

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Miguel Matos

Development of an algorithm for counting vehicles and pedestrians based on video

Dissertation Report

December 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

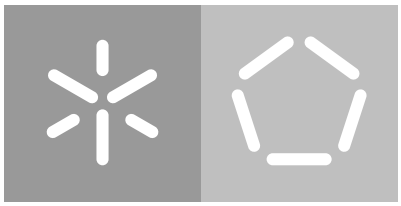


Atribuição-NãoComercial
CC BY-NC
https://creativecommons.org/licenses/by-nc/4.0/

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Miguel Matos

Development of an algorithm for counting vehicles and pedestrians based on video

Dissertation Report

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
António Ramires
André Leite Ferreira

December 2019

Thanks to the advisor António Ramires for the weekly meetings that allowed me to develop the project continuously and to the co-advisor André Ferreira to make available hardware and to help me to collect and think in features for use in the application of this work.

ABSTRACT

The population density in the urban environment has increased significantly, consequently increasing the number of vehicles and people on the public road. Possible monitoring of this flow allows better problem management, and the enhancement of solutions in a smart city context, solutions that promote regular traffic in a city.

This work presents a solution for counting vehicles and people in a video to use the solution developed in cities of Portugal. The solution combines deep learning techniques and traditional computer vision techniques, combining object detection, classification, object tracking, and fingerprint concepts. For each concept is presented the state of the art techniques and techniques used in similar problems.

To choose the best fingerprint methods, a comparative study of different techniques was produced. With a dataset of vehicle and people images, the following techniques were considered: Fourier Transform, Scale Invariant Feature Transform (SIFT), Color Co-occurrence Histogram (CCoH), and Autoencoders, of which CCoH showed better results.

The solution pipeline consists of the YOLOv3 algorithm for the object detection part, having the algorithm a convolutional neuronal network for object classification; Kalman Filter for object tracking was chosen in conjunction with the CCoH technique for object fingerprint. The pipeline ends with the matching of the newly detected objects with previously detected objects, using the Hungarian algorithm for this correspondence.

In order to extract features using the defined pipeline, a python library has been developed, allowing visualization of its operation and easy integration with video sources (video files and cameras). Object counting, area definition, line intersection, heatmap's, and object collision are examples of features that can be obtained by the library.

As a global solution, a web application was developed, including a frontend application, a backend, a relational database, and a service to perform video processing with the help of the developed library. The web application is in use and in a production environment.

Key-words: Computer Vision, Deep Learning, Object Detection, Object Fingerprint, Object Tracking.

RESUMO

A densidade populacional em contexto urbano tem aumentado significamente, aumentando assim, o número de veículos e pessoas na via pública. Uma possível monitorização deste fluxo permite uma melhor gestão do problema e a potenciação de soluções em um contexto de *smart city*, soluções que promovam o normal tráfego em uma cidade.

Este trabalho apresenta uma solução para a contagem de veículos e pessoas em vídeo com o objectivo de utilizar a solução desenvolvida em cidades de Portugal. A solução combina técnicas de *deep learning* e técnicas tradicionais de visão por computador, combinando conceitos de detecção de objetos, classificação, *tracking* de objectos e *fingerprint*, sendo que para cada conceito são apresentadas as técnicas estado de arte e as técnicas utilizadas em problemas semelhantes.

Para escolher os melhores métodos de *fingerprint* um estudo comparativo de várias técnicas foi desenvolvido. Com um dataset de imagens de veículos e pessoas foram consideradas as técnicas: *Fourier Transform*, *Scale Invariant Feature Transform* (SIFT), *Color Co-occurrence Histogram* (CCoH) e *Autoencoders*, dos quais o CCoH apresentou melhores resultados.

A pipeline da solução é composta pelo algoritmo *YOLOv3* para a parte de detecção de objectos, tendo este algoritmo uma rede neuronal convolucional para a classificação de objectos; foi escolhido o *Kalman Filter* para tracking de objectos em conjunto com a técnica de CCoH para *fingerprint* de objectos. A pipeline termina com a correspondência dos novos objectos detectados com objectos previamente detectados, sendo utilizado o algoritmo húngaro para esta correspondência.

De modo a extrair *features* com o uso da pipeline definida, foi desenvolvida uma biblioteca python que permite a visualização do seu funcionamento e uma fácil integração com fontes de vídeo (ficheiros de vídeo e cameras). Contagem de objectos, definição de áreas, intersecção de linhas, *heatmap's* e a colisão de objectos são, entre outras, exemplos de features que podem ser obtidas pela biblioteca.

Como solução global foi desenvolvida uma aplicação web, englobando assim, uma aplicação *frontend*, uma *backend*, uma base de dados relacional e um serviço para executar o processamento de vídeo com o auxílio da biblioteca desenvolvida. A aplicação web encontra-se em utilização e em ambiente de produção.

Palavras-chave: *Deep Learning*, Detecção de Objectos, Monitorização de Objectos, *Object Fingerprint*, Visão por Computador.

ACRONYMS

CCoH Color Co-occurrence Histogram.
CNN Convolutional Neural Network.

Faster R-CNN Faster Region-based Convolutional Neural.
FT Fourier Transform.

HSV Hue, Saturation and Value.

KF Kalman Filter.

Mask R-CNN Mask Region-based Convolutional Network.
MEI Mestrado em Engenharia Informática.

NASNet Neural Architecture Search Net.
NMS Non-maximum suppression.

PCA Principal Components Analysis.
PF Particle Filter.
PSO Particle Swarm Optimization.

RANSAC Random Sample Consensus.
RGB Red, Green and Blue.
RNN Recurrent Neural Network.
ROI Region of interest.

SIFT Scale-invariant feature transform.
SSD Single-Shot Detector.
SURF Speeded up robust features.
SVM Support Vector Machine.

UM Universidade do Minho.

YOLO You Only Look Once.

CONTENTS

1 INTRODUCTION 1
1.1 Context 1
1.2 Motivation 1
1.3 Objectives 2
1.4 Work structure 2
2 STATE OF THE ART 4
2.1 Object Classification 4
2.1.1 Viola-Jones 4
2.1.2 Scale-Invariant Feature Transform 6
2.1.3 Speeded Up Robust Features 8
2.1.4 Support Vector Machines 9
2.1.5 Convolutional Neural Networks 12
2.1.6 Summary 17
2.2 Object Detection 17
2.2.1 Faster Region-based Convolutional Network 19
2.2.2 You Only Look Once 21
2.2.3 Single-Shot Detector 23
2.2.4 Neural Architecture Search Net 25
2.2.5 Mask Region-based Convolutional Network 26
2.2.6 Summary 27
2.3 Object Tracking 28
2.3.1 Kalman Filter 28
2.3.2 Particle Filter 29
2.3.3 Mean Shift 31
2.3.4 Particle Swarm Optimization 32
2.3.5 Recurrent Neural Networks 33
2.3.6 Summary 35
2.4 Object Fingerprint 35
2.4.1 Colour Co-occurrence Histograms 36
2.4.2 Color Invariant SIFT 37
2.4.3 PCA-SIFT 38
2.4.4 Fourier Transform 38
2.4.5 Autoencoders 40
2.4.6 Summary 40

2.5 Related work 41
2.5.1 Surveillance 42
2.5.2 Sports 42
2.5.3 Games 43
3 THE PROBLEM AND ITS CHALLENGES 45
3.1 The problem 45
3.2 Challenges 46
3.3 Proposed approach 47
4 DEVELOPMENT 48
4.1 Decisions 48
4.2 Fingerprint Study 49
4.3 Library 59
4.3.1 Features 59
4.3.2 Project Structure 64
4.3.3 Library distribution 64
4.4 Web application 64
4.5 Summary 67
5 RESULTS 74
5.1 Experiment setup 74
5.2 Results 74
5.3 Summary 76
6 CONCLUSION 77
6.1 Conclusions 77
6.2 Prospect for future work 78

LIST OF FIGURES

Figure 1 Face detection with Viola-Jones 5
Figure 2 Output presented by the authors 6
Figure 3 An example using SIFT to locate one image inside another 7
Figure 4 SIFT descriptor example 8
Figure 5 SURF descriptor 9
Figure 6 Support Vector Machines on 2 classes separating problem, each class with 2 variables (x and y) 10
Figure 7 Kernel trick 11
Figure 8 Image classification with an SVM 11
Figure 9 Array of RGB Matrix 12
Figure 10 Example of an architecture of a CNN for image classification 13
Figure 11 Convolution 13
Figure 12 Common filters 14
Figure 13 Pooling layer 15
Figure 14 Max Pooling 15
Figure 15 Artificial Neural Network 16
Figure 16 Neural Network data flow example 17
Figure 17 Samples of annotated images in the MS COCO dataset 18
Figure 18 Examples of detection with the IoU value 19
Figure 19 Faster R-CNN is a single, unified network for object detection 20
Figure 20 Test-time speed comparision with predecessors algorithms 21
Figure 21 Example detections using RPN proposals on PASCAL VOC 2007 test 21
Figure 22 YOLO pipeline methods 22
Figure 23 YOLO CNN Architecture 23
Figure 24 Multiple Bounding Boxes for Localization (loc) and Confidence (conf) 24
Figure 25 SSD (Top) vs YOLO (Bottom) 24
Figure 26 SSD results on MS COCO 25
Figure 27 Example object detection using Faster-RCNN with NASNet 25
Figure 28 Mask R-CNN results on the COCO test set 26
Figure 29 Mask R-CNN architecture 27
Figure 30 Kalman filter steps 29
Figure 31 Example of a particle filter for localization 30

Figure 32 Next iteration of the previous example 30
Figure 33 Predicted, measured and corrected states of particle filter 31
Figure 34 Mean shifting to the *mode* 32
Figure 35 Mean-shift tracking 32
Figure 36 PSO 33
Figure 37 RNN are neural networks with loops to persist information 34
Figure 38 Results of an LSTM on object tracking 35
Figure 39 Color histogram on biological cells 36
Figure 40 Colour Co-occurrence histogram, for a vehicle image, with radius 20 pixels and 59 colour clusters 37
Figure 41 Example of SIFT 37
Figure 42 Previous image with Color Invariant SIFT 38
Figure 43 A comparison between SIFT and PCA-SIFT 39
Figure 44 Fourier Transform applied to letters 39
Figure 45 Autoencoders 40
Figure 46 Autoencoders architceture 41
Figure 47 Chinese surveillance system 42
Figure 48 Sports video analysis system 43
Figure 49 Video analysis on a video game 44
Figure 50 People counting on a crowd scene 46
Figure 51 Similarity with the current and next instance of the same object 51
Figure 52 Similarity between the first instance and next ones of the same object along time 51
Figure 53 Confusion matrix co-occurrence color histogram, limited to ten objects 52
Figure 54 Confusion matrix co-occurrence color histogram with threshold, limited to 10 objects 52
Figure 55 SIFT similarity with the current and next instance of the same object 53
Figure 56 SIFT similarity with the current and next instance of the same object 54
Figure 57 Confusion Matrix between objects using SIFT 54
Figure 58 Confusion matrix co-occurrence color histogram with threshold, limited to 10 objects 55
Figure 59 Fourier transform similarity with the current and next instance of the same object 56
Figure 60 Similarity with the current and next instance of the same object 56
Figure 61 Confusion Matrix between objects using Fourier Transform 57

Figure 62 Confusion Matrix between objects using Fourier Transform 57
Figure 63 Detection boxes library feature 60
Figure 64 Heatmap library feature 61
Figure 65 Panorma stiching pipeline 62
Figure 66 Panorma stiching example; on the top row the two images to stich; on the middle row the matching descriptors; on the bottom row the panorama stich 63
Figure 67 Two cameras setup on a bus for a panorama stiching 64
Figure 68 Library structure 65
Figure 69 Web application architecture diagram 66
Figure 70 Use Case diagram of the web application 68
Figure 71 Database diagram of the web application 69
Figure 72 Algorithm pipeline 70
Figure 73 Non-Maxima Suppression example 71
Figure 74 Results on oxford pedestrians video 75
Figure 75 Results on highway vehicles video 75

LIST OF TABLES

Table 1 Color Co-occurrence Histogram results in different environments 50
Table 2 Time measurement of using CCoH for image similarity between two images 53
Table 3 Time measurement of using SIFT for image similarity between two images 55
Table 4 Time measurement of using Fourier Transform for image similarity between two images 58
Table 5 Python lisbrary dependencies and version 72
Table 7 Web application components 73
Table 8 Results on pedestrians videos 74
Table 9 Results on car videos 76

1

INTRODUCTION

This chapter presents the context of the problem, the potentialities of the solution on the motivation section, the work objectives, and the whole work structure.

1.1 CONTEXT

Traffic in the city is a challenge for mobility. The population density in the urban context has grown substantially, increasing the number of vehicles in metropolitan areas. Road capacity has grown relatively slowly when compared to the rate of increase in traffic. Public transport systems help to minimize this problem but are not yet a definitive solution to the problem. Another factor is the limited ability to manage traffic, which results primarily from the lack of information on the number of vehicles and pedestrians in metropolitan areas.

A smart city has as its fundamental task the recognition of surrounding objects for the extraction of features, the methods used for the feature extraction are continuously evolving, being presented a state of the art methods on this work.

Data about the evolution of public areas and the impact of construction works is challenging to obtain autonomously, having continuous well-structured data about these areas promotes well-informed decisions for a mobility environment evolution.

1.2 MOTIVATION

The counting of the number of people and vehicles attending different spaces offers the knowledge necessary for the understanding of possible problems and the presentation of potential solutions related to the intensity of road and pedestrian traffic. As, for example, know the number of people who go to a specific street for the installation of a service or a store.

Correct detection of pedestrians and cars in real-time on a city environment also enables traffic monitorization, knowing the traffic flow rate (for example, how many people entered

a region and how much leave) and movement detection (for example, to identify traffic jam occurrences).

The video monitorization of city areas increase the statistical data about the city and with proper feature extraction, it's security, contribution to the evolution of the smart city environment.

