	**9503**	**9528**	*0.997*
100.0	929	4289	19	7284	7303	*0.997*
120.0	*378*	**7340**	*0*	*1768*	*1768*	**1**

Figure 6.13: *Variant B: Minimum average voting tests (data charts).*

Maximum standard deviation

Figure 6.14 shows the results of reconstructing lines with variant B, with different values of the maximum standard deviation parameter. As expected, lower (more conservative) values of the parameter lead to both less incorrect lines and less correct lines, when compared with higher (more liberal) values of the parameter.

Analyzing the charts in Figure 6.15, it can be concluded that the variations in terms of precision are in general low for each scene, with an expected increase of the number of both false and true positives as the value of the parameter increases. The stronger variations in these values occur between the values of maximum standard deviation below 10, with the exception of the bunny scene that still shows some increase above those values (in line with what was observed in variant A).

The number of evaluated lines is substantially smaller than in variant A, for the reasons observed in the previous test. There is however an interesting detail in the number of valid lines: as the parameter value becomes more liberal, the number of valid lines decreases, as opposed to the increase observed in other tests. The relation between the number of valid links and the number of valid points suggests that lower values of standard deviation favor small lines over long lines. In fact, large line segments are prone to partial occlusion, and consequently to fluctuations in voting distribution and higher standard deviation values than smaller lines. Therefore, with a conservative value for the maximum standard deviation of 5 votes, it is natural that, from the few lines that are accepted on a given image, most of them are small. As these small lines extracted from one image are used for contour line removal on the next image, they will fragment the contour lines, giving rise to more small lines to be evaluated (as observed in the previous test), and consequently more small lines to be accepted.

Still, in terms of scene coverage and the actual number of points that are evaluated, more conservative values provide sparser results than more liberal values.

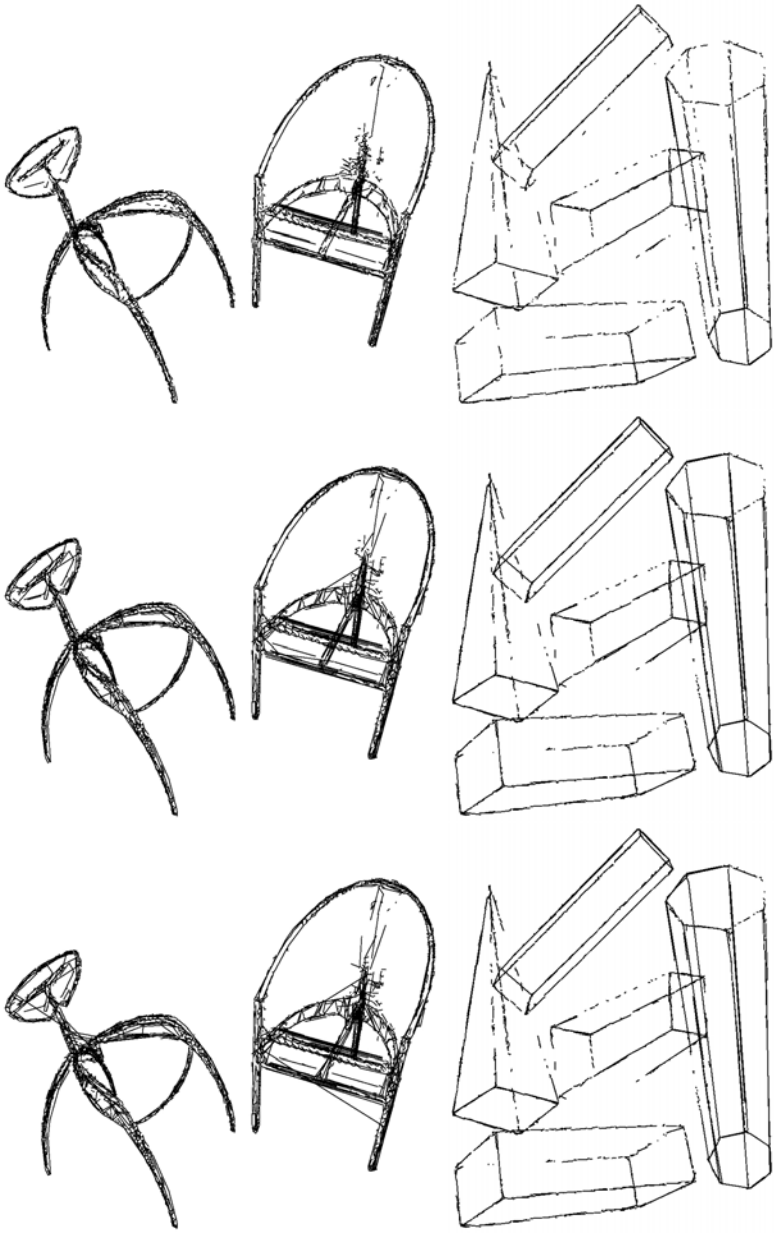
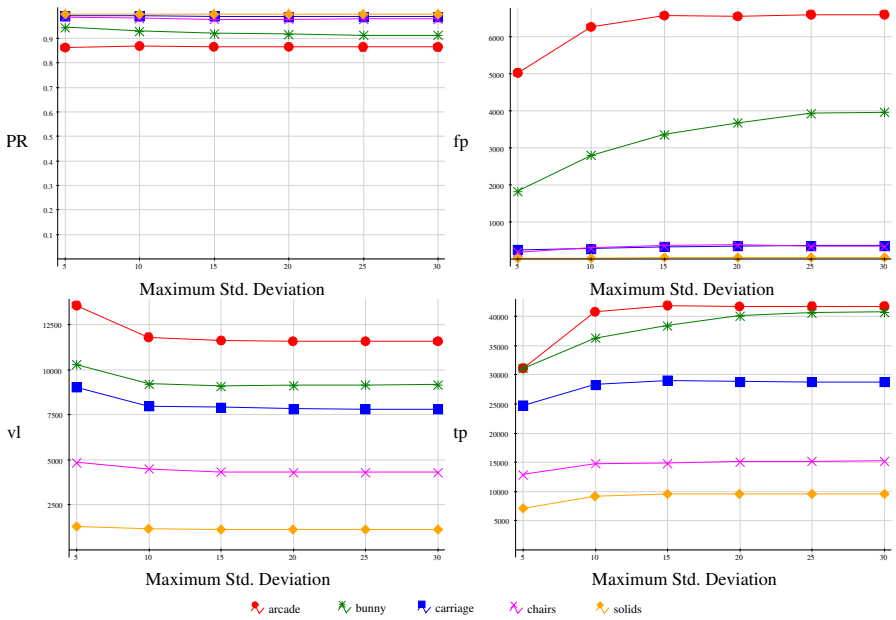


Figure 6.14: *Variant B: Maximum standard deviation tests (reconstruction views).*
Line reconstruction using variant B and a maximum standard deviation of 5, 15 and 25 (top to bottom)



arcade						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	**13568**	**45455**	*5024*	*31057*	*36081*	*0.861*
10.0	11803	31672	6265	40691	46956	**0.867**
15.0	11606	29796	6568	**41814**	**48382**	0.864
20.0	*11569*	29732	6551	41643	48194	0.864
25.0	11570	*29669*	**6597**	41665	48262	0.863
30.0	11570	*29669*	**6597**	41665	48262	0.863

bunny						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	**10276**	**37392**	*1824*	*31028*	*32852*	**0.944**
10.0	9202	23850	2789	36242	39031	0.929
15.0	*9091*	20958	3351	38363	41714	0.92
20.0	9126	19751	3669	40066	43735	0.916
25.0	9140	19362	3925	40547	44472	0.912
30.0	9162	*19202*	**3962**	**40691**	**44653**	*0.911*

carriage						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	**9012**	**26986**	*235*	*24687*	*24922*	**0.991**
10.0	7970	20292	272	28280	28552	0.99
15.0	7898	19731	325	**28943**	**29268**	0.989
20.0	7814	19580	347	28744	29091	*0.988*
25.0	*7794*	*19511*	**348**	28698	29046	*0.988*
30.0	*7794*	*19511*	**348**	28698	29046	*0.988*

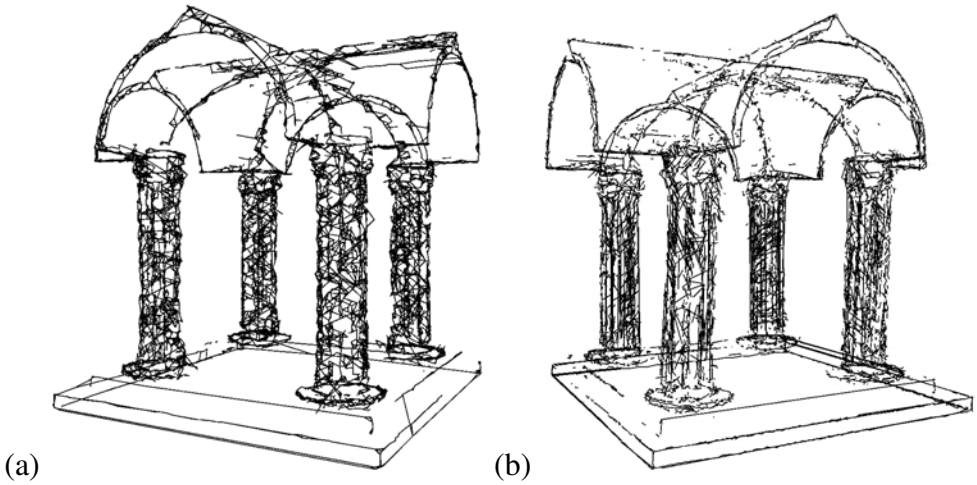
chairs						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	**4846**	**14563**	*181*	*12864*	*13045*	**0.986**
10.0	4476	10613	292	14743	15035	0.981
15.0	4320	9815	351	14789	15140	0.977
20.0	4303	9718	**372**	15040	15412	*0.976*
25.0	*4298*	9644	336	15117	15453	0.978
30.0	*4298*	*9633*	335	**15206**	**15541**	0.978

solids						
Max. Std. Dev.	vl	NL	fp	tp	EP	PR
5.0	**1278**	**5417**	*9*	*7092*	*7101*	**0.999**
10.0	1135	3980	20	9186	9206	0.998
15.0	1110	3802	**25**	9479	9504	*0.997*
20.0	*1107*	*3794*	**25**	**9503**	**9528**	*0.997*
25.0	*1107*	*3794*	**25**	**9503**	**9528**	*0.997*
30.0	*1107*	*3794*	**25**	**9503**	**9528**	*0.997*

Figure 6.15: *Variant B: Maximum standard deviation tests (data charts).*

6.4.5 Comparison

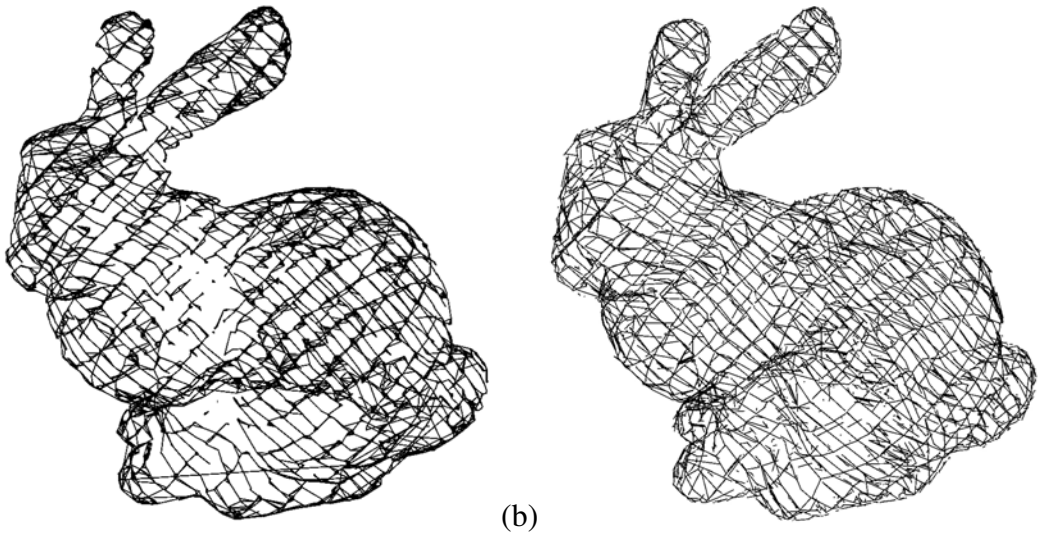
This section contains a side-by-side comparison of line reconstructions of each scene using both variants, including both synthetic and real scenes. The results are shown from Figure 6.16 through Figure 6.22.



(a) (b)

Figure 6.16: *Line reconstruction: Arcade scene*.
Results using:

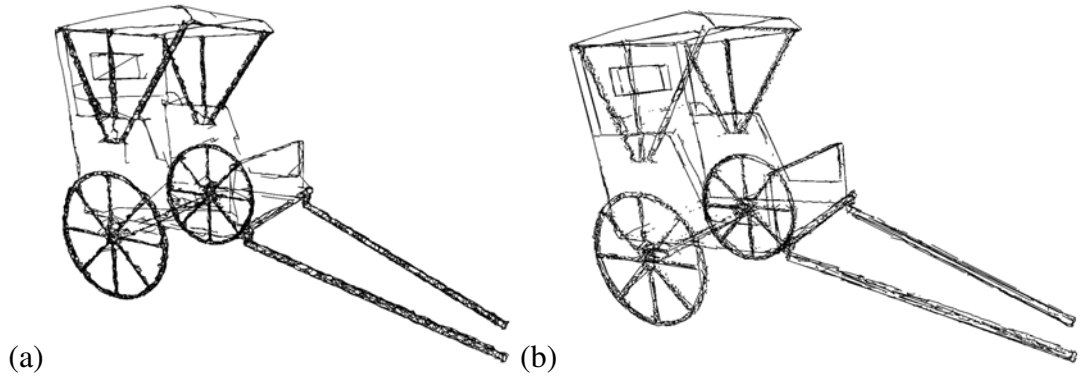
(a) merged points (variant A)
(b) incremental line reconstruction (variant B)



(a) (b)

Figure 6.17: *Line reconstruction: Bunny scene*.
Results using:

(a) merged points (variant A)
(b) incremental line reconstruction (variant B)

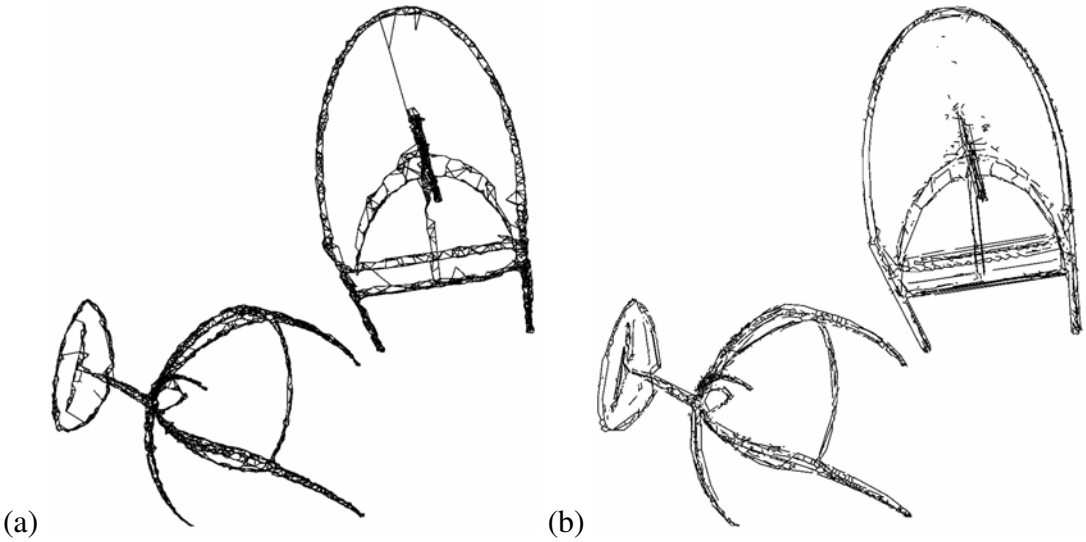


(a) (b)

Figure 6.18: *Line reconstruction: Carriage scene*.
Results using:

(a) merged points (variant A)

(b) incremental line reconstruction (variant B)

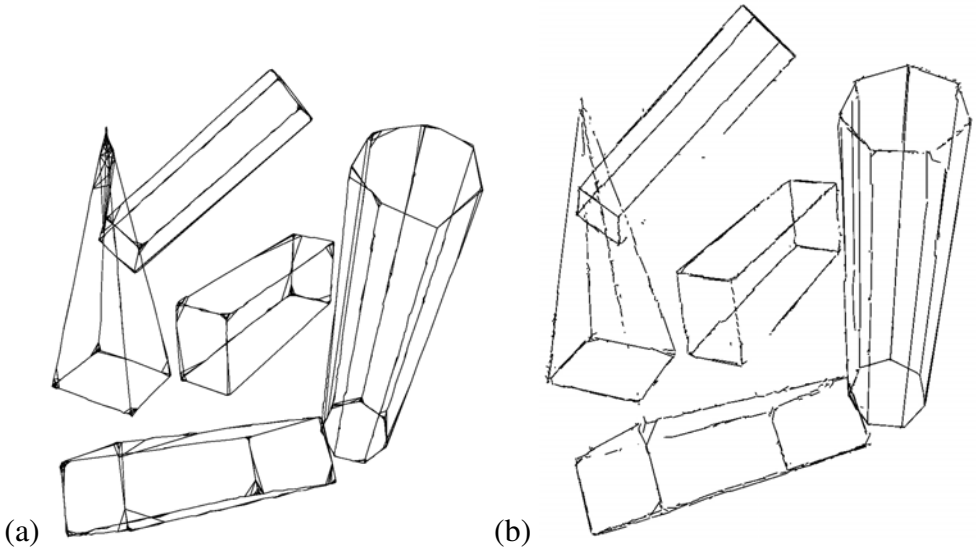


(a) (b)

Figure 6.19: *Line reconstruction: Chairs scene*.
Results using:

(a) merged points (variant A)

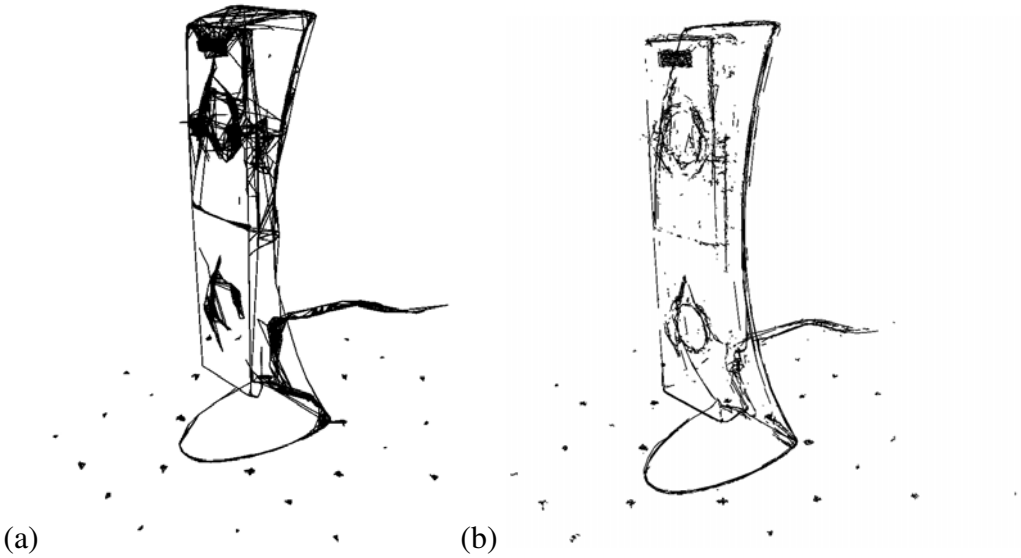
(b) incremental line reconstruction (variant B)



(a) (b)

Figure 6.20: *Line reconstruction: Solids scene*.
Results using:

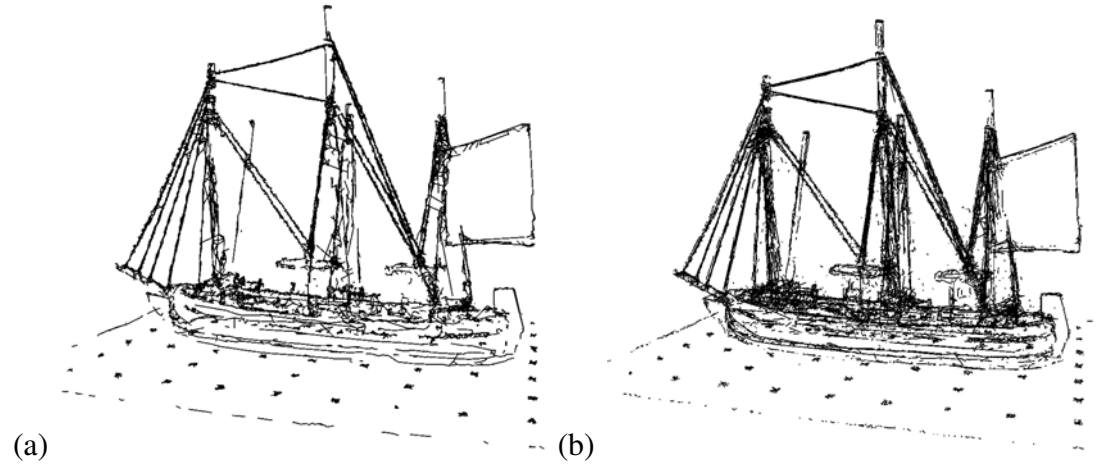
(a) merged points (variant A)
(b) incremental line reconstruction (variant B)



(a) (b)

Figure 6.21: *Line reconstruction: Speaker scene*.
Results using:

(a) merged points (variant A)
(b) incremental line reconstruction (variant B)



(a) (b)

Figure 6.22: *Line reconstruction: Boat scene*.
Results using:

(a) merged points (variant A)

(b) incremental line reconstruction (variant B)

By observing the results, it can be seen that the response of each variant in zones where the contours are not very dense is similar. Typical examples can be found on the steps of the arcade (Figure 6.16), the rooftop of the carriage (Figure 6.18), most of the solids scene (Figure 6.20), or the sail and hull of the boat (Figure 6.22).

In general, using variant B results in small breaks in the continuity of lines expected to be long, more than when using variant A. This is visible, for instance, in the solids scene (Figure 6.20), on the sides of the speaker (Figure 6.21), and on the border of the boat's sail (Figure 6.22). This fragmentation effect was described in the previous section, and is essentially due to the line removal process.

The major visual differences between the variants occur in regions where dense contours exist. One typical case are narrow surfaces such as the thin frontal faces of the arcs in the arcade (Figure 6.16), the bars on the front of the carriage (Figure 6.18), the legs and rims of the chairs (Figure 6.19), or the top and side of the speakers (Figure 6.21). In these cases, variant A produces a series of lines that seem to follow the real surface. Such connections are established when points are reconstructed in opposing borders of a thin surface, and points' view rays are close to tangent to the narrow surface.

In the contour map of Figure 6.23, the two contour points that are connected correspond to one edge of the surface (the one closest to the camera), but the merged points that are associated to it are on opposing edges. As these two opposing edges are close to each other, due to the narrow width of the surface, all the line is close to contours in most images. This leads to high average voting and low standard deviation. The tolerance to point assignment may also lead to connectivity in a sufficient number of images, and consequently the line is accepted as valid. The concentration of lines in the corners of the bunny's squares (Figure 6.17) and of the solids (Figure 6.20) is a result of this effect.