1.3 OBJECTIVES

Counting the number of people on video presents the following challenges:

- The classification of objects (vehicles and pedestrians) in images, to be able to recognize with high probability the presence of a person or vehicle in the image;
- The detection of objects in an image, multi-object detection in an image for classification appliance;
- The tracking of vehicles and people identified, being able to follow the route of objects by the same discipline in different sessions of object detection;
- How to distinct objects detected from previously detected objects.

The objectives for this work are:

- Identify the best deep learning and traditional computer vision techniques for detection and tracking of vehicles and pedestrians on video;
- Develop a useful solution for counting pedestrians and vehicles in a city environment, allowing traffic monitorization on pedestrians and vehicles.;

1.4 WORK STRUCTURE

This work is organized as follows:

- In chapter 1 is presented an introduction to the topic, where the problem related to counting vehicles and pedestrian is firstly approached;
- Chapter 2 is composed by the state of the art, where a set of deep learning methods and traditional computer vision techniques are presented for the right comprehension of the work. It is also presented some related use cases where components of the solution to present are applied. All of these are divided into appropriate sections;
- In chapter 3, is presented the problem and it's challenges;

- The implementation of the system is presented in chapter 4. Presenting the system development, the features and the fingerprint study;
- Chapter 5 presents the results obtained from the model developed in Chapter 4;
- Finally, in chapter 6, the conclusions of the implemented system are presented, as well as some proposals for the evolution of the system.

2

STATE OF THE ART

This chapter presents the state of the art methods for each component. Related systems are also presented in the final section of the chapter.

The following sections will present methods to the main component of the system:

- Object classification, the attribution of a label to an image answering to the question of what object is present in this image;
- Object detection, the discovery of an object in an image;
- Object tracking, after objects being detected what objects corresponds to objects previously detected;
- Object fingerprint, attribution of a description to the object detected for association with previous objects detected.

2.1 OBJECT CLASSIFICATION

This section has presented the state of the art methods for object classification on an image. Object classification is the attribution of a label to an image. The label, on this work, it will be a person or a vehicle.

For the given problem of this work, the classification has to be fast (enabling real-time classification), being this the only restriction.

In this section, some methods perform already detection along with classification, given the nature of the method.

2.1.1 *Viola-Jones*

The Viola-Jones object detection solution was proposed in 2001 by Paul Viola and Michael JonesViola and Jones (2001). This approach is widely used for face detection and is a fast method that enables real-time object detection. Although it was motivated for face

detection, it can be trained to detect a variety of object classes. The following explanation would have the face as an example being also discussed the all body and vehicle appliance.

There are some problems on detection a face, one is that we don't know the size of the face in the image, it can be big, or it can be very small, the another problem is the efficiency, performing detection in a high-resolution image when the objective is to perform detection lots of times for second enabling real-time detection can be very complicated not using the right approach. For detecting faces, it has to be considered the ethnic groups, the differences of age, gender if the face has accessories like glasses, adding this complexity to the problem.

In the face detection problem and considering a greyscale instead of an RGB scale is know that the eyes are darker than the forehead, so, if the rectangle of pixels corresponding to the forehead is subtracted by the rectangle of the eyes zone pixels is expected that the subtraction indicates that the eyes zone is darker, after performing a list of subtractions like this in other parts of the image it can be said with high probability that a face was found. This case is demonstrated in 1.

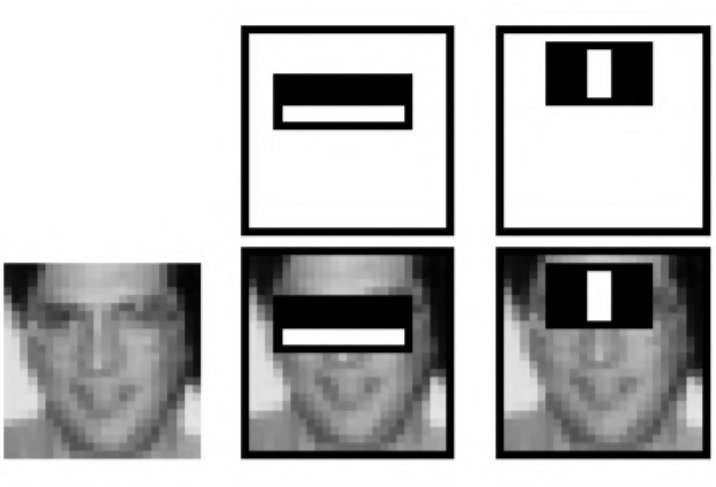


Figure 1: Face detection with Viola-Jones
Source: Viola and Jones (2001)

Viola-Jones identify what type of subtractions have to be applied in a trained phase and apply the subtractions on an image in a very efficient way. The substractions correspond to the features of the object to be detected. The trained phase employs a variant of the learning algorithm AdaBoost Freund and Schapire (1997) to both select the best features and to train classifiers that use them. The subtractions are applied in a cascade architecture where they must spend most time only on potentially positive sub-windows, i.e., only is applied a second subtraction if the first indicates that the potential sub-windows is a face. Before the classification phase, the image is pre-processed for the appliance of subtractions.

The output results are presented in 2.

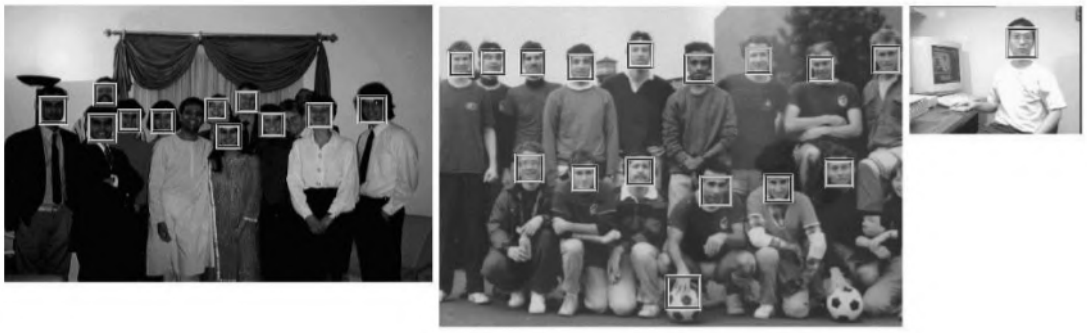


Figure 2: Output presented by the authors
Source: Viola and Jones (2001)

Performing this algorithm for people (full body considered) and vehicle detection as already been shown in real-time Viola et al. (2005),Xu et al. (2017). The results are acceptable, however the classifier is not robust enough to consider that the same object is presented but articulated in another form, like a person sideways (if it was trained with upfront people) unless a new classifier for different scenarios for people and vehicles be done the outputs could have lots of false negatives and it would damage the tracking and the counting problem.

2.1.2 *Scale-Invariant Feature Transform*

The scale-invariant feature transform (SIFT) is a computer vision method and is a feature detection algorithm to detect and describe local features in images. It was and published by David Lowe in 1999 Lowe (1999) and patented in Canada by the University of British Columbia.

We are considering that features are points of interest in the object that can be tracked. SIFT is a technique for feature detection and feature description. If we detect the correct features and track them, we can achieve several goals like object classification and detection. With an image for training, the useful features of the object can be stored. When viewing the test image, we can look for the same features. The detection of the same features gives us the recognized object as well as its location in the test image.

The features that detected in the object should not change unless the object changes. So, the pixel values might change when:

- The image is scaled;
- Brightness is changed;
- Noise increases.

The ideal features detected should be robust against changes in:

- Image scale: The object can be viewed not only from different angles but also from different distance from the camera;
- Illumination;
- Noise.

The SIFT algorithm has four necessary steps. First is to estimate scale-space extrema using the Difference of Gaussian (DoG). Secondly, a key point localization, where the key point candidates are localized and refined by eliminating low contrast points. Thirdly, a key point orientation assignment based on local image gradient and lastly a descriptor generator to compute the local image descriptor for each key point based on image gradient magnitude and orientation. The results are presented in 3.



Figure 3: An example using SIFT to locate one image inside another
Source: https://mi.eng.cam.ac.uk/~cipolla/lectures/4F12/Slides/old/4F12-SIFT-extra-material.pdf

Feature description, 4, is the unique and robust representation of the pixel that makes up the feature. They are stored in a database, which is then used to compare with test image features for object recognition.

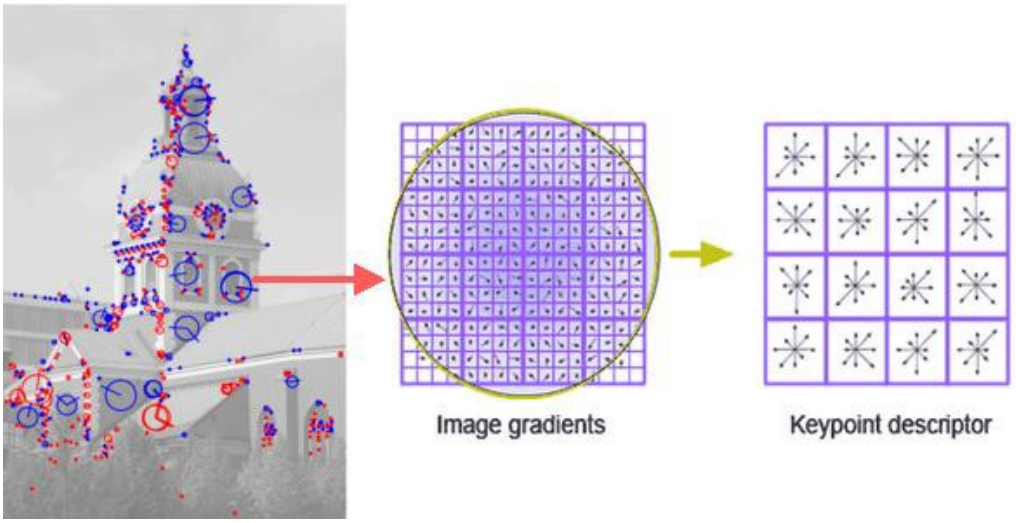


Figure 4: SIFT descriptor example
Source: Lowe (1999)

2.1.3 *Speeded Up Robust Features*

The speeded-up robust features (SURF) is a computer vision method and is a local feature detector and descriptor. It can be used for tasks such as object detection, classification, and 3D reconstruction and was presented in 2006 Bay et al. (2006) and is patented.

SURF was developed to improve the speed of a scale-invariant feature detector being SURF several times faster than SIFT. Instead of using the Difference of Gaussian approach, SURF uses the Hessian matrix approximation to detect interesting points and use the sum of Haar wavelet responses for orientation assignment Bay et al. (2006).

In short, SURF adds many features to improve the speed in every step. The analysis shows it is three times faster than SIFT, while performance is comparable to SIFT. SURF is good at handling images with blurring and rotation but not good at handling viewpoint change and illumination change.

SURF approximates the DoG with box filters. Instead of Gaussian averaging the image, squares are used for approximation since the convolution with a square is much faster if the integral image is used. Also, this can be done in parallel for different scales.

The SURF uses a BLOB detector, which is based on the Hessian matrix to find the points of interest. For orientation assignment, it uses wavelet responses in both horizontal and vertical directions by applying adequate Gaussian weights. An example result is presented at 5.

For feature description, SURF also uses the wavelet responses. A neighborhood around the key point is selected and divided into subregions, and then for each subregion, the wavelet responses are taken and represented to get the SURF feature descriptor. The sign

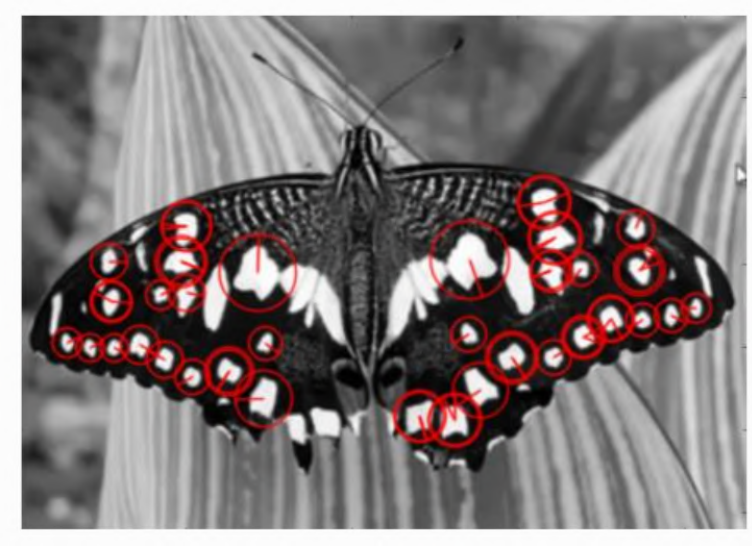


Figure 5: SURF descriptor
Source: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

of Laplacian, which is already computed in the detection, is used for underlying interest points. The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse case. In the case of matching the features are compared if they have the same type of contrast (based on a sign), which allows faster matching Bay et al. (2006).

2.1.4 *Support Vector Machines*

Distinguish persons from non-persons and cars from non-cars is the way the next method works.

Support Vector Machines (SVM) is a machine learning method (supervised learning) for classification and regression problems Cortes and Vapnik (1995).

Support Vector Machines (SVMs) have their roots in Statistical Learning Theory. They have been widely applied to machine vision fields such as character, handwriting digit and text recognition Joachims (1998), and satellite image classification Huang et al. (2002).

SVM can solve linear and non-linear problems. On a classification problem, our problem, the idea of SVM is to create a line which separates the classes. The goal of the line is to maximize the margin between the points on either side of the so-called decision line, 6. The benefit of this process is that after the separation, the model can easily guess the target classes (labels) for new cases.

One advantage of this method is that not need much data for training the model because the only considered cases are the ones in the boundary with another class.