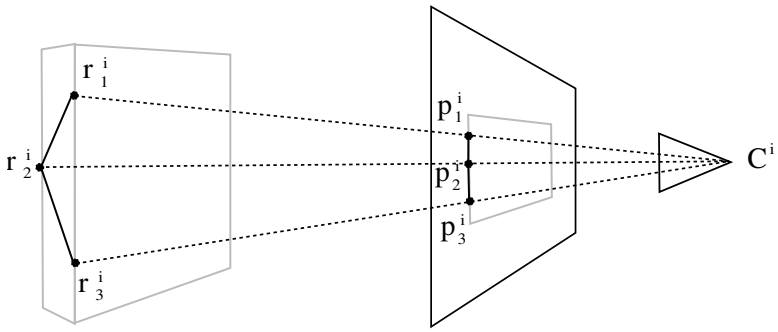


Figure 6.23: *Lines crossing narrow surfaces.*
When the view rays or the camera's look vectors are close to tangent to a surface, connectivity can be incorrectly established between points lying on opposing borders of the surface. However, if the surface is narrow, the lines will be small and close to many contours in different images, and therefore collect a high number of votes.

In variant B, connectivity is extracted from the same contour maps from which the points are reconstructed. Given the robustness of the point reconstruction, in most cases the two contour points are correctly reconstructed on the closest edge of the surface, and therefore, the candidate line is likely to be correct. The voting will be high, so the line is accepted as valid. This justifies the better definition of reconstructed lines on the referred narrow surfaces, when compared to variant A (although some cases similar to A may be found). It is also related to the better response to apparent contours in e.g. the bars in front of the carriage (Figure 6.18) and the masts on the boat (Figure 6.22).

An extreme case of dense contours occurs in the columns of the arcade (Figure 6.16). In this case, the amount of cross voting is very high, so any candidate lines that cross a column from one side to the other is likely to have high voting. These lines occur in both cases. On the results of variant A this effect is not so distinguishable among the cluttering caused by the surface-aligned lines. But in variant B, as that cluttering is smaller, both the volume-crossing lines and the real vertical line pattern of the column are more noticeable.

Still regarding outliers, there are some cases where it is noticeable that the occurrence of outliers in variant B is higher than in variant A. These can be noticed in places such as the arcade's roof (Figure 6.16), the back of the round chair (Figure 6.19) or the center of the boat (Figure 6.22). This was to be expected, given the additional layer of robustness provided by the point merging in variant A.

Regarding performance, Variant B has a clearly lower computational load when compared to variant A. It not only requires a smaller number of line reconstructions, as it also does not require point merging. As an example, Table 6.2 shows a comparison between the number of lines evaluated and the execution time for both variant in the boat scene.

Variant	Nr. of Lines Evaluated	Time (ms)
A	109 373	353 837
B	18 378	95 023

Table 6.2: *Performance comparison between the two line reconstruction variants*. This comparison relates to the boat scene

The slower performance of Variant A compared with Variant B is caused not only by the higher number of lines evaluated, but especially by the point merging procedure, that takes about 50% of the processing time.

6.5 Comparison with other line reconstruction methods

Section 2.2 discussed existing work on reconstruction, including methods that performed line reconstruction. The methods that do line reconstruction and that have a closer relation to the work presented here will now be reviewed in more detail.

Seitz and Dyer's method [Seit 95] establishes correspondences between line segments in different images based on intersection of depth ranges (as described in Section 2.2). Its use of a single point to represent a line from the reference image, together with the requirement that only depth ranges that get intersected by line projections from *all* images are considered, may lead to the loss of many correspondences, if the part of the real 3D line corresponding to the representative point is missed in some images. Furthermore, they rely on rectification to perform epipolar search. In the method presented here, by using whole lines, collecting them from multiple reference images, and using the voting-based approach to evaluate them, the freedom in camera placement and the robustness to missing points are bound to be higher.

Schmid and Zisserman [Schm 97] only process pairs or triplets of images for line reconstruction. They propose a "short-range" version of the algorithm, in which for each line segment in one image, a matching score is computed for each of the segments in another image. This matching score is based on individual correlations for each pixel of the original line and the corresponding points in the candidate lines in another image (only points common to both segments are used for this). Epipolar geometry is used to assist in correspondence.

The "long range" version corrects the area surrounding a line corresponding to the correlation using a projective warping, before computing the matching scores. As the large baseline is prone to cause significant occlusion, correlation is computed separately for each side of the line, and the best of the two correlation scores is used to represent that correspondence separately, choosing the best correlation.

One drawback of this method, compared to the method presented here, is the use of correlation, that may limit its applicability in the case of wide baseline settings and low textures, as the projective warping as proposed may not be enough to cope with strong perspective effects. The use of triplets may also not be sufficient to cope with

occlusion or problems in edge detection.

Taillandier and Deriche [Tail 02] propose an approach using voxels instead of depth ranges. Using a plane-sweeping technique, sets of possible correspondences are collected if two or more labeled line segments from different images intersect in at least one voxel. The corresponded 2D line segments are then used to reconstruct a 3D infinite line by determining the intersection of the planes defined by each camera's center and the 2D line segment of that camera. The intersections are determined using an iterative method that seeks convergence to an infinite 3D line by minimizing a residual error. To prune the final line set, only correspondences matching segments from three or more images are considered; lines with more correspondences are selected first, and a unicity constraint is iteratively enforced, such that a 2D segment can only be associated with one 3D line. To obtain a final 3D line segment, the corresponded 2D line segments are projected on the infinite 3D line, and the union of all projected 2D line segments (i.e. the longest 3D segment possible) is taken. This technique was presented with a focus on aerial and small baseline settings, and it is not clear how good it would perform in wider baseline settings. It also does not consider errors in camera calibration, which may cause problems e.g. when the the planes corresponding to two corresponding 2D line segments do not intersect within the swept volume, leading to missed correspondences. The use of the distance-weighted metric proposed in this thesis would help in such cases.

The method used by Kawamoto and Imiya [Kawa 01] is based on random sampling of pairs of points and Hough-like accumulation. The accumulator space describes a set of possible infinite 3D lines. Pairs of points from different images are randomly chosen, and if they obey the epipolar constraint, the corresponding reconstructed point is used to vote on the line accumulator space. This method is very inefficient, essentially due to the random choice of the point pairs.

Kahl and August [Kahl 03] energy minimization technique over parameterized curves has high computational complexity, both in the selection of the initial line set and the energy minimization, and there are problems dealing with occlusions.

A few techniques exist oriented to the structure and motion problem, such as [Tayl 95, Bart 03, Roth 03, Mart 03, McLa 00], where the focus is usually set more on the calibration than on the reconstruction. In some cases, correspondences are given or manually computed (and mostly correct) [Tayl 95, Bart 03, Roth 03]. In other cases (e.g. [Mart 03, McLa 00]), the methods employed to estimate correspondences rely on line tracking methods that are constrained to specific scenarios (e.g. small baseline), or are applicable only to a relatively low number of features (those that are trackable), making them sensitive to lost or corrupted features between images.

6.6 Final remarks

The point reconstruction algorithm presented in Chapter 3 has been extended in this chapter to line reconstruction. By combining and re-using concepts and data from point reconstruction, a series of benefits were obtained:

- The number of points that need to be reconstructed to represent a scene can be drastically reduced;
- Line reconstruction inherits some advantages of point reconstruction: less sensitivity to wide baseline, robustness in estimation, etc.;
- Lines provide a better representation for certain types of scenes;

Two variants were proposed, one based on connectivity between merged points – variant A – and another based on incremental point and line reconstruction – variant B.

These variants were implemented using the same hardware-based techniques used for point reconstruction and applied to several scenes. The line reconstructions obtained with both variants are encouraging, showing in general good coverage of the scenes' contours, as well as good localization.

Results have shown that variant A, by being based on merged points, presents less outliers, and better connectivity on long lines than variant B. On the other hand, it shows some problems in cases of narrow surfaces or close contours, generating 3D lines on those surfaces that do not correspond to any real contour. Variant B behaves better in these cases, providing line reconstructions with better definition in those regions. Furthermore, due to its incremental process, variant B requires much less computation than variant A.

As a conclusion, variant A would be more adequate for scenes where there are long lines, low density of contours or few narrow surfaces, and when the existence of outliers (even if few) is a major issue, and computation time is less important. For more general scenes, with more dense or apparent contours, or when performance is more important, variant B should be more suitable.

Chapter 7

Conclusions and future work

7.1 Conclusions

Three-dimensional reconstruction can be the source of multiple and interesting applications. The more robust, fast and widely available it becomes, the higher the number of possibilities and the more attractive and useful they become.

Although different methods exist for 3D reconstruction, each represents a compromise between a series of factors, such as constraints of the type of scenes and objects, limitations in the placement of cameras, robustness to camera calibration errors or noisy images, and performance. Some of the more relevant existing methods were discussed in Section 2.2.

The motivation for this thesis was to try to develop a method that would simultaneously:

relax constraints that are common to many reconstruction methods Examples of such constraints include small baseline limitations of correlation-based methods, color-consistency requirements of voxel-based approaches, or background separation of visual hull techniques.

be robust to error sources commonly found in real scenes Image noise, problems in contour extraction and errors in camera calibration are bound to occur in the reconstruction process. Reconstruction methods that are correlation-based can be sensitive to image noise, feature-based methods suffer in the absence of features or the existence of false features, and volumetric methods are sensitive to errors in camera calibration.

be able to deal with occlusion Occlusion occurs in many types of scenes, and causes problems especially to methods based in correspondence between a small number of images (e.g. pair-wise, triple-wise), as partially occluded parts have higher probability of being incorrectly corresponded.

provide good results without resorting to regularization Many methods provide reconstructions visually appealing, by applying strong regularization to the reconstructed entities. Such a practice leads e.g. to the removal of strong surface

discontinuities, thus smoothing out outliers, but also affects real discontinuities, such as sharp edges or occlusion borders.

provide performance in current off-the-shelf systems The common trade-off between quality and speed is in many cases high, especially in the case of view-independent reconstruction methods, and the complexity of metrics often leads to long reconstruction times, resulting in very low interactivity, or require specialized hardware or multiple computers.

To accomplish those goals, an innovative combination of concepts was explored, that included:

- ideas from different reconstruction methods, such as
  - the use of contour points, that reduces baseline restrictions;
  - the use of multiple cameras, that leads to an increase in confidence of estimates;
  - the use of an absolute referential for estimation, facilitating integration of information;
- techniques from other fields of image processing, namely the use of distance transforms to provide an implicit correspondence metric for contour points;
- the use of a voting mechanism that yields robustness by accumulation of evidence;
- tools from the computer graphics and graphics hardware fields, such as projective texturing and graphics cards standard API's, that increase performance to the method and make it usable in current off-the-shelf hardware.

Some of these choices led to new problems and issues that had to be addressed. For instance, the low disambiguation power of contours causes cross-voting situations, that had to be solved by performing cross-checking. In Chapter 3 the reconstruction method was described, together with the issues involved and proposed solutions.

Given the computational load required for reconstruction from multiple images, and the desire to provide a reconstruction method that would be able to perform reconstruction at interactive rates, an hardware-accelerated implementation of the reconstruction method was developed, and it was presented in Chapter 4.

With this implementation, a series of experiments and tests were performed and reported in Chapter 5. These tests were done with different goals in mind, namely:

- to measure the quality of the reconstructions;
- to analyze the influence of some parameters of the algorithm in the resulting reconstructions;
- to assess the robustness of the algorithm to adverse conditions;

- to verify its applicability to real scenes;
- to evaluate the performance of the hardware-based implementation.

Tests were first performed with synthetic scenes for which ground truth data was available. With this information, it was possible to have a measure of the quality of the reconstruction, and to analyze the sensitivity of the algorithm and its parameters.

The tests with synthetic scenes were performed with both exact and perturbed camera calibration data. The results were promising in both cases. With exact camera calibration, very high values of precision and recall were achieved, and it was possible to isolate in most cases the influence of each parameter in the results. With errors in camera calibration, the values of precision and recall lowered as expected, but the reconstructed point sets have shown a graceful degradation, as desired.

The tests with real scenes followed, to show the method's applicability. The results were quite satisfactory: occlusion, highlights, lighting variations, apparent contours and other issues were robustly dealt with in most cases, and the reconstructed point sets have shown very good localization. It is important to notice that the good results obtained did not benefit from any regularization whatsoever.

The performance of the GPU-based implementation was also tested, to assess the influence of some implementation variants and to compare it to a CPU-based implementation. The results obtained show significant gains in performance when rendering multiple lines in a single buffer (up to 43%), as well as substantial speed differences when comparing the GPU-based implementation with the CPU-based (up to 54 times). This level of acceleration make it possible e.g. to obtain a feature point reconstruction of the boat scene using all images as reference in less than 30 seconds (not including startup overhead). It was also concluded that the hardware-accelerated implementation could run unmodified in not-so-recent graphic cards, and still provide acceleration, when compared to the CPU version.