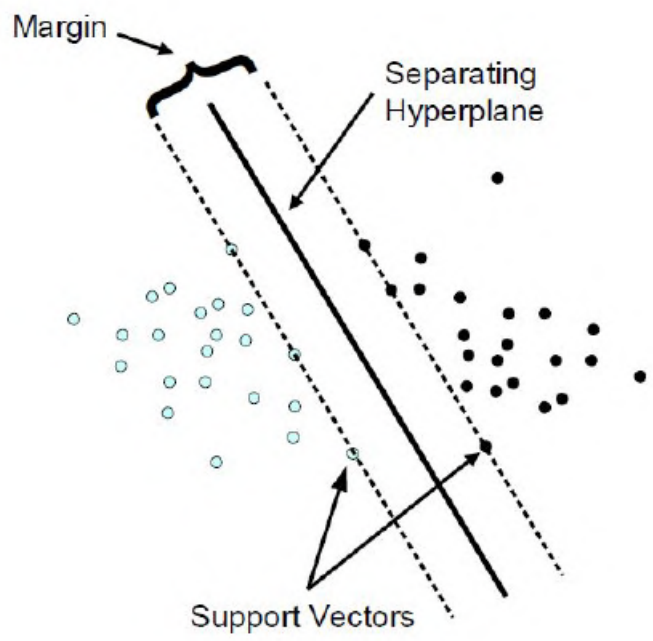


Figure 6: Support Vector Machines on 2 classes separating problem, each class with 2 variables (x and y)

Source: https://www.researchgate.net/figure/Classical-example-of-SVMs-linear-classifier_fig1_259079789

SVM always do a linear separation of the classes, but there are cases that the data is not linearly separable (in our n dimensional space), but maybe linearly separable in a higher-dimensional space. This high dimensional approach is being called the kernel trick presented at 7.

SVM classification is essentially a binary (two-class) classification technique, to perform classification on more than 2 classes there are two conventional methods, the methods are the one-against-one (1A1), comparing one class to another class for 3 classes we need a model for each two combinations (A against B, B against C and C against A) and a policy for choosing the best three given the classifications, the other techniques is the one-against-all (1AA), where is necessary one model for every class and on a 3 classes examples would it be A against B and C, B against A and C, and C against A and B Anthony et al. (2007).

For an SVM to classify pedestrians and cars on images, it needs to create a model based on data from the image. The phase where data is collected from the image is called the feature extraction phase. In this phase are extracted features from the image. The features can be extracted in various forms, could be HOG features Cao et al. (2011), SIFT Chatfield et al. (2011), Gabor Filters Sun et al. (2002), etc. These features are after used to train an SVM classifier.

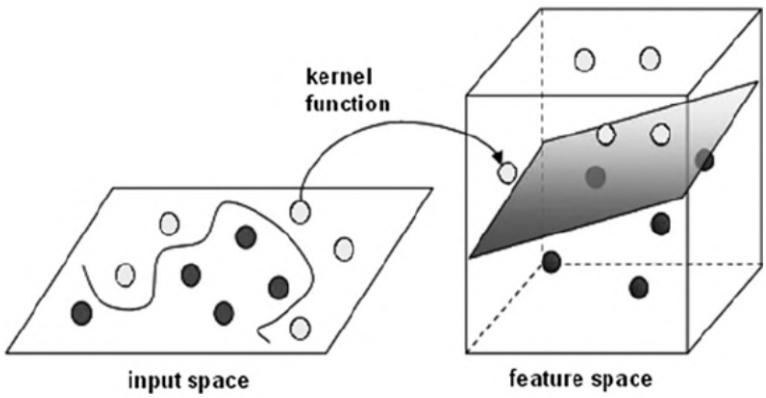


Figure 7: Kernel trick
Source: https://medium.com/@LSchultebraucks/introduction-to-support-vector-machines-9f8161ae2fcb

SVM requires a productive feature extraction phase for image classification on a multiclass problem, and this phase can be followed by other phases to collect max unique information about the image to train the SVM model, as seen in 8.

For example, given an image, the classification task is to decide whether an image is a pedestrian or a car. The image, before being into the SVM might have gone through some image processing filters so that some features might be extracted such as edges, color and shape. The model will be trained with this features enabling classification on new images.

SVM also needs to know what a non pedestrian and a non car is to perform a better classification.

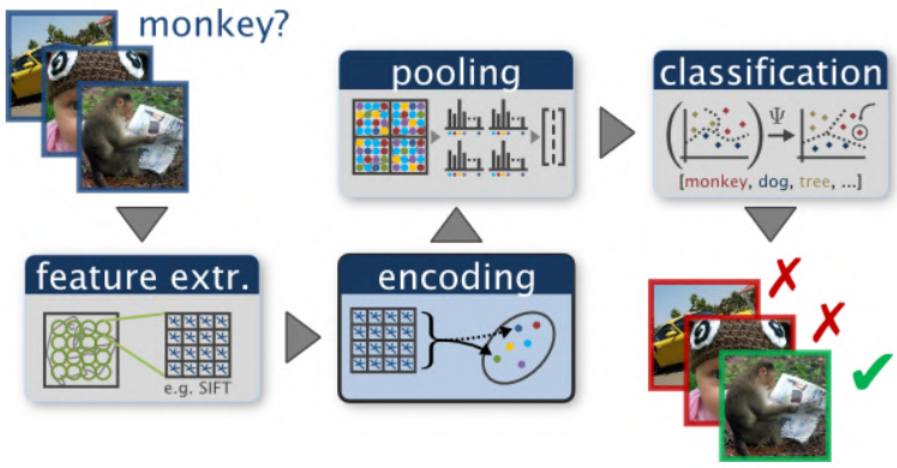


Figure 8: Image classification with an SVM
Source: Chatfield et al. (2011)

2.1.5 *Convolutional Neural Networks*

The following method is a powerful method on image classification and a very fundamental deep learning technique. It creates a model for classification given images and his associated label, learning itself (with tuned variables) what the image is.

CNN's are especially useful for image classification and recognition. They have two main parts: a feature extraction part and a classification part.

For image classification, CNN takes an input image, process it, and classify it under certain categories like pedestrian, car, bicycle, truck, and others. The image represented as an array of pixels. Based on the image resolution, it will see h x w x d values (h = Height, w = Width, d = Dimension). Eg., An image of 10 x 10 (10 pixels of height and 10 pixels of width) can have dimension = 3 if is a RGB image (being 10 x 10 x 3) or dimension 1 if is greyscale (being 10 x 10 x 1). The representation of an image can be found at 9.

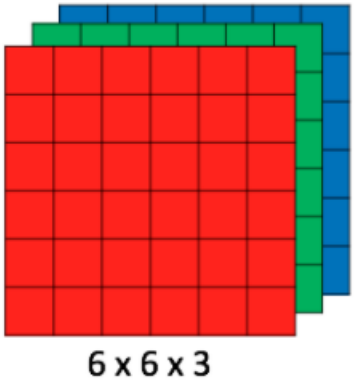


Figure 9: Array of RGB Matrix
Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

To classify an image, deep learning CNN models will first extract features from the image (corners, lines, shapes, among others), relate the features founded, and find the best possible label for the image. CNN is a powerful method because it will learn what features have to detect and how they are related to the label. For that, CNN has a training phase where it receives images and the givens labels to learn how to classify.

Technically, to train and test, each input image will pass through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC), and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

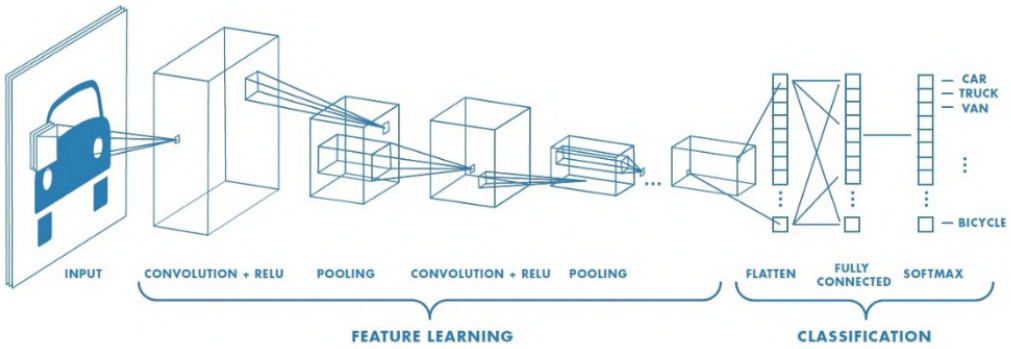


Figure 10: Example of an architecture of a CNN for image classification
Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Convolution Layer

The main special technique in CNNs is convolution, where a filter slides over the input and merges the input value plus the filter value on the feature map. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernal.

As gure 11 suggests, consider a 8 x 8 pixels image and a 3 x 3 filter. Then a convolution is applied to the image (with the given filter), the filter will be applied as in 10 for each 3 x 3 area of the image and the result is called a Feature Map, a matrix which values are the results of the convolution application.

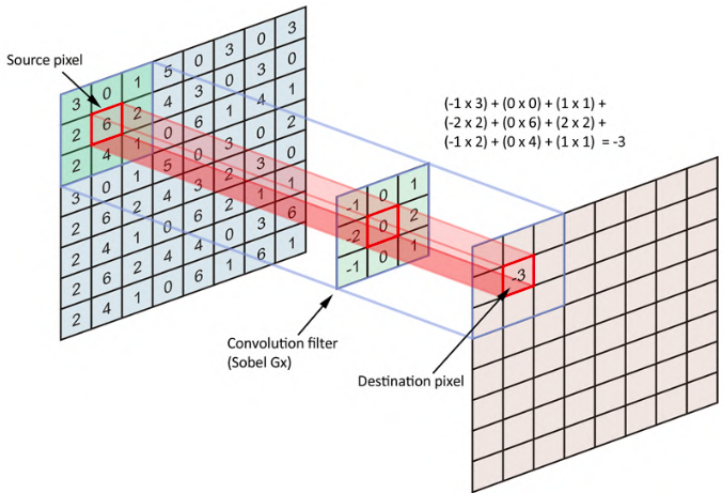


Figure 11: Convolution
Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

This example was a convolution operation shown in 2D using a 3x3 filter. However, in reality, convolutions are performed in 3D, a pixel matrix for each color, as in 11.

Convolution of an image with different filters can perform operations such as edge detection, blur, and sharpen by applying filters. The following image shows various convolution image after applying different types of filters (Kernels).

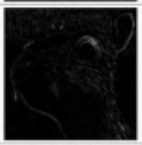
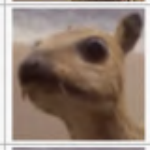
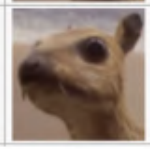
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 12: Common filters
Source: https://www.apsl.net/blog/2017/11/20/use-convolutional-neural-network-image-classification/

Pooling Layer

After a convolution operation, pooling is usually performed to reduce the dimensionality of the resulted matrix. This reduces the number of parameters, which both shortens

the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact, as in 13.

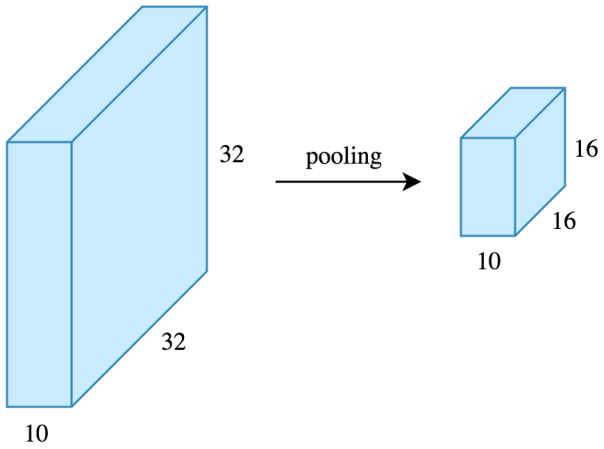


Figure 13: Pooling layer
Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Spatial pooling can be of different types:

- Max Pooling;
- Sum Pooling;
- Max Pooling.

The most common type of pooling is max pooling, 14, which takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input and takes the max value in the window. Similar to a convolution, we specify the window size and stride.

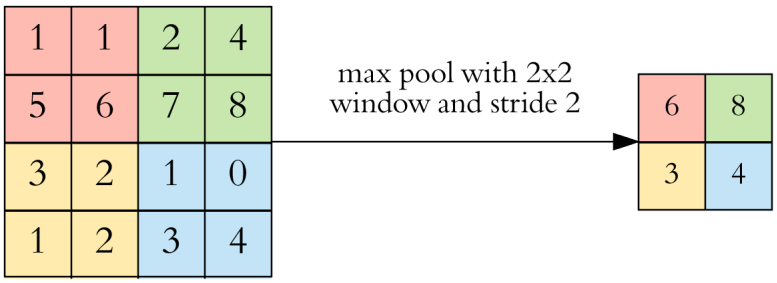


Figure 14: Max Pooling
Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling, which reduces the

dimensionality of each map but retains the relevant information. Spatial pooling can be of different types:

Max pooling takes the most significant element from the rectified feature map. Taking the most significant element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Fully Connected Layer

After the convolution and pooling layers are added, a couple of fully connected layers. This group of fully connected layers is a standard artificial neural network, such as 15.

The output of both convolution and pooling layers are 3D volumes. However, a fully connected layer expects a 1D vector of numbers. So the output is flattened of the final pooling layer to a vector, and that becomes the input to the fully connected layer. Flattening is merely arranging the 3D volume of numbers into a 1D vector.

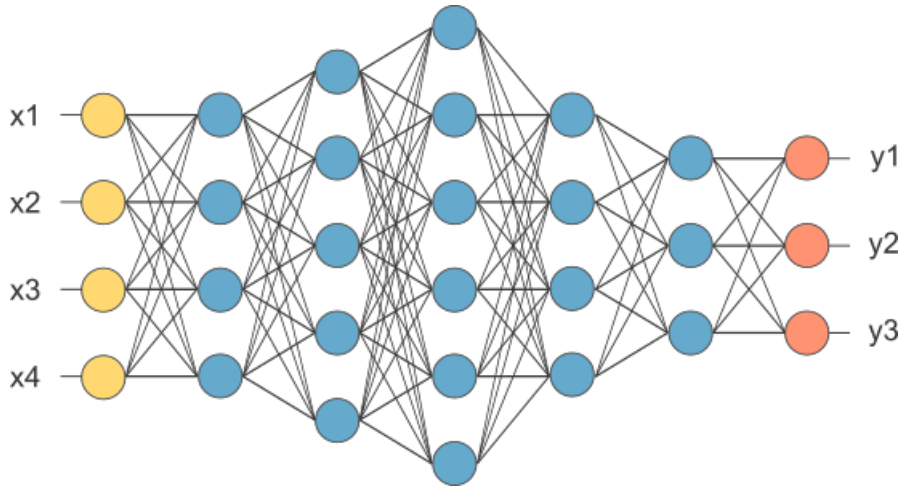


Figure 15: Artificial Neural Network
Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

In the above image, the feature map matrix will be converted as a vector (x1, x2, x3, ...). With the fully connected layers, we combined these features to create a model. With the model created and an activation function such as softmax is now possible to classify the outputs as a pedestrian, car, among others.

For a visual knowledge of how an artificial neural network works, is suggest the following demo https://playground.tensorflow.org/.

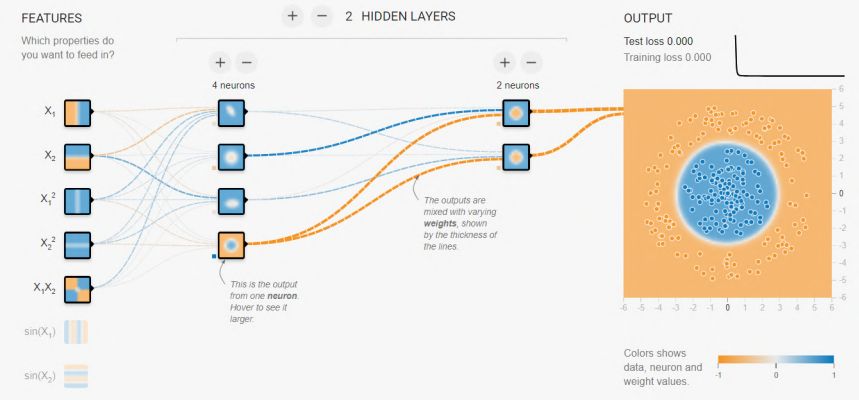


Figure 16: Neural Network data flow example
Source: https://playground.tensorflow.org/

2.1.6 *Summary*

The Viola-Jones is a fast inference method, to use this technique lots of cascade models would be necessary for the same object and each model would also have lots of features, given the ways that this types of objects can appear, it would be difficult to obtain a cascade model to have inference with success 95% of the cases.

The SIFT and SURF techniques are a solution for classification, but for these types of objects, the keypoints detected can not be matched with the expected key points.

The Support Vector Machine is a more robust technique compared with the previous ones and does not need lots of trained examples like the Convolutional Neural Networks. To be a multiple object classifier, multiple SVMs models would be necessary. This is not a big problem. However, the approach to solving this problem not seems to be the natural way to classify objects, which can show problems with unforeseen problems.

For detecting pedestrians and cars, the Convolutional Neural Networks is the best solution based on the complexity of the task, the CNN's after being trained with thousands of objects can learn high-level features and correlate them being a robust multiple object classifier.

2.2 OBJECT DETECTION

This section has presented the state of the art methods that enables object detection on video frames where the goal is to detect pedestrians and cars, allowing a real-time analysis. The methods presented perform segmentation/localization of objects in the image and classification, being able to change the classification method by another in some object detection methods. The expected result from the application of object detection is the attribution of a label to a region of an image, for example, on an image of a road with five cars and two

pedestrians are expected output with five regions with a label of a car and two regions with a pedestrian label. The region can be a box around the corresponded image or a more specific region with the exact form of the object.

The object detection challenge is, at the same time, a regression and a classification task. A regression task because it is needed to find the object in the image and the classification task to classify the object found. Usually, an object detection model outputs many mores boxes (regions in the images) than the actual/useful objects to detect. To have a high precision of detection, the boxes of more needs to be removed.

Several datasets have been released for object detection challenges the most referenced ones are PASCAL VOC VOC, COCO COCO and ImageNet ImageNet, being presented at 17. These datasets contain lots of labeled images, the COCO dataset contains about 200k labeled images and 80 object categories, being this large quantity of data extremely convenient for deep learning object detection models.



Figure 17: Samples of annotated images in the MS COCO dataset
Source: Lin et al. (2014)

The main terms of comparison between object detection methods will be the results presented on datasets and the computational cost time.

An important measurement to be considered on object detection is the Intersection over Union (IoU) area, 18, a value between 0 and 1. It corresponds to the overlapping area between the predicted box and the ground-truth box. The higher the IoU, the better the predicted location of the box for a given object.

Another measurement is the Average Precision (AP). The AP metric is a summary of the precision-recall curve. The commonly used metric used for object detection challenges is

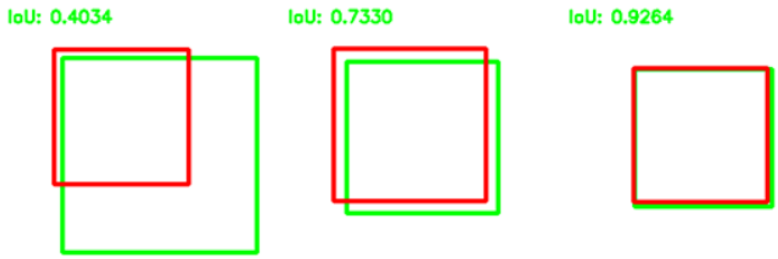


Figure 18: Examples of detection with the IoU value
Source: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

called the mean Average Precision (mAP). It is the mean of the Average Precisions computed over all the classes of the dataset. The mAP metric avoids to have extreme specialization in a few classes and thus weak performances in others.

The mAP score is usually computed for a fixed IoU, but a high number of bounding boxes can increase the number of candidate boxes. The COCO challenge has developed an official metric to avoid extra boxes creation. It computes a mean of the mAP scores for variable IoU values in order to penalize a high number of bounding boxes with wrong classifications.

2.2.1 *Faster Region-based Convolutional Network*

There are two main predecessors algorithms of the following method: R-CNN and Fast R-CNN. The predecessors use selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, In 2015, Ren et al. (2015) introduced Faster Region-based Convolutional Network (Faster R-CNN), an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

The image is provided as an input to a convolutional neural network (CNN), which provides a convolutional feature map. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using an RoI pooling layer, which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. The architecture is presented at 21

The best Faster R-CNNs have obtained mAP scores of 78.8% over the 2007 PASCAL VOC test dataset and 75.9% over the 2012 PASCAL VOC test dataset. They have been trained with PASCAL VOC and COCO datasets. One of these models is 34 times faster than the Fast R-CNN using the selective search method. Comparison with previous iterations are presented at 20.

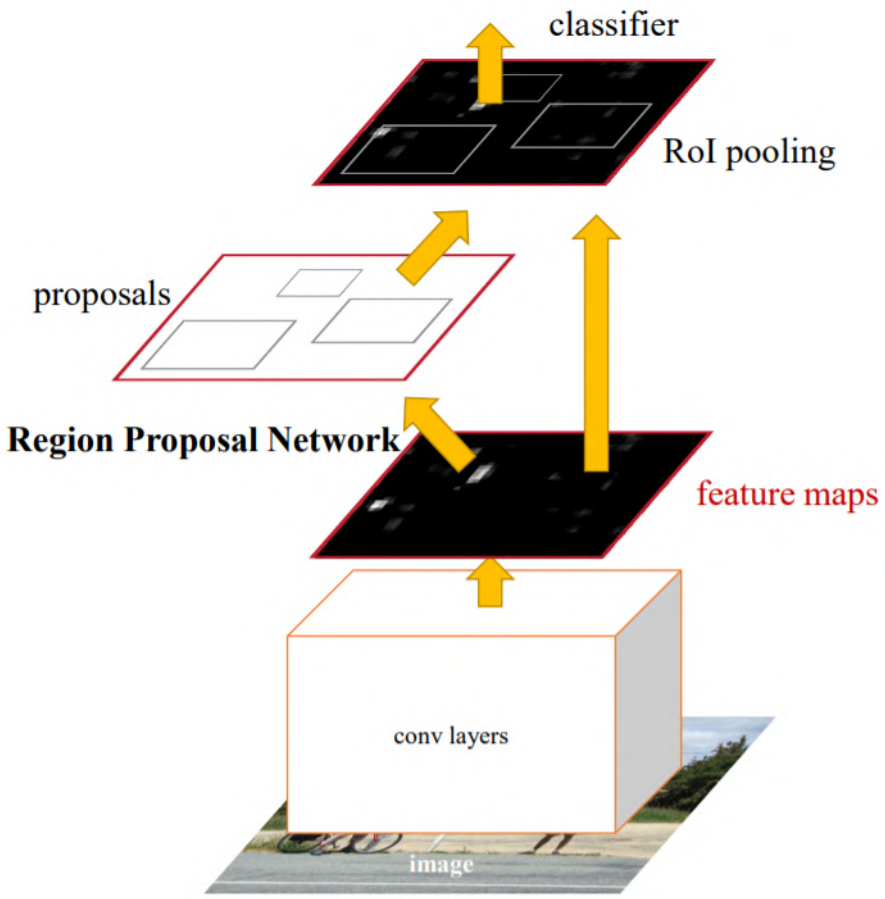


Figure 19: Faster R-CNN is a single, unified network for object detection
Source: Ren et al. (2015)

Faster R-CNN has two networks: region proposal network (RPN), for generating region proposals and a network using these proposals to detect objects. The main difference here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects.

Faster R-CNN uses RPN to avoid the selective search method; it accelerates the training and testing processes, and improve the performances. The RPN uses a pre-trained model over the ImageNet dataset for classification, and it is fine-tuned on the PASCAL VOC dataset. Then the generated region proposals with anchor boxes are used to train the Fast R-CNN. This process is iterative.

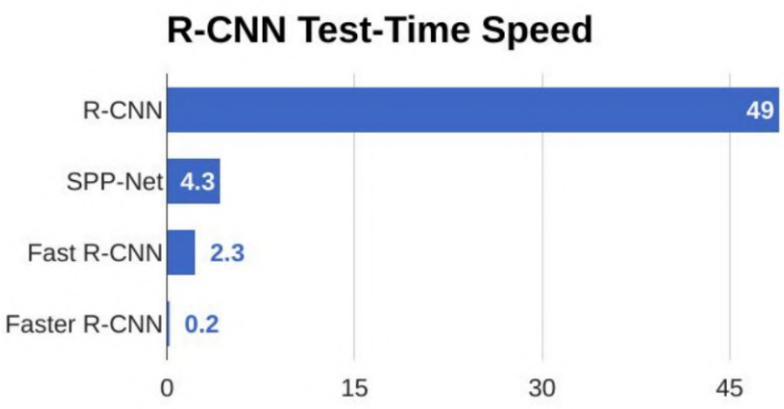


Figure 20: Test-time speed comparision with predecessors algorithms
Source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

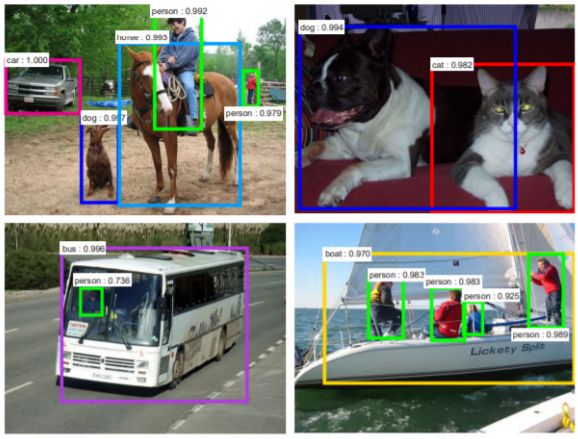


Figure 21: Example detections using RPN proposals on PASCAL VOC 2007 test
Source: Ren et al. (2015)

2.2.2 *You Only Look Once*

The YOLO model Redmon et al. (2016) directly predicts bounding boxes and class probabilities in a single evaluation with a single network. The simplicity of this model allows for real-time predictions.

Initially, the model takes an image as input. It divides it into an SxS grid. Each cell of this grid predicts B bounding boxes with a confidence score. This confidence score is the probability of detecting the object multiply by the IoU between the predicted and the ground truth boxes.

The CNN used is inspired by the GoogLeNet model Szegedy et al. (2015) introducing the inception modules. The network has 24 convolutional layers, followed by two fully-

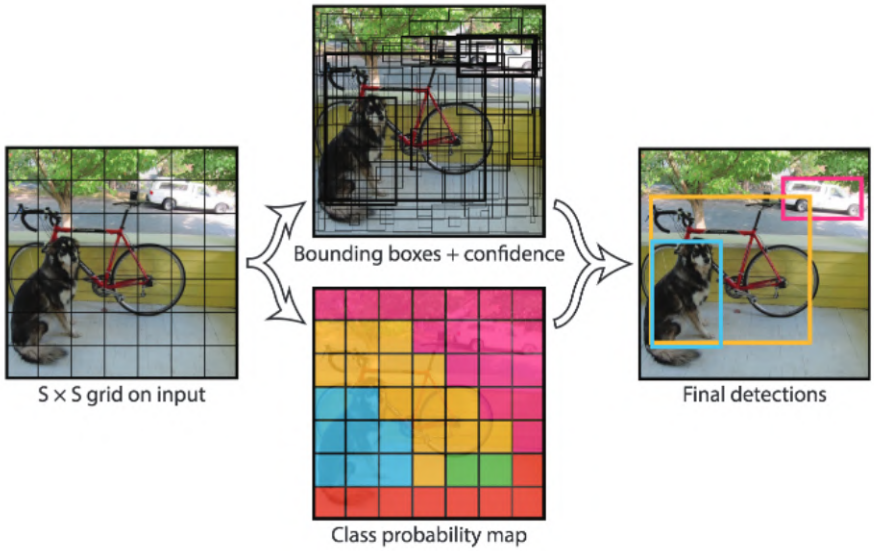


Figure 22: YOLO pipeline methods
Source: Redmon et al. (2016)

connected layers. Reduction layers with 1x1 filters, reducing the features space from the previous layers, followed by 3x3 convolutional layers replace the initial inception modules. The Fast YOLO model is a lighter version with only nine convolutional layers and fewer number of filters. Most of the convolutional layers are pre-trained using the ImageNet dataset with classification. Four convolutional layers followed by two fully-connected layers are added to the previous network, and it is entirely retrained with the 2007 and 2012 PASCAL VOC datasets.