Based on the results with points, an extension to reconstruct lines was proposed in Chapter 6, although still in an early stage. This extension has two important properties, namely it benefits from being bootstrapped with a robust point set, and it is based on the same concepts and rationale behind point reconstruction – voting based on distance transforms and multiple images – and it can re-use information that had been pre-computed for point reconstruction, such as the vote maps. Two variants of line reconstruction were proposed and implemented using the same GPU-based techniques used for point reconstruction. The line reconstructions obtained with both variants are very encouraging, showing in general good coverage of the scenes' contours, as well as good localization. Apart from enriching the model resulting from point reconstruction, the proposed extension to line reconstruction also serves as an example of how point reconstruction could be integrated in a pipeline with other methods to provide more complete or specialized reconstructions.

As a conclusion, the results show that the main objectives of the thesis were achieved. The proposed point reconstruction method can:

Be applied to wide-baseline, weakly-constrained camera settings The use of contour points instead of color-consistency or correlation-based metrics provides

freedom in camera placement.

Be robust to image noise, problems in contours and errors in camera calibration The direct incorporation of closeness in the correspondence metric seamlessly integrates the correspondence and the tolerance to contour offsets and flaws, caused by various sources.

Resist to lighting variations By relying on contours, the sensitivity to lighting variations can be as low as that of the contour detection algorithm used.

Cope with occlusion The accumulation of votes over multiple images, together with the world-space formulation of the reconstruction problem, allows to gather evidence from objects which are visible only in some images, even if such images are not related by any kind of spatial order.

Produce good results without regularization The results obtained do not rely on regularization as a way of smoothing or removing outliers, thus preserving sharp depth discontinuities of the scene.

Perform at interactive rates The proposed GPU-based implementation allows for fast reconstruction, which can be controlled in terms of level of detail, enabling both fast previews and slower, more complete reconstructions.

Overall the point reconstruction method presented in this thesis was able to deal with many important issues that hamper previous works. It was shown that the chosen metric, albeit apparently simple, has a potential to be leveraged by the accumulation of information from multiple images, and allows to reconstruct, and implicitly establish correspondences between contour points without requiring additional gradient or color information. Performance-wise, the method was able to fulfill the desire for interaction. Finally, the same methodology proved to be able to provide interesting results for line reconstruction. Although not yet as polished as the point reconstruction counter part, it shows how point reconstruction can be used as a bootstrap for other methods.

7.2 Future work

The insight provided by the work described in this thesis leaves room for exploration in various directions. Three major lines of future improvement are now identified: robustness, applicability and performance.

Robustness

Although the robustness levels achieved with the presented method are high, there is still room for improvement. In the point reconstruction algorithm, it would be interesting to try to map other types of metrics to pre-computed vote maps, e.g. an hash of color distributions.

Some intricacies surround this idea. The metrics used should be independent of the point to reconstruct, otherwise they cannot be pre-computed. They should also be resistant to wide baseline, otherwise the freedom in camera placement would be lost. Another issue is the integration with the distance-based metric, such that the robustness to offsets is not lost or severely reduced.

Working on the inputs may also allow an increase in robustness. For instance, some feedback could be introduced between reconstruction and camera calibration, so that a camera with bad calibration could be detected from e.g. the distribution of votes it casts on view rays from other images, or other statistical measures. Such cameras could then have their parameters adjusted in order to correct the calibration and improve the reconstruction. Also the use, in the same reconstruction, of multiple contour maps for the same image, using either the same edge detector with different parameters, or multiple edge detectors, could also provide better source data for the reconstruction.

Applicability

The method presented can be used in different applications or scenarios, for which additional processes would be needed. One possible scenario is one of an autonomous active system, that can control camera positioning, such that it can iteratively capture images from a set of viewpoints, perform reconstruction, and automatically determined additional viewpoints required to extract missing details. The system would then be able to position the camera in the new positions, and improve the reconstruction. This type of systems exist, e.g. in laser-based systems [Sequ 96], and it would be interesting to see how the method presented here would be able to fit in such a model. In this context, the exploration of incremental reconstruction would also play an important part.

Other scenario could include extensions for providing contours in scenes with little contour information. One could e.g. use simple forms of structured lighting to produce contours in such scenes, with the advantage that the system did not need to be optimized or bound to any specific kind of pattern.

Extending the type of output of reconstruction also leads to a wider range of applications. In some cases, techniques for reconstructing surfaces from point sets such as the Power Crust [Amen 01] can deliver a mesh-like representation of a scene. Following the reasoning behind the extension of point to line reconstruction, it would also be possible to extend line reconstruction to triangle (or polygon) reconstruction, by deriving 3D triangles from 2D line sets, and evaluating and classifying those. Some experiments were performed in these types of representation, showing that there are promising possibilities in those alternatives and in combinations of them.

Performance

In terms of the hardware-based implementation, some procedures could be optimized. More recent graphic cards allow the use of more capable buffers, with higher number of bits per pixel, and also the ability to program the card to perform specific

tasks on geometry or pixels – GPU programming. The possibility of programming the GPU would allow to implement other parts of the reconstruction algorithm to be performed in the GPU. The generation of vote maps, the detection of maxima in the candidate set or the point merging are some examples of tasks that could benefit from this.

Appendix A

Instructions for the accompanying CD

This thesis includes a CD containing additional data and information about this work. The following sections provide some instructions on how to use the CD, namely how to access it, and how to use the reconstruction viewer included. In terms of content, the following can be found in the CD:

PDF version of the thesis An Adobe PDF file containing this thesis, with hyperlinks. A PDF viewer is required to view this file. If one is not available, free viewers for multiple plataforms can be downloaded from Adobe – *http://www.adobe.com*. A good lightweight alternative for Microsoft Windows systems can be found at *http://www.foxitsoftware.com* [1]

Interactive reconstruction stereo viewer This is an interactive 3D viewer of point and line files generated by the 3D reconstruction software. It can be used to chose a scene, and browse through its point and line reconstructions. The user can navigate in the scene and control a set of visualization parameters – point and line size, view or hide the original model (in the case of synthetic scenes), activate point splatting (when available), and activate anaglyph stereo viewing and control its separation. This viewer is currently only available for Microsoft Windows systems only. For those that do not have access to such systems, VRML files are also provided (see below). Instructions for the use of the reconstruction viewer are provided in Section A.2.

Reconstruction VRML files These files have been included for those that for some reason cannot use the reconstruction viewer. A VRML viewer (or browser, as it is also referred to) is required to view and manipulate the VRML files. VRML viewers exist for a variety of platforms, and an up-to-date list can be found at the Web 3D consortium: *http://www.web3d.org/x3d/vrml/tools/viewers_and_browsers/*.

Reconstruction images Images of the reconstruction are also available, if for some reason none of the options above can be used, or the reader wishes to browse

[1] Due to redistribution licensing requirements and the variety of platforms, neither of these options were included in the CD.

through the scenes or print some particular scene. Normal and stereo versions are provided.

A.1 Accessing the contents of the CD

A.1.1 Using the menu

In MS Windows Systems, a menu (Figure A.1) should appear when the CD is inserted into the CD drive. If that does not happen (e.g. due to the "autorun" feature being disabled), the user can use a file explorer (or file browser) to navigate to the CD and execute the file *menu.exe* in the root directory of the CD.

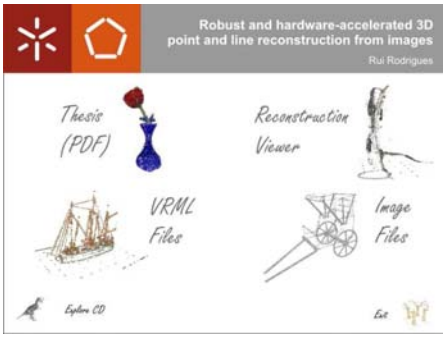


Figure A.1: *CD menu*.

The menu provides one option for each of the four types of contents mentioned before, plus an extra option to explore the CD and another option to exit. The "Thesis PDF" launches the default PDF viewer installed in the system, and the "Reconstruction Viewer" option launches the viewer. The other two options, "Images" and "VRML files", open a file explorer in the directories with the corresponding files, so that the user may choose which files to view. The extra "Explore CD" option launches a file explorer in the root directory of the CD.

A.1.2 Navigating through the directories and the files

Alternatively, one may navigate through the directories with a file explorer and open the files of interest. The thesis file is located in the root directory, and each of the other contents can be found in directories with the corresponding names: "recon-viewer", "vrml" and "images". The "vrml" and "image" directories contain one sub-directory per scene. Inside each scene, file names are based on the parameters used to generate the corresponding reconstructions. The first two parts identify the scene and the type of test, and the remaining the parameter values used. For instance, *arcade.bcct.cct0.dt3.vpt0.9.cnds800.sca30.csv0.png* is the image file corresponding to the cross-check threshold test (*bcct*) of the arcade scene, with parameters cross check threshold = 0 (*cct0*), vote range = 3 (*dt3*), vote percent threshold = 0.9 (*vpt0.9*), 800

Label	Test set
bbasis	Default values
bfull	Full reconstruction
bdt	Vote map range
bcct	Cross-check threshold
bsca	Similar camera angle
bvpt	Vote percent threshlod
bcnds	Nr. of candidates
bm2D	Merge 2D threshold
bm3D	Merge 3D threshold
ecamerr	Defaults, with camera errors
edt	As bdt, with errors
ecct	As bcct, with errors
esca	As bsca, with errors
em2D	As bm2D, with errors
em3D	As bm3D, with errors
mergeconn	Lines, variant A
incrline	Lines, variant B

Label	Parameter
dt	Vote map range
cct	Cross-check threshold
sca	Similar camera angle
vpt	Vote percent threshold
cnds	Nr. of candidates
m2D	Merge 2D threshold
m3D	Merge 3D threshold
cnt	Lines image count
avg	Lines average
stdev	Lines std. deviation

Table A.1: *List of file name labels*. The table on the left indicates the labels used to identify the test sets, and the table on the rught the labels that identify the parameter values.

candidates (*cnds800*) and similar camera angle = 30 (*sca30*). The last part is an internal identifier. For merging tests, the values of the two merging parameters are added e.g. *arcade.bm3D.cct1.dt3.vpt0.9.cnds800.sca30.m3D0.01.m2D3.csv0.png*. Table A.1 contains the list of possible labels and the corresponding tests and parameters that can be found for most scenes, including all the reconstructions associated with the test results presented in the tables of Chapter 5 and Chapter 6.

A.2 Interactive reconstruction viewer

The interface of the interactive reconstruction stereo viewer, from this point on referred as "viewer", consists of two windows (Figure A.2): one containing a series of control parameters – the control window – and another containing the visualization area, where the reconstruction and models are displayed, and the view point can be manipulated – the display window. Closing any of the windows will close the program.

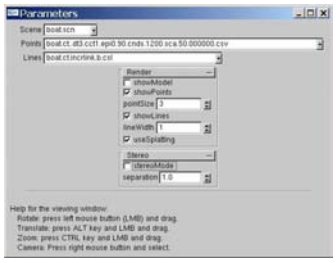
(a)  (b) 

Figure A.2: *Interactive reconstruction viewer.*
(a) Control window
(b) Display window

A.2.1 The control window

The control window (Figure A.2(a)) contains a top area consisting of three list boxes that allow the user to choose which scene and reconstruction is to be shown in the display window and a second area containing parameters that define how the chosen scene's model and reconstructions are rendered.

Each time a new scene is selected in the scene list box, the lists of point and line reconstructions available for that scene are loaded into the corresponding list boxes, and the first point reconstruction available for that scene is loaded into the display window. The user may then choose different point and line reconstructions from the corresponding list boxes[2]. The names of the reconstructions follow the same conventions used for VRML and image file names described in Section A.1.2.

In the "Render" panel, the user may choose which elements should be rendered in the display window: the original model (if available), the point reconstruction or the line reconstruction. By checking the corresponding boxes, the user defines if a given type of data (model, points, lines) is visible or not. Any combination of the three types of data is possible. If some of the types marked as visible is not available, e.g. there is no original model, or no line reconstruction has been selected, it simply is not rendered.

The user may also define what is the point or line size to be used for rendering point and line primitives, by typing in the values (in pixels) in the corresponding text boxes, or using the up and down arrows next to the boxes.

Some point reconstructions have color information extracted from the original images and associated to the points. If the option "useSplatting" is selected, that color will be used in the point rendering.

It is possible to view the 3D data in red-cyan stereo anaglyphs using the "Stereo" panel. Stereo viewing is activated by checking the "stereoMode" box. Using a pair of

[2] A practical way to browse through the multiple point reconstructions of a given scene is to use the 'Tab' key to activate the corresponding list box, and then use the 'up' or 'down' arrow keys to change between reconstructions. The same applies to line reconstructions, and even for the scenes, although in this last case it may be less useful.

red-cyan glasses (red lens on the left eye, cyan on the right eye), the user should be able to perceive the depth of the 3D data displayed[3]. The depth effect can be increased or decreased using the "separation" parameter. The eye accommodation varies from person to person, so the user should experiment with values until finding one that he or she feels comfortable with.

In the bottom of the control window there is a small help regarding the navigation in the display window, explained next.

A.2.2 The display window

In the display window (Figure A.2(b)), the user can change the view of the scene using the left mouse button (LMB) and some keys. To rotate the scene, the user should simply click on the window and drag the mouse in the desired direction. Translation is performed by pressing and holding the "Alt" key before and while clicking and dragging. To zoom, the "Ctrl" key has to be pressed and held before clicking and dragging.

The right mouse button gives access to a list of cameras available for the scene. By selecting one of the cameras, the view point is automatically placed in the corresponding position and orientation. The angle bracket keys '<' and '>' also shift between cameras.

The display window may be freely resized, being the scene scale adjusted to the new window dimensions.

[3]It is not advisable to use the glasses continuously for long periods of time. Small breaks should be taken from time to time, to avoid eye stress and headaches.

Appendix B

Additional reconstruction images

This appendix contains point and line reconstructions of the scenes presented throughout the thesis, plus some additional scenes. The reconstructions are printed as red-cyan stereo anaglyphs, such that the viewer can have a perception of depth when viewing them with red-cyan glasses. This and other material can be found in the accompanying CD, as described in Appendix A [1].

Figure B.1 shows reconstructions of the speaker and boat scenes. Figure B.2 shows reconstructions of the synthetic scenes: arcade, bunny, carriage, chairs and solids.

The additional scenes are shown in Figure B.3 and Figure B.4. Figure B.3 contains two real scenes – clayjar (29 images) and cube (32 images). For each of the scenes, one of the original images and the reconstruction using all images are shown. Figure B.4 contains one real scene – dino (30 images) – and two synthetic scenes – rose (69 images) and knot (69 images) – also with one original image per reconstruction.

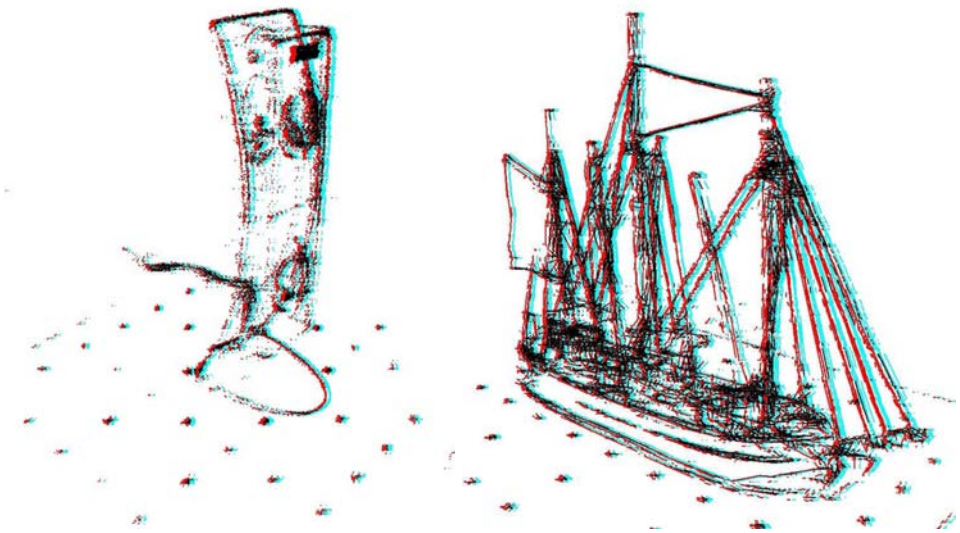


Figure B.1: *Reconstructions of the speaker and boat scene*.

[1] Due to color fluctuations in printed material, some ghosting may be noticeable. Better results can be obtained when viewing on a monitor the anaglyphs provided in the CD.

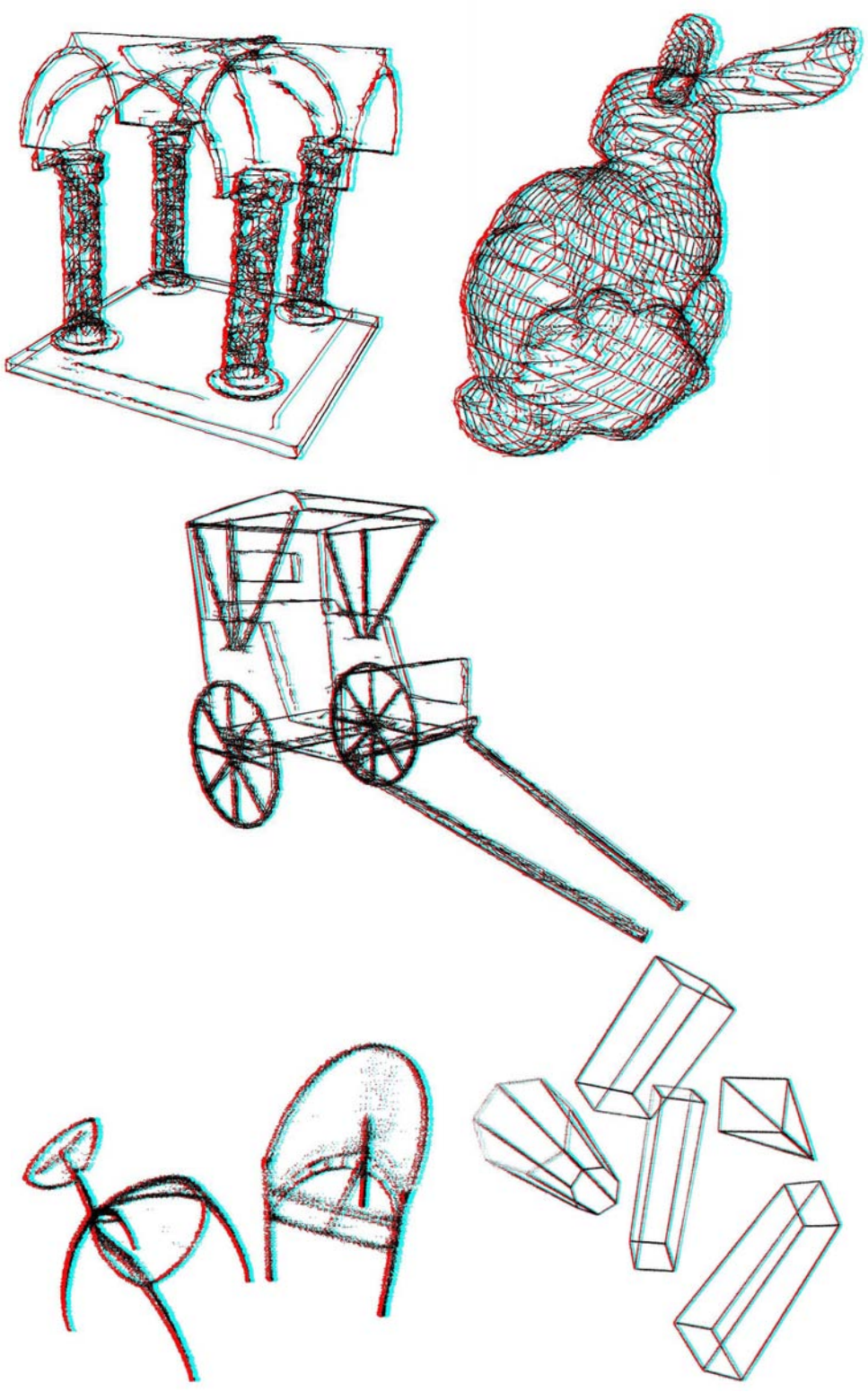


Figure B.2: *Reconstructions of the synthetic scenes.*

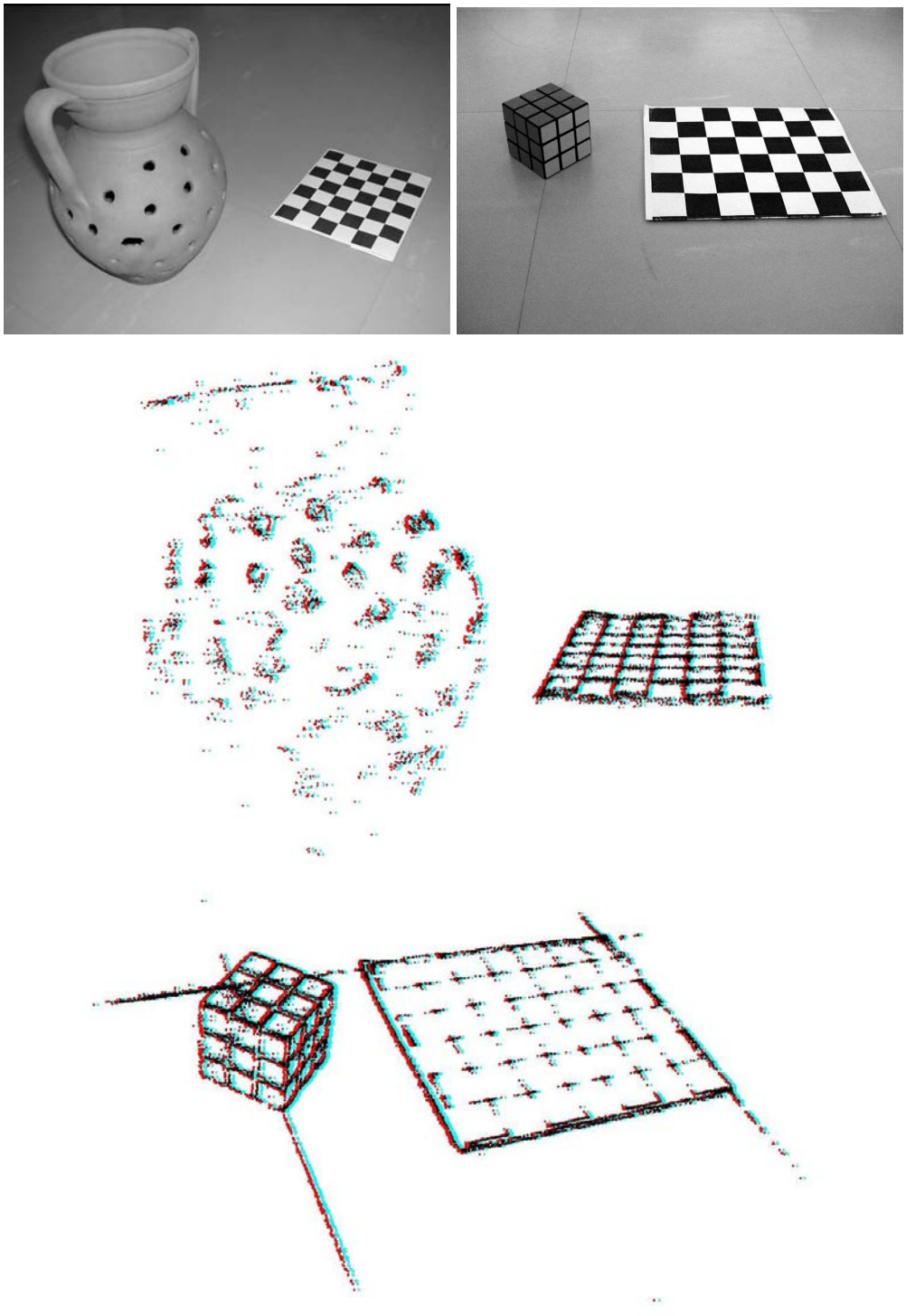


Figure B.3: *Additional scenes: clayjar and cube.*
Original images of the clayjar and cube scenes (top row) and the corresponding reconstructions (bottom rows)