The final layer outputs an S*S*(C+B*5) tensor corresponding to the predictions for each cell of the grid. C is the number of estimated probabilities for each class. B is the fixed number of anchor boxes per cell; each of these boxes is related to 4 coordinates (coordinates of the center of the box, width, and height) and a confidence value.

With the previous models, the predicted bounding boxes often contained an object. The YOLO model, however, predicts a high number of bounding boxes. Thus there are many bounding boxes without any object. The Non-Maximum Suppression (NMS) method is applied at the end of the network. It consists of merging highly-overlapping bounding boxes of the same object into a single one. The authors noticed that there are still a few false-positive detected.

The detection network has 24 convolutional layers, followed by two fully connected layers. Alternating 1 1 Convolutional layers reduce the features of space from preceding layers. We pre-train the convolutional layers on the ImageNet classification task at half the resolution (224 224 input image) and then double the resolution for detection.

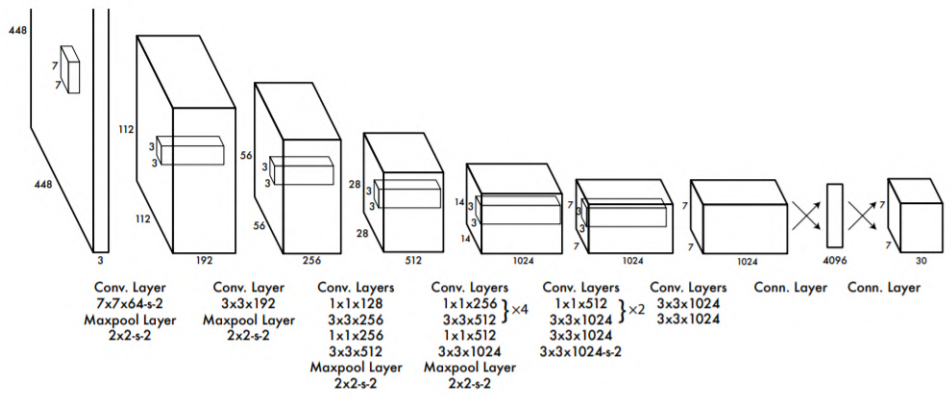


Figure 23: YOLO CNN Architecture
Source: Redmon et al. (2016)

The YOLO model has a 63.7% mAP score over the 2007 PASCAL VOC dataset and a 57.9% mAP score over the 2012 PASCAL VOC dataset. The Fast YOLO model has lower scores results, but they have both real-time performances.

2.2.3 *Single-Shot Detector*

Single-Shot Detector (SSD), similarly to YOLO, only is need to take one single shot to detect multiple objects within the image, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal Liu et al. (2016), being SSD faster compared with two-shot RPN-based approaches.

SSD predicts all the bounding boxes and the class probabilities with an end-to-end CNN architecture. The model takes an image as input, which passes through multiple convolutional layers with different sizes of filter (10x10, 5x5, and 3x3). Are used feature maps from convolutional layers at different positions of the network to predict the bounding boxes. They are processed by specific convolutional layers with 3x3 filters called extra feature layers to produce a set of bounding boxes similar to the anchor boxes of the Faster R-CNN.

Each box has four parameters: the coordinates of the center, the width, and the height. At the same time, it produces a vector of probabilities corresponding to the confidence over each class of object.

The Non-Maximum Suppression method is also used at the end of the SSD model to keep the most relevant bounding boxes. The Hard Negative Mining (HNM) is then used because a lot of contrary boxes are still predicted. It consists of selecting only a subpart of these boxes during the training. The boxes are ordered by confidence, and the top is

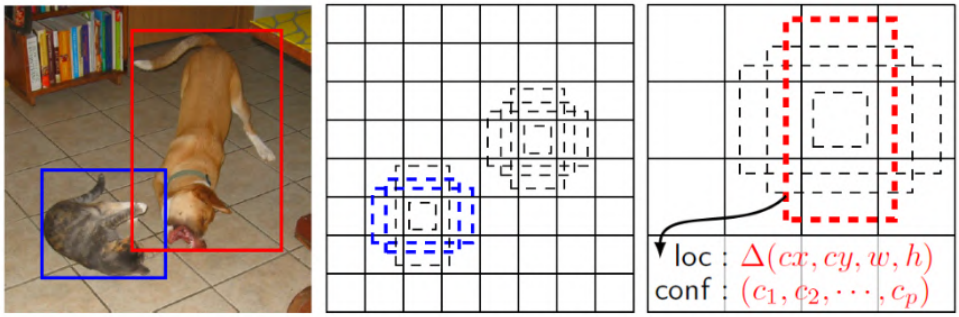


Figure 24: Multiple Bounding Boxes for Localization (loc) and Confidence (conf)
Source: Liu et al. (2016)

selected depending on the ratio between the negative and the positive, which is at most 1/3.

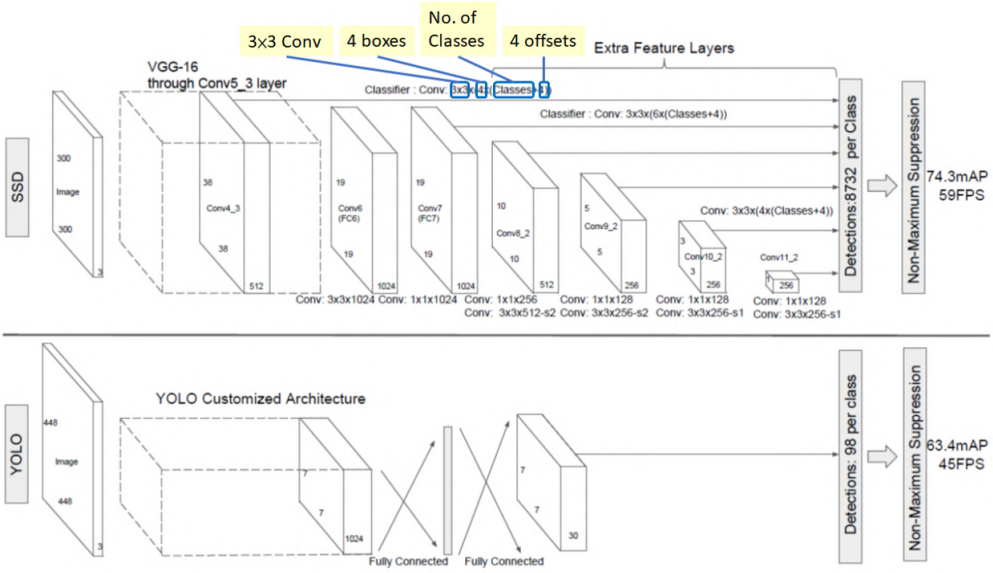


Figure 25: SSD (Top) vs YOLO (Bottom)
Source: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

The best SSDs models are trained with the 2007, 2012 PASCAL VOC datasets and the 2015 COCO dataset with data augmentation. They have obtained mAP scores of 83.2% over the 2007 PASCAL VOC test dataset and 82.2% over the 2012 PASCAL VOC test dataset. Over the test-dev dataset of the 2015 COCO challenge, they have had a score of 48.5% for an IoU = 0.5, 30.3% for an IoU = 0.75, and 31.5% for the official mAP metric. Results are presented at 26.



Figure 26: SSD results on MS COCO
Source: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

2.2.4 *Neural Architecture Search Net*

The Neural Architecture Search, presented in 2016, consists of learning the architecture of a model to optimize the number of layers while improving the accuracy over a given dataset Zoph and Le (2016). NASNet has reached higher performances with a lighter model than previous works over the 2012 ImageNet classification challenge.

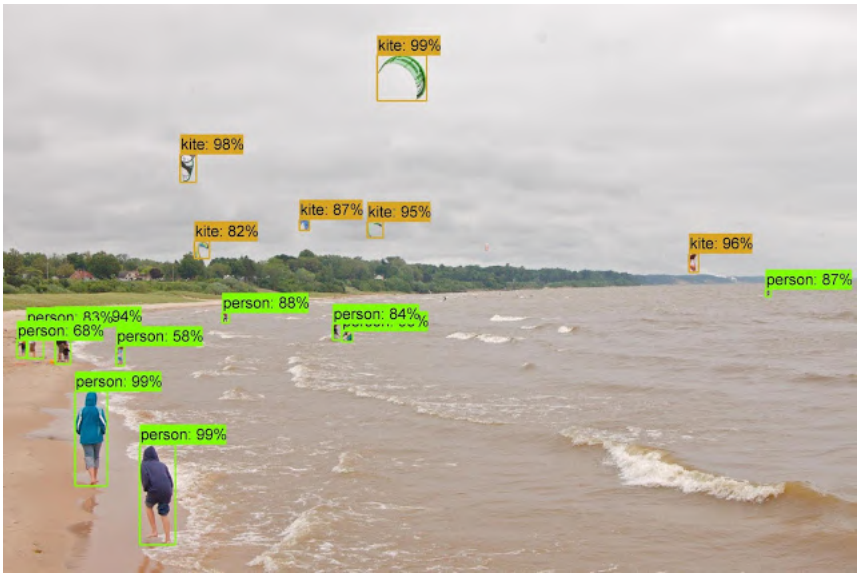


Figure 27: Example object detection using Faster-RCNN with NASNet
Source:https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html

The authors have released a new concept called Neural Architecture Search (NAS). NAS is used as a cell in a Recurrent Neural Network to learn its architecture using reinforcement learning.

The authors have applied this method to spatial object detection. The NASNet network has an architecture learned from the CIFAR-10 dataset and is trained with the 2012 ImageNet dataset. This model is used for feature maps generation and is stacked into the Faster R-CNN pipeline. Then the entire pipeline is retrained with the COCO dataset.

The best NASNet models for object recognition have obtained a 43.1% mAP score over the test-dev dataset of the COCO challenge with an IoU = 0.5.

2.2.5 *Mask Region-based Convolutional Network*

Mask Region-based Convolutional Network (Mask R-CNN) is an extension of the Faster R-CNN model presented in 2017 He et al. (2017) and detects objects in an image while simultaneously generates a segmentation mask for each instance.

Mask R-CNN adds a parallel branch to the bounding box detection in order to predict an object mask. The mask of an object is its segmentation by pixel in an image. This model outperforms the state-of-the-art in the four COCO challenges: the instance segmentation, the bounding box detection, object detection, and the key point detection.

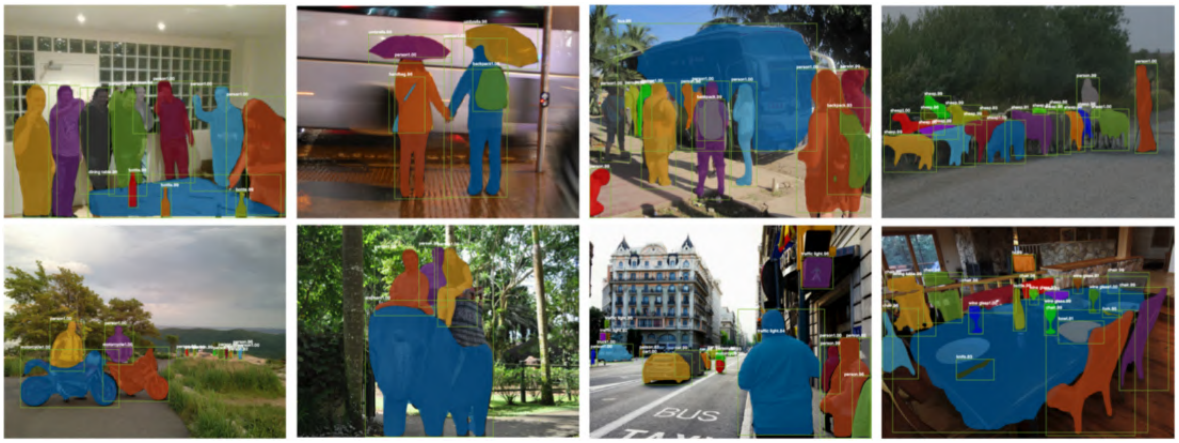


Figure 28: Mask R-CNN results on the COCO test set
Source: He et al. (2017)

For each candidate object, Mask R-CNN uses the Faster R-CNN pipeline with three output branches:

- a class label;
- a bounding box offset;
- the object mask

It uses Region Proposal Network (RPN) to generate bounding box proposals and produces the three outputs at the same time for each Region of Interest (RoI).

The model takes an image as input and feeds a ResNet network with 101 layers. This model looks like a ResNet, but each residual block is cut into lighter transformations, which are aggregated to add sparsity in the block. The model detects RoI's, which are processed using a RoIAlign layer. One branch of the network is linked to a fully-connected layer to compute the coordinates of the bounding boxes and the probabilities associated with the objects. The other branch is linked to two convolutional layers, and the last one computes the mask of the detected object.

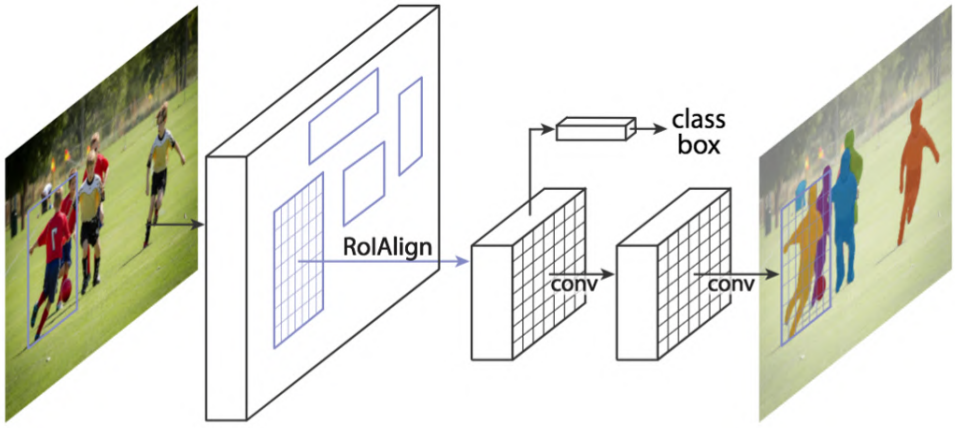


Figure 29: Mask R-CNN architecture
Source: He et al. (2017)

Three loss functions associated with each task to solve are summed. This sum is minimized and produces great performances because solving the segmentation task improves the localization and thus the classification.

The Mask R-CNN has reached mAP scores of 62.3% for an IoU = 0.5, 43.4% for an IoU = 0.7, and 39.8% for the official metric over the 2016 COCO test-dev dataset.

2.2.6 *Summary*

The detectors are all Convolutional Neural networks based and differ on how to apply the region delimiter of the image to perform classification, The faster RCNN first try to find the regions and after performing the classification of the given regions of the image, this can be called a two-phase technique (find the region and classify). The SSD and YOLO techniques try to be a one-phase technique to perform detection being more fast, being the YOLO model the only that can be used for real-time detection (with a typical/high setup).

These detectors have an involved CNN having such as 124 layers. The NASNet solution tries to find the best model using Reinforcement learning and a Recurrent Neural Network.

The output is a rectangle detection box, however the Mask RCNN, a technique based on the Faster RCNN finds the mask of the image.

The training dataset also has a vital role in training the model detector and in comparing the performance of the model. The best datasets for this type of problem are the ImageNet, COCO and Pascal VOC.

2.3 OBJECT TRACKING

This section has presented a state of the art methods that enables object tracking on multi targets where the goal is to automatically track pedestrians and cars and assign an ID for each of them on video.

The challenges of tracking are:

- Noisy observation, the three main reasons are:
  - occlusion, a target is partially or totally covered by another object;
  - disocclusion, a target that may be new or have seen in the past;
  - articulation, the nature of the object changes, an example of heavy articulation is present in sports like gymnastics and athletics;
- Confusion background, having a lot of targets moving very similarly or targets moving very close to each other;
- Abrupt motion, targets that not follow a motion model, sports are an example.

There are assumptions considered for tracking pedestrians and cars:

- objects do not disappear and reappear in different places in the scene, like teleport;
- the camera is not moving instantly to a new viewpoint, there is a gradual change between the camera and the scene.

2.3.1 *Kalman Filter*

Kalman filter is a method for tracking linear dynamical models with Gaussian noise. Every predicted and corrected state is a Gaussian. The only variables to maintain are the mean and covariance of the model.

Given a model of expected motion, linear dynamical models predict where the object will be in the next frame, even before seeing the image.

For each target will be given a model, a Gaussian model. The model starts with measurement and an initial belief. This model is now able to predict the location, and when

we get a new detection/measurement of the object, the model is updated, enabling a better prediction.

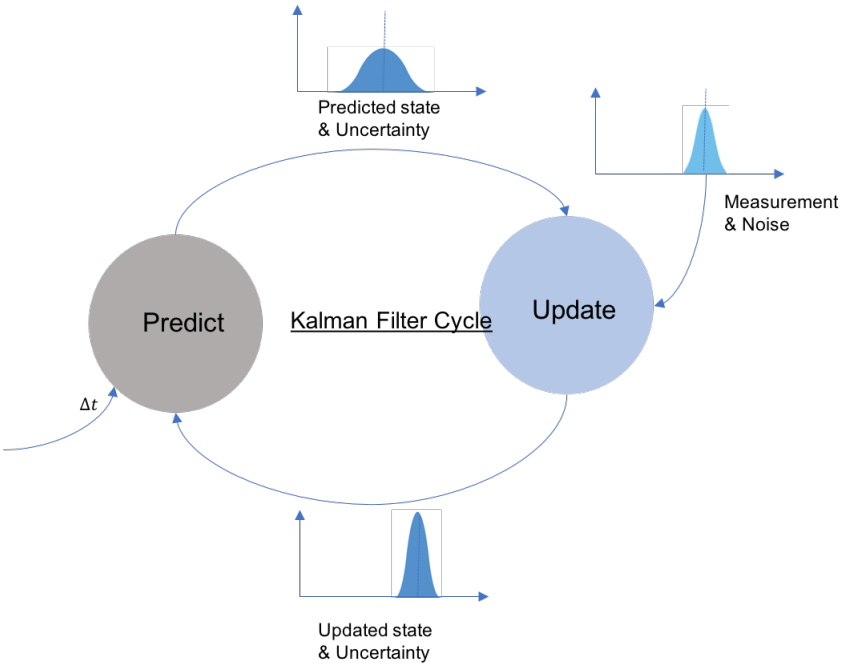


Figure 30: Kalman filter steps
Source:
https://www.ness.com/sensor-fusion-algorithms-for-autonomous-driving-part-1%E2%80%8A-%E2%80%8Athe-kalman-filter-and-extended-kalman-filter/

At the first detection/measurement we start with an initial belief for where the object will be in the next frames, on this initial belief a high noise will be set on the model, the measurement will be represented as the mean, and the noise will be represented as a high covariance on the Gaussian model. With the model set, we can predict, initially severely, where the object will be. The prediction will be worst over time, and this will result that the covariance of the model will be higher (resulting in a flattened Gaussian). On the next measurement that will also be a Gaussian with some noise, we will update the set model with this new measurement, and the result will be a Gaussian a lower covariance, lower even than the measurement Gaussian as presented at 35.

2.3.2 *Particle Filter*

The Kalman filter assumes a linear motion model and Gaussian noise and returns only one hypothesis (e.g., the probable position of a tracked object). If the movements are rapid and unpredictable (e.g., leaf on a tree during the windy day), the Kalman filter is likely to fail.

The particle filter is a generic algorithm for filtering problems, in computer vision is used for localization and tracking, and it was first presented by Del Moral (1996).

Particle filter returns multiple hypotheses and uses a set of particles (each particle presents one hypothesis) to represent the posterior distribution of some stochastic process given noisy and partial observations.

The following two figures are shown an example where particle filter is used for localization. The particles, shown in red, are the hypothesis for the localization, being the real localization shown in green. The blue lines represent the sonar that is the partial observations from the environment. In this example, particles were initially distributed by the environment.

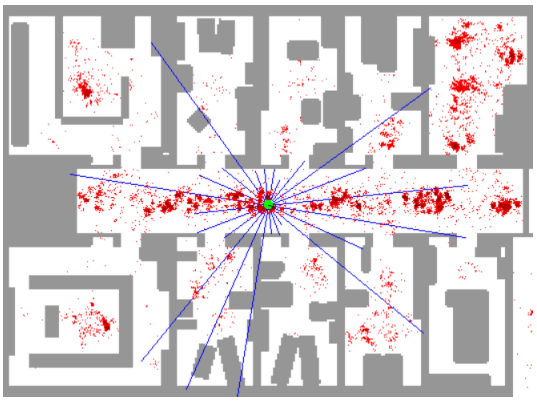


Figure 31: Example of a particle filter for localization
Source:
https://rse-lab.cs.washington.edu/projects/mcl/animations/global-floor.gif

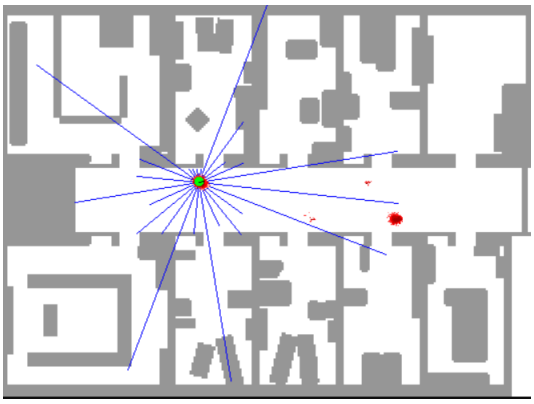


Figure 32: Next iteration of the previous example
Source:
https://rse-lab.cs.washington.edu/projects/mcl/animations/global-floor.gif

The particle filter initially may fall a bit, but then it catches back on because particle filters entertain multiple hypotheses, so even if some weird hypothesis is looking good as long as the real one eventually becomes better, it will come to it.

Particle filters implement the prediction-updating updates in a fair manner. Each particle incorporates tests on whether how it is likely that the object is at the position where the particle is. After the particles have been evaluated, the weights are assigned according to how good the particles are. Then the useful particles are multiplied, and the harmful particles are removed through the re-sampling process. The next particle generation then predicts where the object might be. This generation is evaluated, and the cycle repeats.

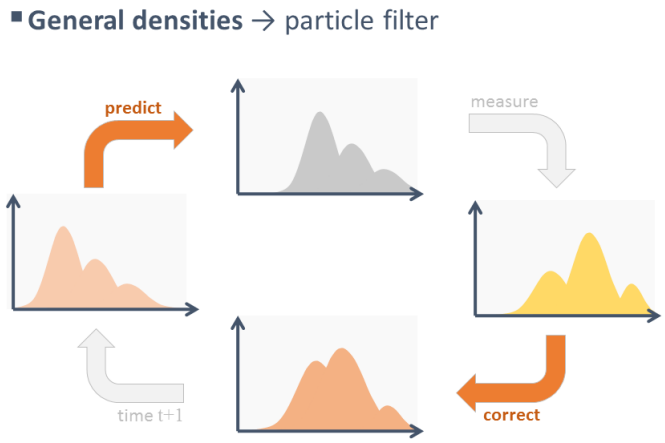


Figure 33: Predicted, measured and corrected states of particle filter
Source: https://www.codeproject.com/Articles/865934/%2FArticles%2F865934%2FObject-Tracking-Particle-Filter-with-Ease

The performance of the filter depends on the number of particles where the higher number of particles will lead to a better estimate, but it is costly when compared with KF.

2.3.3 *Mean Shift*

With origins in 1975, the mean-shift method idea is to find the *modes* (peaked areas) of a distribution or a probability density. The assumption is that there is a set of instances drawn from a probability density function, and the objective is to find the *mode* Fukunaga and Hostetler (1975).

Instead of just tracking the mean of the distribution, the mean-shift tracking is now applied to a distribution of a region based on the color histogram of the object. There are three stages for the mean-shift tracking:

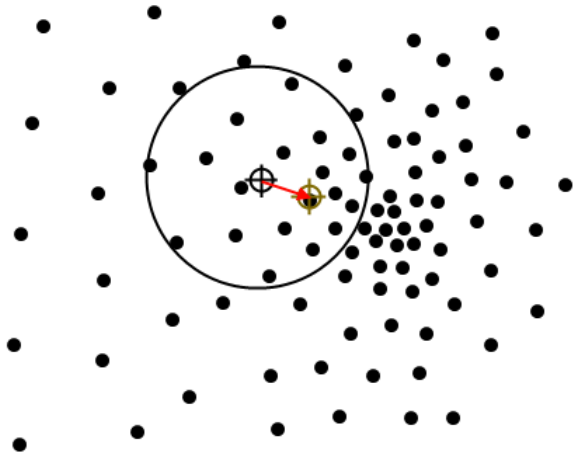


Figure 34: Mean shifting to the *mode*
Source: http://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconcepts/object_tracking_techniques/

1. Target model, the target object in the given frame that represents the target region in the given feature space (color histogram) ;
2. Search neighborhood, search in the neighborhood of the next frame that maximizes a similarity function of the target model;
3. Update the target model, and the new maximum becomes the start from the next frame.

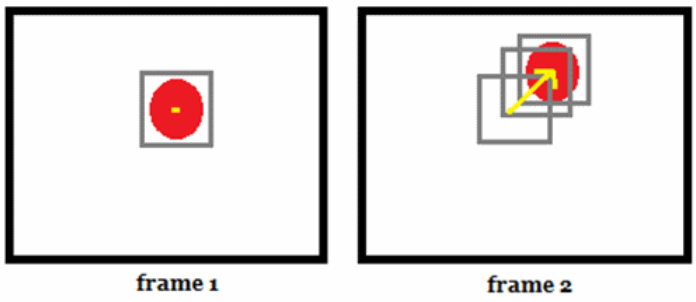


Figure 35: Mean-shift tracking
Source: http://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconcepts/object_tracking_techniques/

2.3.4 *Particle Swarm Optimization*

Particle swarm optimization (PSO), introduced by Eberhart and Kennedy (1995), is a population-based algorithm inspired by the foraging behavior of swarms. It is a method that optimizes

a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO is similar to particle filter in the way that exists multi-hypothesis candidates. However, in PSO, the particles have different behavior, and the other particles are considered to find the target.

PSO works by having a population of particles (a swarm), particles move in the search space of an optimization problem, that can be to find the minimum value of a function or, on this problem, find an object previous detected. The position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for better positions in the search space by changing its velocity according to rules originally inspired by behavioral models of bird flocking. The movements of the particles are guided by their own best-known position in the search-space as well as the entire swarm's best-known position. When improved positions are being discovered, these will then come to guide the movements of the swarm. The process is repeated, and by doing so, it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

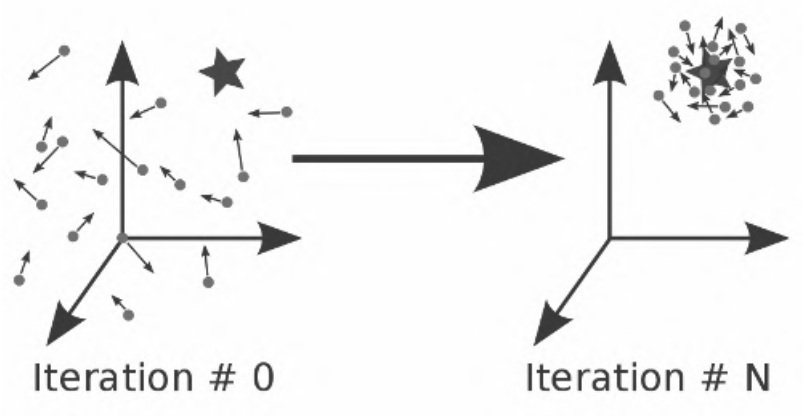


Figure 36: PSO
Source: https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html

To enable PSO for object tracking, features from the image needed to be extracted, these features will be the content that particles will try to find. The features to be extracted, on every iteration, are the unique information from the image, being the quality of the features decisive for a multi-object distinction, an example of features can be HOG features Hussain et al. (2013).