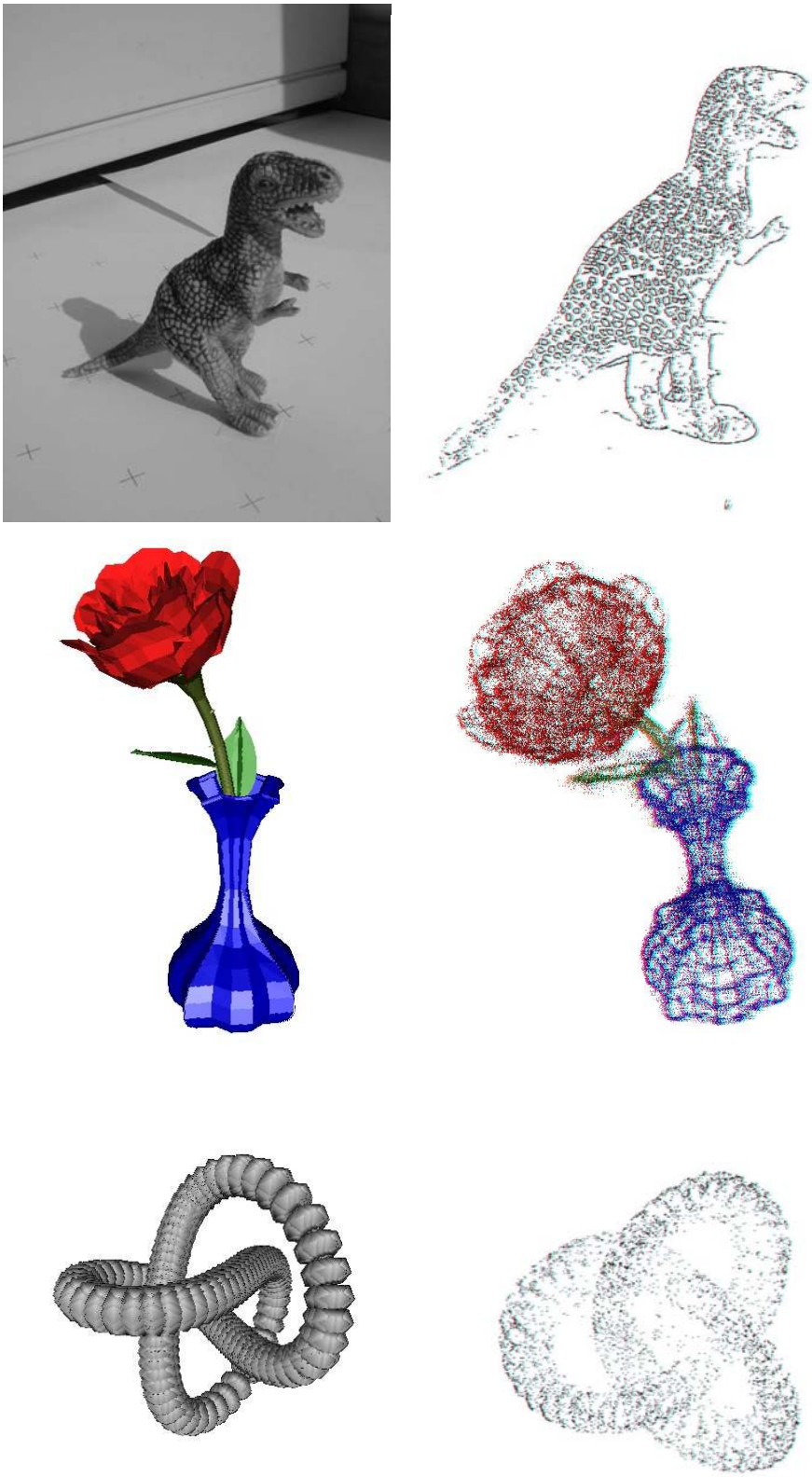


Figure B.4: *Additional scenes: dino, rose and knot.*
Original images of the dino, rose and knot scenes (left column) and the corresponding reconstructions (right)

Bibliography

[Adel 85] E. H. Adelson and J. R. Bergen. "Spatiotemporal energy models for the perception of motion.". *Journal of the Optical Society of America A*, Vol. 2, No. 2, pp. 284–299, 1985.

[Agui 01] P. M. Q. Aguiar and J. M. F. Moura. "Three-dimensional modeling from two-dimensional video". *IEEE Transactions on Image Processing*, Vol. 10, No. 10, pp. 1541–1551, October 2001.

[Aken 02] T. Akenine-Moller, T. Moller, and E. Haines. *Real-time rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[Amen 01] N. Amenta, S. Choi, and R. Kolluri. "The power crust". In: *Sixth ACM Symposium on Solid Modeling and Applications*, pp. 249–260, 2001.

[Asad 98] N. Asada, H. Fujiwara, and T. Matsuyama. "Edge and depth from focus". *International Journal of Computer Vision*, Vol. 26, No. 2, pp. 153–163, 1998.

[Bart 03] A. Bartoli and P. Sturm. "Multiple-view structure and motion from line correspondences". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'03*, pp. 207–212, IEEE CSP, October 2003.

[Baue 04] J. Bauer, A. Klaus, M. Sormann, and K. Karner. "Sparse 3D reconstruction by edgel sweeping". In: *Proceedings of Computer Vision Winter Workshop - CVWW'04*, pp. 11–20, Piran, Slowenien, February 2004.

[Beau 95] S. S. Beauchemin and J. L. Barron. "The computation of optical flow". *ACM Computing Surveys*, Vol. 27, No. 3, pp. 433–466, 1995.

[Bhot 02] R. Bhotika, D. Fleet, and K. Kutulakos. "A probabilistic theory of occupancy and emptiness". In: *Proceedings of European Conference on Computer Vision - ECCV'02*, pp. 112–132, 2002.

[Blum 89] G. W. Bluman and S. Kumei. "Symmetries and differential equations". In: *Applied Mathematical Sciences*, Springer-Verlag, New York, 1989.

[Boll 87] R. C. Bolles, H. H. Baker, and D. H. Marimont. "Epipolar-plane image analysis: an approach to determining structure from motion". *International Journal of Computer Vision*, Vol. 1, No. 1, pp. 7–55, 1987.

[Bone 99] J. D. Bonet and P. Viola. "Roxels: responsibility weighted 3D volume reconstruction". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'99*, pp. 418–425, 1999.

[Borg 84] G. Borgefors. "Distance transformations in arbitrary dimensions". *Computer Vision, Graphics and Image Processing*, Vol. 27, No. 3, pp. 321–345, 1984.

[Bren 73] R. P. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Broa 00] A. Broadhurst and R. Cipolla. "A statistical consistency check for the space carving algorithm". In: *Proceedings of the British Machine Vision Conference - BMVC'00*, pp. 282–291, 2000.

[Cann 86] J. Canny. "A computational approach to edge detection". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679–698, 1986.

[Capl 95] A. Caplier and F. Luthon. "A new spatiotemporal approach for image analysis - application to motion detection". In: V. Hlavac and R. Sara, Eds., *Computer Analysis of Images and Patterns*, pp. 246–253, Springer, Berlin,, 1995.

[Chen 01] Y.-Q. Cheng, X. G. Wang, R. T. Collins, E. M. Riseman, and A. R. Hanson. "Three-dimensional reconstruction of points and lines with unknown correspondence across images". *International Journal of Computer Vision*, Vol. 45, No. 2, pp. 129–156, 2001.

[Cheu 03] G. Cheung, S. Baker, and T. Kanade. "Visual hull alignment and refinement across time:a 3D reconstruction algorithm combining shape-from-silhouette with stereo". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR03*, June 2003.

[Chie 86] C. H. Chien and J. K. Aggarwal. "Volume/Surface Octrees for the Representation of Three-Dimensional Objects". *Computer Vision, Graphics and Image Processing*, Vol. 36, pp. 100–113, 1986.

[Cipo 90] R. Cipolla and A. Blake. "The dynamic analysis of apparent contours". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'90*, pp. 616–623, Osaka, Japan, December 1990.

[Cipo 92] R. Cipolla and A. Blake. "Surface shape from the deformation of apparent contours". *International Journal of Computer Vision*, Vol. 9, No. 2, pp. 83–112, 1992.

[Cler 02] M. Clerc and S. Mallat. "The texture gradient equation for recovering shape from texture". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 24, No. 4, pp. 536–549, 2002.

[Coll 96] R. Collins. "A space-sweep approach to true multi-image matching". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR96*, pp. 358–363, 1996.

[Corn 01] K. Cornelis, M. Pollefeys, M. Vergauwen, and L. J. V. Gool. "Augmented reality using uncalibrated video sequences". *Lecture Notes in Computer Science*, Vol. 2018, pp. 144–160, 2001.

[Cox 94] I. Cox. "A maximum likelihood n-camera stereo algorithm". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR94*, pp. 733–739, Seattle, WA, USA, June 1994.

[Culb 00] W. B. Culbertson, T. Malzbender, and G. Slabaugh. "Generalized voxel coloring". In: *Vision Algorithms: Theory and Practice*, pp. 100–115, Springer-Verlag, 2000.

[Debe 96] P. E. Debevec, C. J. Taylor, and J. Malik. "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach". In: *Proceedings of SIGGRAPH'96*, pp. 11–20, 1996.

[Doug 73] D. Douglas and T. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature". *The Canadian Cartographer*, Vol. 10, No. 2, pp. 112–122, 1973.

[Duc 95] B. Duc, P. Schroeter, and J. Biguen. "Spatio-temporal robust motion estimation and segmentation". *Lecture Notes in Computer Science*, Vol. 970, pp. 238–245, 1995.

[Eise 99] P. Eisert, E. Steinbach, and B. Girod. "Multi-hypothesis, volumetric reconstruction of 3-D objects from multiple calibrated camera views". In: *International Conference on Acoustics Speech and Signal Processing*, pp. 3509–3512, Phoenix, March 1999.

[EOS] EOS Systems. "Photomodeler". `http:/www.photomodeler.com/`.

[Faug 93] O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[Faug 95] O. Faugeras and B. Mourrain. "About the correspondences of points between n images". In: *Proceedings of Workshop on Representation of Visual Scenes*, pp. 37–44, Cambridge,Massachusetts, USA, June 1995.

[Faug 98] O. D. Faugeras and R. Keriven. "Complete dense stereovision using level set methods". In: *Proceedings of European Conference on Computer Vision - ECCV'98*, pp. 379–393, Springer-Verlag, London, UK, 1998.

[Fitz 98] A. Fitzgibbon and A. Zisserman. "Automatic 3D model acquisition and generation of new images from video sequences". In: *Proceedings of European Signal Processing Conference - EUSIPCO '98*, pp. 1261–1269, Rhodes, Greece, 1998.

[Fors 03] D. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice-Hall, 2003.

[Frak 92] W. B. Frakes and R. A. Baeza-Yates, Eds. *Information retrieval: data structures & algorithms*. Prentice-Hall, 1992.

[From 95] T. Fromherz and M. Bichsel. "Shape from multiple cues: integrating local brightness information". In: *Fourth International Conference for Young Computer Scientist - ICYCS'95*, pp. 855–862, Beijing, P. R. China, 1995.

[Fua 95a] P. Fua and Y. G. Leclerc. "Object-centered surface reconstruction: combining multi-image stereo and shading". *International Journal of Computer Vision*, Vol. 16, No. 1, pp. 35–55, 1995.

[Fua 95b] P. Fua. "Reconstructing complex surfaces from multiple stereo views". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'95*, pp. 1078–1085, 1995.

[Gort 96] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. "The lumigraph". In: *Proceedings of SIGGRAPH'96*, pp. 43–54, ACM Press, New York, NY, USA, 1996.

[Gros 02] E. Grossmann and J. Santos-Victor. "Maximum likelihood 3D reconstruction from one or more images under geometric constraints". In: P. L. Rosin and A. D. Marshall, Eds., *Proceedings of the British Machine Vision Conference - BMVC'02*, pp. 343–351, British Machine Vision Association, 2002.