2.3.5 *Recurrent Neural Networks*

Object tracking is a sequence modeling problem, where the sequence is composed of the image frames. A deep learning solution for sequence modeling problems is Recurrent Neural Networks (RNN's).

RNN's have in them a sense of some memory about what happened earlier in the sequence of data. This helps the system to gain context. They have the capability to look back on all previous inputs.

RNN's can be used on a variety of problems:

- Text generation;
- Text Translation;
- Speech recognizing;
- Object tracking;

RNN's are neural networks with loops to persist information. RNN is called a recurrent as they perform the same task for every element in the sequence, and output elements are dependent on previous elements or states. This is how RNN's persist information to use context to draw inferences.

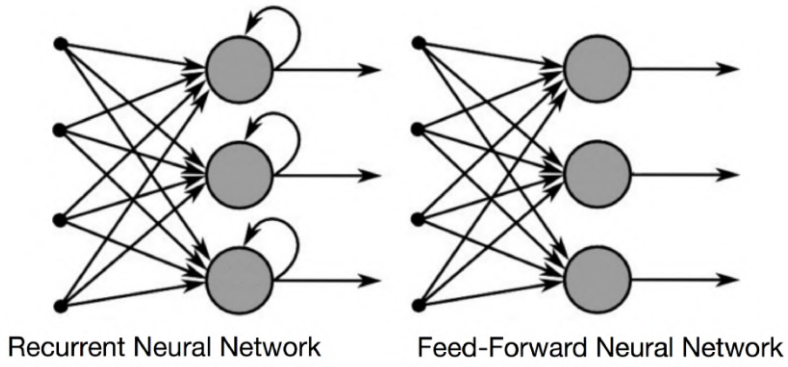


Figure 37: RNN are neural networks with loops to persist information
Source: https://medium.com/datadriveninvestor/recurrent-neural-network-rnn-52dd4f01b7e8

Unlike standard feedforward neural networks, recurrent networks retain a state that can represent information from an arbitrarily long context window. Although RNN's have traditionally been challenging to train and often contain millions of parameters, recent advances in network architectures, optimization techniques, and parallel computation have enabled successful large-scale learning with them. In recent years, systems based on long short-term memory (LSTM) and bidirectional (BRNN) have demonstrated ground-breaking performance on varied tasks. Lipton et al. (2015)

RNN's can directly learn robustness to complex visual phenomena such as occlusion and appearance change. Also, they are not computational expensive being able to perform tracking at 155 FPS Gordon et al. (2018).

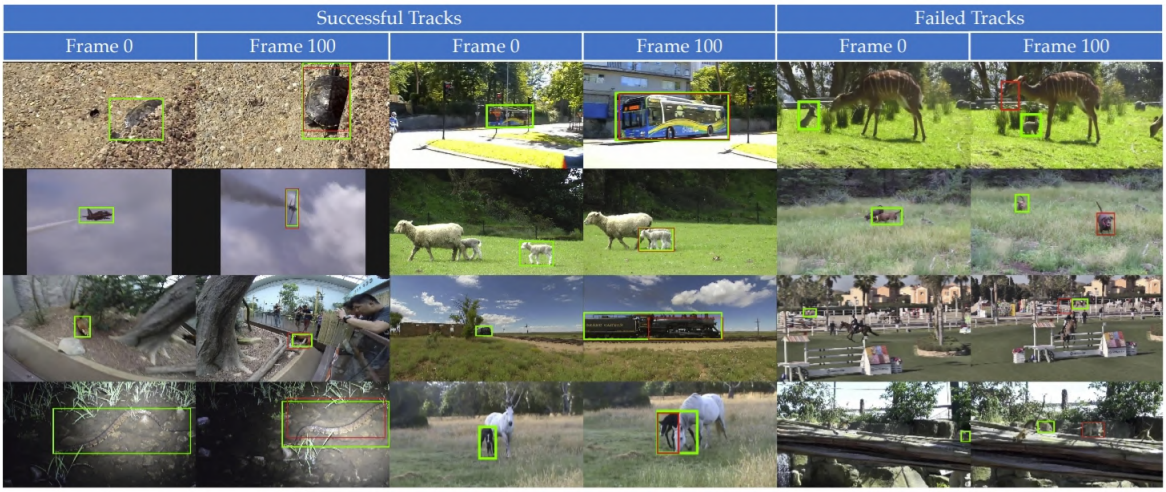


Figure 38: Results of an LSTM on object tracking
Source: Gordon et al. (2018)

2.3.6 *Summary*

From the tracking methods presented, the Kalman Filter (KF), the Particle Filter (PF), and the Particle Swarm Optimization (PSO) have a similar procedure based on prediction, measurement, and correctness. The Kalman Filter is a simple solution and is dependent only on the previous state. The PF and PSO are between them similar, and the main difference is that the particles on the PSO consider the results of other particles, not being independent. The Mean Shift (MS) is an excellent method, especially assuming that huge occlusions and variations of the same object may not occur. A Recurrent Neural Network performing tracking presents good results, but some flexibility on the algorithm for other use cases can be lost, and to set up, this large algorithm amounts of data can be needed.

For this work, the Kalman Filter, in combination with a fingerprint method, is a reliable choice. The Men Shift could also be a great choice; the use of this method as a secondary tracking method enables the tracking system to be very robust.

2.4 OBJECT FINGERPRINT

This section has presented the state of the art methods for object fingerprint. Object fingerprint is the capacity to assign to a detected object description or set of values that enable a posterior detection of a previously detected object, although there may be slight differences, the objective is to know if it is the same object. Assign a fingerprint to each detected object have the potential to robust tracking, which is enabled to be cross cameras given that the newly detected object (by a camera on a different location) can be compared with objects passed on other cameras.

2.4.1 *Colour Co-occurrence Histograms*

A straightforward method for object fingerprint is the creation of a color histogram from the colors presented on the detected object.

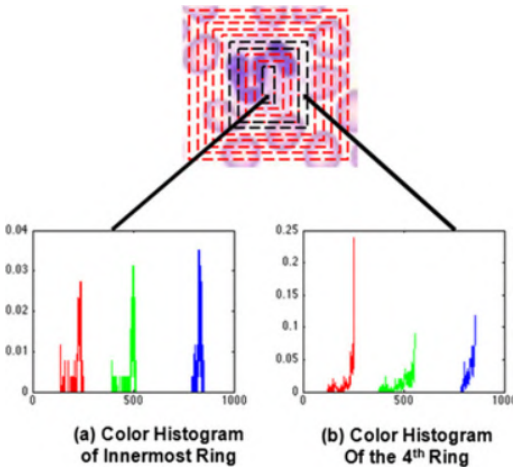


Figure 39: Color histogram on biological cells
Source: https://www.researchgate.net/figure/An-illustration-of-HAH-calculation-a-Color-histogram-of-the-central-ring-b-Color$_fig7_{264938374}$

Colour Co-occurrence Histograms (CCoH's) are a way of combining the color and spatial layout of an image into a single holistic descriptor. Applications for CCoHs are usually found in object classification Palm (2004).

CCoHs combine color and spatial information by inspecting an input image pixel-wise and within a certain radius count what colors co-occur along with the central pixel. This raises the matter of how to represent color. The simple answer is to use RGB information. However, in the RGB color, space intensity and color are intertwined. The commonly used HSV color space. However, it has no such interdependencies among its channels, and incidentally, it more closely represents how humans perceive color.

A way to overcome the waste in color representation is to use clustering on colors actually present in a training set, instead of quantizing the entire color space by dividing it into cubes. From a video track, the colors present in vehicle detection windows can be gathered, and then be clustered into the desired number of color templates, each with a unique label. By assigning the unique template labels to colors as found in input images, a CCH can be constructed in the usual manner the (2009).

Clustering here is performed using k-means clustering, which continues until there is no change in cluster center membership for data points. The following figure shows an example of a CCH and the corresponding vehicle.

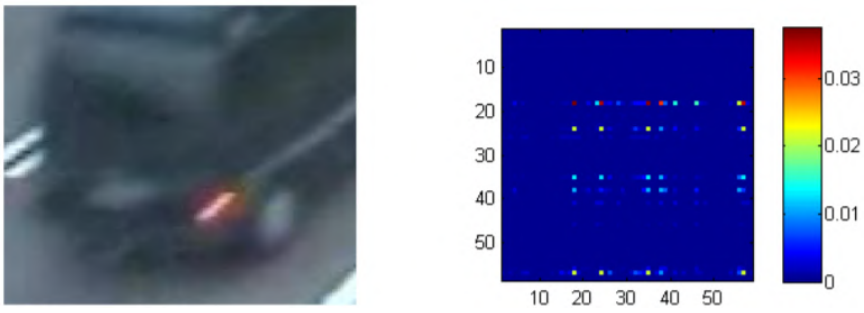


Figure 40: Colour Co-occurrence histogram, for a vehicle image, with radius 20 pixels and 59 colour clusters
Source: https://www.researchgate.net/figure/An-illustration-of-HAH-calculation-a-Color-histogram-of-the-central-ring-b-Color$_fig7_{264938374}$

2.4.2 *Color Invariant SIFT*

An elegant approach to extend SIFT with color information is Color Invariant SIFT. Instead of creating an alternative descriptor, it is also possible to perform SIFT keypoint detection on alternative color spaces, besides just the intensity channel, where on each channel of the HSV color space, SIFT is applied. Another more advanced technique is through the usage of invariant color models Abdel-Hakim and Farag (2006), each with specific invariance to shadows, illumination, highlights, and noise the (2009).

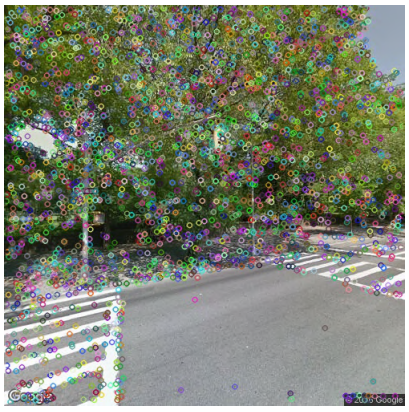


Figure 41: Example of SIFT
Source:https://stackoverflow.com/questions/40694713/is-sift-algorithm-invariant-in-color

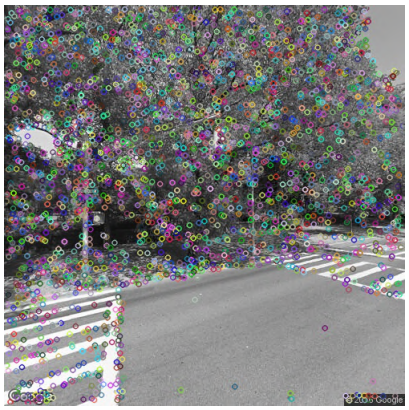


Figure 42: Previous image with Color Invariant SIFT
Source:
https://stackoverflow.com/questions/40694713/is-sift-algorithm-invariant-in-color

2.4.3 *PCA-SIFT*

Another variation on SIFT comes in the shape of PCA-SIFT Ke and Sukthankar (2004b), again the same method for detecting keypoints is employed as with regular SIFT. However, the descriptor which is computed at the given location, orientation, and scale are different.

Principal Components Analysis (PCA) is a commonly used technique to reduce the dimensionality of data, with minimal loss of information. By computing the covariance matrix of some acquired data, normalized by the data mean, and then performing Eigenvalue decomposition on the resulting matrix, and Eigenspace is obtained. Then onto this Eigenspace, a projection can be made with a new data instance, which reduces the dimensionality of the given data to the number of components specified in the construction of the Eigenspace.

Inherent to this approach lays the assumption that the data is a Gaussian distribution in which mean and the standard deviation is meaningful, and moreover, large covariances actually yield essential information. Also, PCA is restricted to orthogonal linear combinations. Despite this, PCA is found to be usefully combined with SIFT for object fingerprinting the (2009).

2.4.4 *Fourier Transform*

The Fourier Transform can be used to analyze the frequency of an image. It is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image Van Heel (1987).



(A1) SIFT: 4/10 correct | (A2) PCA-SIFT (n=20): 9/10 correct | (B1) SIFT: 6/10 correct | (B2) PCA-SIFT (n=20): 10/10 correct

Figure 43: A comparison between SIFT and PCA-SIFT
Source: Ke and Sukthankar (2004a)

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction, and image compression. Bracewell and Bracewell (1986), the application of the Fourier Transform on letters of the alphabet can be seen on 44.

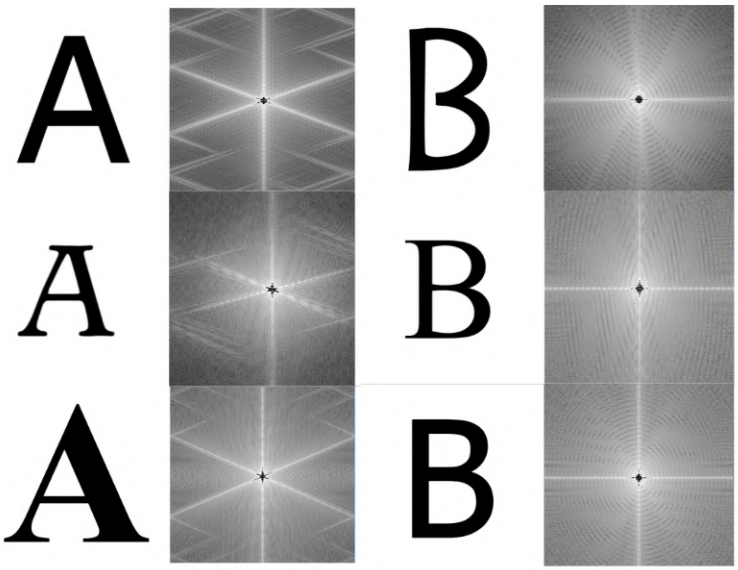


Figure 44: Fourier Transform applied to letters
Source: https://www.youtube.com/watch?v=cLHAUhGkzwQ

The results of 44 are produced given the following methods. The Fourier Transform is used if we want to access the geometric characteristics of a spatial domain image. Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy

to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain.