[Hart 00] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Press Syndicate of the University of Cambridge, 2000.

[Hema 03] E. Hemayed. "A survey of camera self-calibration". In: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance - AVSBS'03*, pp. 351–357, 2003.

[IEEE 90] IEEE, Ed. *IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries*. Institute of Electrical and Electronics Engineers, New York, NY, 1990.

[Illi 88] J. Illingsworth and J. Kittler. "A survey of the hough transform". *Computer Vision, Graphics and Image Processing*, Vol. 44, No. 1, pp. 87–116, 1988.

[Javi 02] B. Javidi. *Three-dimensional television, video and display technology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[Jung 02] F. Jung, V. Tollu, and N. Paparoditis. "Extracting 3D edgel hypothesis from multiple calibrated images: a step towards the reconstruction of curved and straight object boundary lines". In: *Proceedings of Photogrammetric Computer Vision - PCV'02*, pp. 100–104, Graz, 2002.

[Kahl 03] F. Kahl and J. August. "Multiview reconstruction of space curves.". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'03*, pp. 1017–1024, 2003.

[Kana 89] K. Kanatani and T.-C. Chou. "Shape from texture: general principle". *Artificial Intelligence*, Vol. 38, No. 1, pp. 1–48, 1989.

[Kana 94] T. Kanade and M. Okutomi. "A stereo matching algorithm with an adaptive window: theory and experiment". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 9, pp. 920–932, 1994.

[Kana 97] T. Kanade, P. Rander, and P. Narayanan. "Virtualized reality: constructing virtual worlds from real scenes". *IEEE Multimedia, Immersive Telepresence*, Vol. 4, No. 1, pp. 34–47, January 1997.

[Kawa 01] K. Kawamoto and A. Imiya. "Detection of spatial points and lines by random sampling and voting procedure". *Pattern Recognition Letters*, Vol. 22, No. 2, pp. 199–207, February 2001.

[Kiri 98] S. Kirihara and H. Saito. "Shape modeling from multiple view images using gas". In: *Proceedings of Asian Conference on Computer Vision - ACCV'98*, pp. 448 – 454, January 1998.

[Klau 03] A. Klaus, J. Bauer, and K. Karner. "An edgel-based two step approach for simultaneous matching of image sequences". In: *Proceedings of the 27th Workshop of the Austrian Association for Pattern Recognition*, pp. 51–57, Laxenburg, Austria, June 2003.

[Koha 98] R. Kohavi and F. Provost. "Glossary of terms". *Machine Learning: Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Vol. 30, pp. 271–274, 1998.

[Kutu 00] K. N. Kutulakos and S. M. Seitz. "A theory of shape by space carving". *International Journal of Computer Vision*, Vol. 38, No. 3, pp. 199–218, 2000.

[Laur 94] A. Laurentini. "The visual hull concept for silhouette-based image understanding". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, pp. 150–162, February 1994.

[Li 03a] M. Li, M. Magnor, and H. Seidel. "Hardware-accelerated visual hull reconstruction and rendering". In: *Proceedings of Graphics Interface 2003*, 2003.

[Li 03b] M. Li, M. A. Magnor, and H.-P. Seidel. "Improved hardware-accelerated visual hull rendering". In: *Proceedings of the Vision, Modeling, and Visualization Conference - VMV 2003*, pp. 151–158, 2003.

[Lu 04] Y. Lu, J. Zhang, Q. Wu, and Z. Li. "A survey of motion-parallax-based 3-D reconstruction algorithms". *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 34, No. 4, pp. 532–548, 2004.

[Marr 80] D. Marr and E. Hildreth. "Theory of edge detection". In: *Proceedings of the Royal Society of London*, pp. 187–217, 1980.

[Mart 03] D. Martinec and T. Pajdla. "Line reconstruction from many perspective images by factorization". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR03*, pp. 497–502, 2003.

[Mart 83] W. Martin and J. K. Aggarwal. "Volumetric descriptions of objects from multiple views". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 150–158, March 1983.

[Matu 00] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. "Image-based visual hulls". In: *Proceedings of SIGGRAPH'00*, pp. 369–374, July 2000.

[Matu 01] W. Matusik, C. Buehler, and L. McMillan. "Polyhedral visual hulls for Real-Time rendering". In: *Proceedings of 12th Eurographics Workshop on Rendering*, pp. 115–126, 2001.

[Matu 02] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. "Image-based 3D photography using opacity hulls". In: *Proceedings of SIGGRAPH'02*, pp. 427–437, July 2002.

[Matu 04] W. Matusik and H. Pfister. "3D tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes". *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 814–824, 2004.

[McLa 00] P. McLauchlan. "A batch/recursive algorithm for 3D scene reconstruction". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR00*, pp. 738–743, 2000.

[McMi 95] L. McMillan and G. Bishop. "Plenoptic modeling: an image-based rendering system". In: *Proceedings of SIGGRAPH'95*, pp. 39–46, ACM Press, New York, NY, USA, 1995.

[Mell 96] J. Mellor, S. Teller, and T. Lozano-Pérez. "Dense depth maps from epipolar images". Tech. Rep. AIM-1593, MIT, 1996.

[Mend 00] P. R. S. Mendonça, K.-Y. K. Wong, and R. Cipolla. "Camera pose estimation and reconstruction from image profiles under circular motion". In: *Proceedings of European Conference on Computer Vision - ECCV'00*, pp. 864–877, Springer-Verlag, London, UK, 2000.

[Mlay 03] A. Y. Mülayim, U. Yilmaz, and V. Atalay. "Silhouette-based 3-D model reconstruction from multiple images". *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 33, No. 4, pp. 582–591, August 2003.

[Mosc 98] F. Moscheni, S. Bhattacharjee, and M. Kunt. "Spatiotemporal segmentation based on region merging". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 20, No. 9, pp. 897–915, September 1998.

[Mull 00] J. Mulligan and K. Daniilidis. "View-independent scene acquisition for telepresence". In: *Proceedings of International Symposium on Augmented Reality*, October 2000.

[Mull 02] J. Mulligan, V. Isler, and K.Daniilidis. "Trinocular stereo: a real-time algorithm and its evaluation". *International Journal of Computer Vision*, Vol. 47, pp. 51–61, 2002.

[Nara 98] P. Narayanan, P. Rander, and T. Kanade. "Constructing virtual worlds using dense stereo". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'98*, pp. 3 – 10, January 1998.

[Okut 93] M. Okutomi and T. Kanade. "A multiple-baseline stereo". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 4, pp. 353–363, 1993.

[Otsu 97] K. Otsuka, T. Horikoshi, and S. Suzuki. "Image velocity estimation from trajectory surface in spatiotemporal space". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR97*, pp. 200–205, 1997.

[Peng 91] S. Peng. "Temporal slice analysis of image sequences". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR91*, pp. 283–288, 1991.

[Poel 94] C. J. Poelman and T. Kanade. "A paraperspective factorization method for shape and motion recovery". In: *Proceedings of European Conference on Computer Vision - ECCV'94*, pp. 97–108, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.

[Poll 00] M. Pollefeys, R. Koch, M. Vergauwen, B. Deknuydt, and L. van Gool. "Three-dimensional scene reconstruction from images". In: *Proceedings of the Conference on Three-Dimensional Image Capture and Applications II*, pp. 215–226, Bellingham, Washington, January 2000.

[Poll 99] M. Pollefeys, R. Koch, and L. V. Gool. "Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters". *International Journal of Computer Vision*, Vol. 32, No. 1, pp. 7–25, 1999.

[Potm 87] M. Potmesil. "Generating octree models of 3D objects from their silhouettes in a sequence of images". *Computer Vision, Graphics and Image Processing*, Vol. 40, No. 1, pp. 1–29, 1987.

[Prit 98] P. Pritchett and A.Zisserman. "Wide baseline stereo matching". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'98*, pp. 754–760, Bombay, India, January 1998.

[Rede 99] A. Redert, E. Hendriks, and J. Biemond. "Correspondence estimation in image pairs". *IEEE Signal Processing Magazine*, Vol. 16, No. 3, pp. 29–46, 1999.

[Robe 65] L. Roberts. "Machine perception of three-dimensional solids". In: J. Tippett, Ed., *Optical and Electro-Optical Information Processing*, pp. 159–197, Massachusetts Inst. of Technology Press, Cambridge, Massachusetts, 1965.

[Rodr 02] R. Rodrigues, A. R. Fernandes, K. van Overveld, and F. Ernst. "Reconstructing depth from spatiotemporal curves". In: *Proceedings of the 15th International Conference on Vision Interface*, pp. 252–259, Calgary, Canada, 2002.

[Rodr 04a] R. Rodrigues and A. R. Fernandes. "Accelerated epipolar geometry computation for 3D reconstruction using projective texturing". In: *Proceedings of the 20th spring conference on Computer Graphics SCCG '04*, pp. 200–206, ACM Press, New York, NY, USA, April 2004.

[Rodr 04b] R. Rodrigues and A. R. Fernandes. "Robust reconstruction of 3D points from images.". In: *Proceedings of IEEE International Conference on Image Processing - ICIP'04*, pp. 3017–3020, IEEE, October 2004.

[Rodr 04c] R. Rodrigues, A. R. Fernandes, K. van Overveld, and F. Ernst. "From spatiotemporal curves to reconstructed depth". *Image and Vision Computing*, Vol. 22, No. 12, pp. 959–970, October 2004.

[Roth 02] C. Rother and S. Carlsson. "Linear multi view reconstruction and camera recovery using a reference plane". *International Journal of Computer Vision*, Vol. 49, No. 2-3, pp. 117–141, 2002.

[Roth 03] C. Rother. "Linear multi-view reconstruction of points, lines, planes and cameras using a reference plane". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'03*, pp. 1210–1217, Nice, France, October 2003.

[Sain 02] M. Sainz, N. Bagherzadeh, and A. Susin. "Hardware accelerated voxel carving". In: *1st Ibero-American Symposium in Computer Graphics - SIACG 2002*, pp. 289–297, Guimarães, Portugal, July 2002.

[Sait 99] H. Saito and T. Kanade. "Shape reconstruction in projective grid space from large number of images". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR99*, pp. 49–54, Los Alamitos, June 1999.

[Sato 99] J. Sato and R. Cipolla. "Affine reconstruction of curved surfaces from uncalibrated views of apparent contours". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 21, No. 11, pp. 1188–1197, 1999.

[Scha 00] F. Schaffalitzky and A. Zisserman. "Planar grouping for automatic detection of vanishing lines and points". *Image and Vision Computing*, Vol. 18, No. 9, pp. 647–658, June 2000.

[Scha 02] D. Scharstein and R. Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". *International Journal of Computer Vision*, Vol. 47, No. 1-3, pp. 7–42, 2002.

[Scha 03] D. Scharstein and R. Szeliski. "High-accuracy stereo depth maps using structured light". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR03*, pp. 195–202, June 2003.

[Scha 96] D. Scharstein and R. Szeliski. "Stereo matching with non-linear diffusion". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR96*, pp. 343–350, IEEE Computer Society, Washington, DC, USA, 1996.

[Schm 00] C. Schmid, R. Mohr, and C. Bauckhage. "Evaluation of interest point detectors". *International Journal of Computer Vision*, Vol. 37, No. 2, pp. 151–172, 2000.

[Schm 97] C. Schmid and A. Zisserman. "Automatic line matching across views". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR97*, pp. 666–671, IEEE Computer Society, Washington, DC, USA, 1997.

[Se 03] S. Se and M. Brady. "Road feature detection and estimation". *Machine Vision and Applications*, Vol. 14, No. 3, pp. 157–165, 2003.

[Sega 92] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. "Fast shadows and lighting effects using texture mapping". In: E. E. Catmull, Ed., *Proceedings of SIGGRAPH'92*, pp. 249–252, ACM, 1992.

[Seit 95] S. Seitz and C. Dyer. "Complete scene structure from four point correspondences". In: *Proceedings of IEEE International Conference on Computer Vision - ICCV'95*, pp. 330–337, June 1995.

[Seit 97] S. M. Seitz and C. R. Dyer. "Photorealistic scene reconstruction by voxel coloring". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR97*, pp. 1067– 1073, 1997.

[Sequ 96] V. Sequeira, J. Gonçalves, and M. Ribeiro. "Active view selection for efficient 3D scene reconstruction". In: *Proceedings of International Conference on Pattern Recognition - ICPR'96*, pp. 815 – 819, August 1996.

[Shas 97] A. Shashua. "Trilinear tensor: the fundamental construct of multiple-view geometry and its applications". In: *Proceedings of the International Workshop on Algebraic Frames for the Perception-Action Cycle - AFPAC'97*, pp. 190–206, Springer-Verlag, London, UK, 1997.

[Shre 00] D. Shreiner, Ed. *OpenGL Reference Manual*. Addison-Wesley, third Ed., 2000.

[Slab 00] G. G. Slabaugh, T. Malzbender, and W. B. Culbertson. "Volumetric warping for voxel coloring on an infinite domain". In: *3D Structure from Multiple Images of Large-Scale Environments and Applications to Virtual and Augmented Reality*, pp. 109–121, 2000.

[Slab 01] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. "A survey of methods for volumetric scene reconstruction from photographs". In: K. Mueller and A. Kaufmann, Eds., *Proceedings of the Joint IEEE TCVG and Eurographics Workshop (VolumeGraphics-01)*, pp. 81–100, Wien, June 2001.

[Smit 95] S. Smith and J. Brady. "ASSET-2: real-time motion segmentation and shape tracking". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 814–820, 1995.

[Sonk 99] M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis and machine vision*. O'Reilly, 1999.

[Sriv 90] S. K. Srivastava. "Octree generation from object silhouettes in perspective views". *Computer Vision, Graphics and Image Processing*, Vol. 49, No. 1, pp. 68–84, 1990.

[Stei 00a] G. P. Stein and A. Shashua. "Model-based brightness constraints: on direct estimation of structure and motion". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, No. 9, pp. 992–1015, 2000.

[Stei 00b] E. Steinbach, B. Girod, P. Eisert, and A. Betz. "3-D reconstruction of real-world objects using extended voxels". In: *Proceedings of IEEE International Conference on Image Processing - ICIP'00*, pp. 138–141, 2000.

[Stur 96] P. F. Sturm and B. Triggs. "A factorization based algorithm for multi-image projective structure and motion". In: *Proceedings of European Conference on Computer Vision - ECCV'96*, pp. 709–720, Springer-Verlag, London, UK, 1996.

[Szel 98] R. Szeliski and R. Weiss. "Robust shape recovery from occluding contours using a linear smoother". *International Journal of Computer Vision*, Vol. 28, No. 1, pp. 27–44, 1998.

[Szel 99] R. Szeliski and P. Golland. "Stereo matching with transparency and matting". *International Journal of Computer Vision*, Vol. 32, No. 1, pp. 45–62, 1999.

[Tail 02] F. Taillandier and R. Deriche. "Reconstruction of 3d linear primitives from multiples views for urban areas modelisation". In: *Symposium on Photogrammetric Computer Vision*, Graz, September 2002.

[Tayl 95] C. J. Taylor and D. J. Kriegman. "Structure and motion from line segments in multiple images". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 17, No. 11, pp. 1021–1032, 1995.

[Tell 00] D. Tell and S. Carlsson. "Wide baseline point matching using affine invariants computed from intensity profiles". In: *Proceedings of European Conference on Computer Vision - ECCV'00*, pp. 814–828, 2000.

[Theo 04] C. Theobalt, J. Carranza, M. A. Magnor, and H.-P. Seidel. "3D video: being part of the movie". *ACM Computer Graphics*, Vol. 38, No. 3, pp. 18–20, 2004.

[Toma 92] C. Tomasi and T. Kanade. "Shape and motion from image streams under orthography: a factorization method". *International Journal of Computer Vision*, Vol. 9, No. 2, pp. 137–154, 1992.

[Torr 00] P. H. S. Torr and A. Zisserman. "Feature based methods for structure and motion estimation". In: W. Triggs, A. Zisserman, and R. Szeliski, Eds., *Proceedings of ICCV'99: International Workshop on Vision Algorithms*, pp. 278–294, Springer-Verlag, London, UK, 2000.

[Tsai 83] R. Tsai. "Multiframe image point matching and 3-D surface reconstruction". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 159–174, March 1983.

[Tsai 87] R. Y. Tsai. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses". *IEEE Journal of Robotics and Automation*, Vol. 3, No. 4, pp. 323–344, August 1987.

[Turk 94] G. Turk and M. Levoy. "Zippered polygon meshes from range images". In: *Proceedings of SIGGRAPH '94*, pp. 311–318, ACM Press, New York, NY, USA, 1994.

[Vail 92] R. Vaillant and O. D. Faugeras. "Using extremal boundaries for 3D object modeling". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, pp. 157–173, 1992.

[Veen 86] J. Veenstra and N. Ahuja. "Efficient Octree Generation from Silhouettes". In: *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition - CVPR86*, pp. 537–542, June 1986.

[Wilc 02] M. Wilczkowiak, E. Boyer, and P. Sturm. "3d modeling using geometric constraints: a parallelepiped based approach.". In: *Proceedings of European Conference on Computer Vision - ECCV'02*, pp. 221–237, May 2002.

[Witk 81] A. Witkin. "Recovering surface shape and orientation from texture". *Artificial Intelligence*, Vol. 17, No. 1-3, pp. 17–45, August 1981.

[Yang 02] R. Yang, G. Welch, and G. Bishop. "Real-time consensus-based scene reconstruction using commodity graphics hardware". In: *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications - PG'02*, p. 225, IEEE Computer Society, Washington, DC, USA, 2002.

[Yang 05] A. Y. Yang, K. Huang, S. Rao, W. Hong, and Y. Ma. "Symmetry-based 3-D reconstruction from perspective images". *Computer Vision and Image Understanding*, Vol. 99, No. 2, pp. 210–240, 2005.

[Zach 03] C. Zach, A. Klaus, B. Reitinger, and K. F. Karner. "Optimized stereo reconstruction using 3D graphics hardware". In: *Proceedings of the Vision, Modeling, and Visualization Conference - VMV 2003*, pp. 119–126, 2003.

[Zach 04] C. Zach, K. F. Karner, and H. Bischof. "Hierarchical disparity estimation with programmable graphics hardware.". In: *Proceedings of 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2004)*, pp. 275–282, 2004.

[Zhan 00] L. Zhang and S. Seitz. "Image-based multiresolution modeling by surface deformation". Tech. Rep. CMU-RI-TR-00-07, Carnegie Mellon University, January 2000.

[Zhan 99] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. "Shape from shading: a survey". *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 21, No. 8, pp. 690–706, 1999.

[Ziss 96] A. Zisserman, P. Beardsley, and P. Torr. "3D model acquisition from extended image sequences". In: *Proceedings of European Conference on Computer Vision - ECCV'96*, pp. 683–695, Springer-Verlag, 1996.

[Ziss 99] A. Zisserman, A. W. Fitzgibbon, and G. Cross. "VHS to VRML: 3D graphical models from video sequences". In: *IEEE International Conference on Multimedia and Systems, Florence*, pp. 51–57, June 1999.

[Zucc 02] M. Zucchelli, J. Santos-Victor, and H. I. Christensen. "Maximum likelihood structure and motion estimation integrated over time". In: *Proceedings of International Conference on Pattern Recognition - ICPR'02*, pp. 260–263, August 2002.

Index

A

additive blending, 90, 95
aliasing, 4
ambiguity, 6
area-based methods, *see* correlation-based methods
assigned point, 169

B

baseline, 6
binocular visual system, 2
block, 23
blocking artifacts, 24

C

c-point, 60
camera, 12
  calibration, 4
  center, 12
  extrinsics, 13
  intrinsics, 12
  obscura, 12
clamped, *see* clamping
classifier, 108
  performance, 105
clipped image plane, 13
closeness, 64
color channel masking, 96
color interpolation, 4
color-consistent, 40
commodity graphics hardware, 89
computer vision, 1
conditioning, 6
confusion matrix, 108
connectivity graph, 171
constructive solid geometry (CSG), 39
consumer-grade graphics hardware, 89
contingency table, *see* confusion matrix
contour
  detection, 53, 55
  extraction, 53, 55
  map, 55
  maps, 53
  paths, 78
  points, 78
contours
  apparent, 26, 54
corner points, 81
corners, 78
correct, 108
correctness, 108
correlation-based methods, 23
correspondence, 2, 4
  error, 18
  measure, 18
  problem, 4
cross voting, 68
cross-check threshold, 74
cross-checking, 73

D

depth, 14, 61
depth cues, 20
distance
  map, 64
  maps, 64
  metric, 65
  transforms, 64
distance mask, 65
distortions, 3

E

edge
  detection, *see* contour detection
  extraction, *see* contour extraction
  following, 56
edgels, 36
epipolar
  constraint, 18
  geometry, 18, 22
  image, 35
  line, 18
error function, *see* correspondence error
external camera parameters, *see* camera extrinsics
extremal boundaries, 26
extremities, 78, 80

F

false negatives, 108, 109
false positives, 108

feature points, 79
filtered contour map, 174
first derivative, 56
focal
    distance, *see* focal length
    point, *see* camera center
focal length, 12
frustum, 14
fundamental matrix, 23

G

Generalized Voxel Coloring (GVC), 42
gradient, 56
graphic cards (GC), 89
Graphics APIs (GAPI), 90
graphics hardware, 10
graphics processing unit (GPU), 89

H

highlights, 6
homogeneous coordinates, 93
hysteresis, 56

I

image discontinuities, 53
image plane, 12
incorrect, 108
incremental line reconstruction, 173
inferred visual hull, 38
interactive, 8
internal camera parameters, *see* camera intrinsics
invalid, 104
isolated points, 79, 80

J

junctions, 78, 80

L

layered depth images, 42
lens, 3
level of detail, 77, 82
light sensors, 3
line simplification, 81
luminance texture, 96

M

match function, *see* correspondence measure, 19
match penalty, *see* correspondence error
matching value, 18
Maximum Likelihood Estimation (MLE), 26
merging, 73
mixed-baseline, 6
motion estimation, 29

N

noise, 4
non-maximum suppression, 56
numerical instability, 6

O

obscuring edges, 26
occluding contours, 26
occlusion, 5
optical center, *see* camera center
optical flow, 29
overlapped points, 68

P

p-point, 15
parallax, 2
perspective effects, 6, 20
photo-consistent, 40
pinhole, 12
pinhole camera, 12
plateau, 69
point reconstruction, 9
point samples, 177
precision, 109
precision/recall charts, 110
principal point, 13
projected candidate, 60
projected point, 15
projection plane, *see* image plane
projective camera, 12
projector, 92

Q

quadrifocal tensor, 22, 27
quantization, 4

R

range, 61
recall, 109
reconstruction
    3D, 1
reflections, 6
refractions, 6
region-based methods, *see* correlation-based methods
repetitive patterns, 6
resolution, 4
rims, 27

S

sampling, 4

scene space, 28
set of candidate points, 60
SfM, *see* structure from motion
shape
    from focus, 20
    from shading, 20
    from shadows, 22
    from silhouette, 38
silhouette, 38
similar, 73
similarity measure, *see* correspondence measure
singular value decomposition, 33
smoothing, 56
source points, 134
space carving, 41
space sweep, 38
sparse, 25
spatiotemporal, 46
    volume, 29
SSSD-in-inverse-depth, 33
structure from motion, 22, 26, 29
structured lighting, 20, 22
surface voxel, 40
synchronized, 99

T

texture, 91
texture analysis, 20
texturing, 91
    projective, 90, 91
topological feature points, 78
trajectory surface, 46
triangulation, 3
trifocal tensor, 22, 27
trilinear tensor, 22
true negatives, 108, 109
true positive rate, 109
true positives, 108

U

usability, 8

V

valid, 104
validity, 108
videocube, 48
view ray, 15, 18
view-dependent, 9, 54
view-independent, 9
view-independent sources, 54
visual hull, 38
volumetric intersection, 38
vote maps, 64
vote percent threshold, 70
vote range, 67
voting, 60
voxel coloring, 38, 41
voxels, 28, 40
    empty space, 40

W

w-point, 14
weighted voting, 60
window, 23
world point, 14
world space, 11, 28
    formulation, 60

Z

Z-buffering, 107
zero crossings, 56