In most implementations, the Fourier image is shifted in such a way that the DC-value (i.e., the image mean) F(0,0) is displayed in the center of the image. The further away from the center an image point is, the higher is its corresponding frequency.

2.4.5 *Autoencoders*

Autoencoders is a deep learning method useful for image compression and reconstruction.

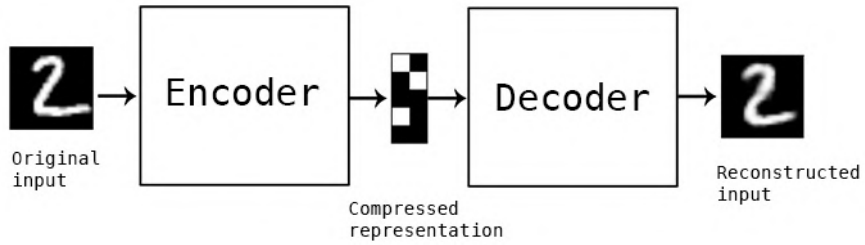


Figure 45: Autoencoders
Source: https://blog.keras.io/building-autoencoders-in-keras.html

An autoencoder is an artificial neural network with architecture with two critical changes.

The first one is that the number of inputs on the input layer is the same as the outputs on the output layer, which is expected to the output image not only be the same size but also to be the same image given at input.

The second part is that in one layer, there is a bottleneck, the numbers of neurons on that layer are much less than on the input and output layer forcing the network to compress the information on that layer for recompression Vincent et al. (2008).

The network should output the same input after applying a series of operations to it. The first part of the network that compresses the input into fewer bits is called encoder, and the second part reconstructs the image of the decoder.

Being the output a replica of the input (with some information losses) the gain is in the middle layer where the image is encoded, this layer must contain enough information to represent the output, given that the hidden layer is smaller than the input and output layers, it represents the same information in a lower density. Being the image fingerprint, the hidden layer information.

2.4.6 *Summary*

The fingerprint methods are very different from each other.

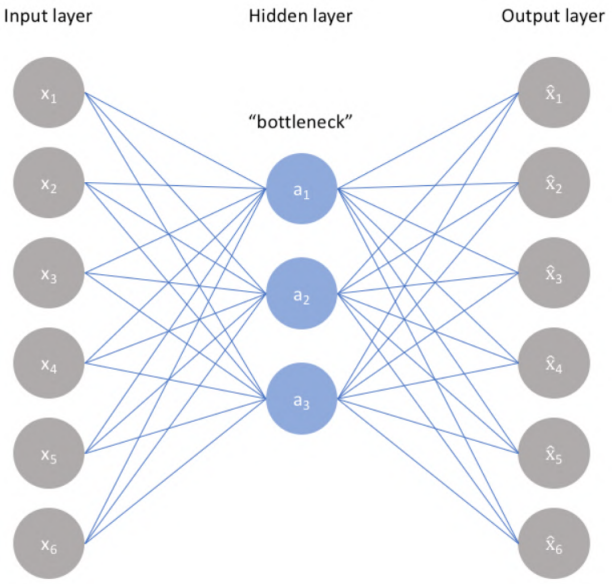


Figure 46: Autoencoders architceture
Source:https://www.jeremyjordan.me/autoencoders/

The SIFT based methods are a solution that is not great if there is a lot of object articulation, although it must be capable of guarantee a minimum of similarity value for the same object along time.

The Fourier Transform may not be useful to be a suitable similarity measure for objects that presents much noise, having the risk of not detecting that two objects with high noise are different.

The color co-occurrence histogram is a simple method that can distinguish different objects, but its capacity to distinguish similar objects is low.

The autoencoder technique to extract a similarity with the encoded nodes may be the right solution if the network learns huge high levels features, but if the network learns to encode and to perform the inverse operation with a small loss, then it will fail. Due to the impossibility of choosing the best method, all the methods will be tested in this work.

2.5 RELATED WORK

This section is presented work that shares components of the people and car counting problem. The related problems, although they different, helps to identify how the methods and components are integrated.

2.5.1 *Surveillance*

Surveillance is a problem very similar to the presented on this work. Of all surveillance solutions, the focus in this section will be on the Chinese surveillance system. China's mass surveillance system is a complicated network of cameras used by the Chinese government to supervise the actions of Chinese citizens Times.

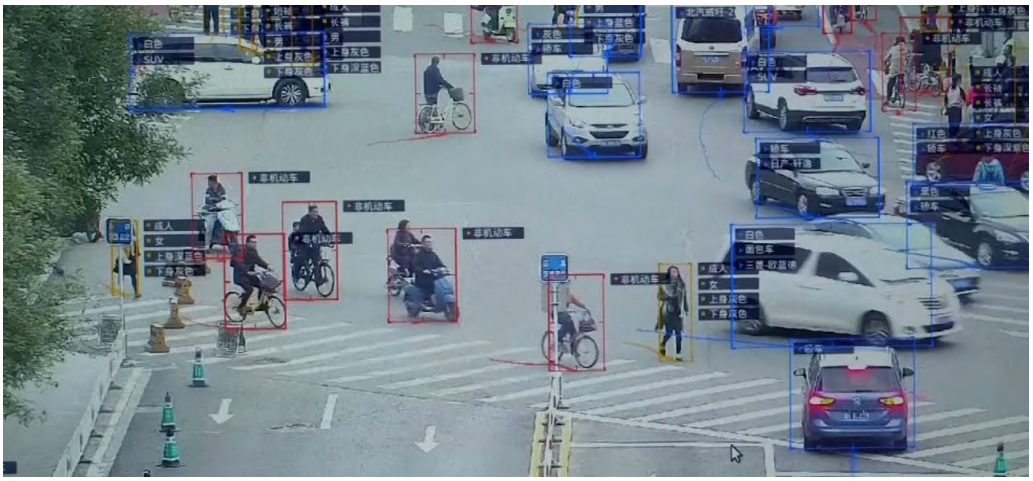


Figure 47: Chinese surveillance system
Source:
https://www.teslarati.com/china-surveillance-ai-startup-4-5b-valuation/

China surveillance system performs object detection, classification, tracking, and fingerprint. The system detects pedestrians and vehicles. For each object detected is performed classification for a description of the object, like gender. Tracking is performed; being able to see was the object passed. The system also uses a face recognition algorithm for pedestrian identification Fortune. The China system may seem a complete solution to the problem of this work. However, it is not considered perfect with occlusions and crowd detection. The methods used were not revealed. However, the most likely technique for classification is convolutional neural networks.

2.5.2 *Sports*

Another related work and with a very similar problem are the sports video analysis systems. A sports video analysis system is a multiple player tracking problems. In an uncalibrated camera setting, robust multi-object tracking can be challenging due to a number of reasons: noise, occlusion, fast camera motion, low-resolution image capture, varying viewpoints, and illumination changes. .

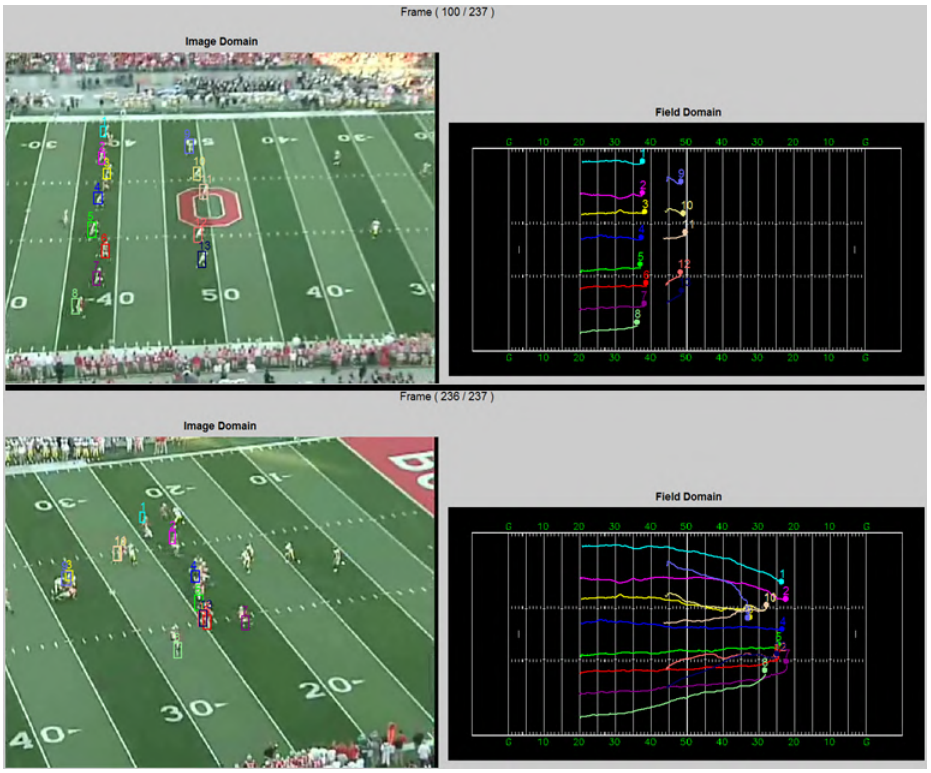


Figure 48: Sports video analysis system
Source: http://autoscout.adsc.illinois.edu/publications/robust-video-registration-applied-to-field-sports-video-analysis-robust-multi-object-tr

To address the problem of multi-object tracking in sports videos, a solution is to go beyond the video frame domain and make use of information in a homography transform domain that is denoted the homography field domain — using a particle filter based tracking algorithm that uses both object appearance information (e.g. color and shape) in the image domain and cross-domain contextual information in the field domain to improve object tracking Ghanem et al. (2012).

Another solution founded for tracking was using the mean shift algorithm with a combination of a color histogram algorithm Yan (2018).

2.5.3 *Games*

Games are a different problem; however the object detection and tracking methods in people introduce a new concept, the tracking by appearance.

People are tracked, particularly by their limbs, by learning their appearance. The idea is to build a person model, and the person model is made up of different things: appearance, structure, and dynamics. The structures and dynamics are generic. Structures are the

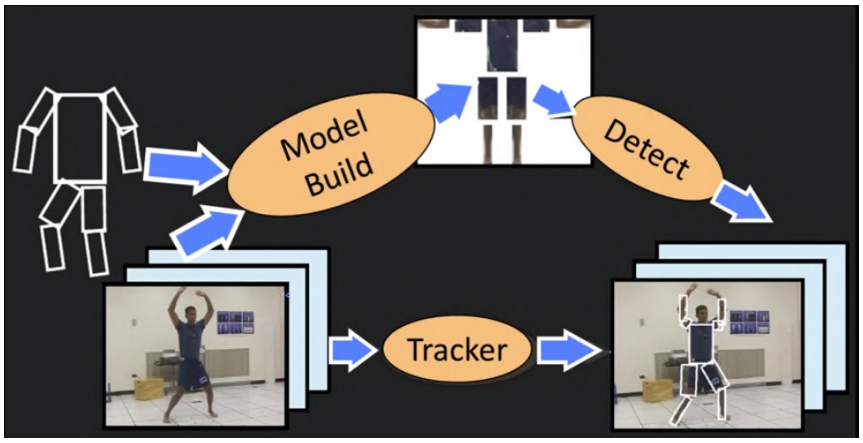


Figure 49: Video analysis on a video game
Source: Ramanan et al. (2007)

lengths of body parts and dynamics of how they can move, and the appearance is person-specific Ramanan et al. (2007).

3

THE PROBLEM AND ITS CHALLENGES

This chapter has presented the problem, challenges, and the proposed solution of this work.

3.1 THE PROBLEM

To build an algorithm for counting vehicles and pedestrians based on video subproblems were identified.

The first problem is how the algorithm knows which type of object (vehicle or pedestrian) is presented in a frame of the video, how to classify an image? This is the object classification problem. The human object has high articulation it can be defined in various forms and always be an easy detection for a human, to achieve the correct classification of the pedestrian and vehicle objects a machine or deep learning is required, which requires the creation of a model trained for this types of objects with propers datasets.

The second problem is how to detect objects in an image of the video, how to detect multiple targets in an image. This problem is the object detection problem. Knowing that an object is present in an image is similar to the classification problem, to enable multi objects being detecting in an image computer vision, segmented techniques or multi-classification is needed.

The third problem is how it is possible to associate a new object detected with a previous detected given only the content of the image. This is called the object fingerprint problem. The following color point is a solution that can be developed in other systems. However, the tracking in this system has to be resistant to occlusions and to objects that have similar behavior with proximate objects.

The fourth problem is how to track objects and assign an ID for each of them on video, called object tracking problem. An image hash does not solve the problem; for this problem, a flexible solution that permits the articulation of the object is needed.

These four subproblems identified compose the counting objects problem.

The combination of methods for tracking and fingerprint raises the problem of which method of association is the best to unify the results of both components to improve the solution.

3.2 CHALLENGES

Detecting pedestrians and vehicles is the proposed task; however, when the density of objects (pedestrians and vehicles) is massive, the ability to perform detecting and tracking decreases. On high-density scenarios (thousands of objects) like on a football stadium, manifestations, start of marathons, concerts, among others, the detection component, the way is developed will likely fail, that is, will not detect a significant percentage of objects. For this type of problems, that is also very relevant to know how many objects are present, different methods are required. These methods are not presented because the crowd counting problem is assumed as a different counting problem, such as on 50.



Figure 50: People counting on a crowd scene
Source: cro

Another challenge is detecting objects at night or rain scenarios. In these scenarios, objects may not be detected by the detection algorithm. Objects in these scenarios change their color, and their visible form, training a detection algorithm for these scenarios can be complicated by lack of information and the different combinations that an object form can have at these conditions.

The challenges identified are mainly because of the limitations of the detection method. However, these limitations can be slightly extended with traditional computer vision methods, like background subtraction Piccardi (2004) and shadow removal techniques Tiwari et al. (2016).

3.3 PROPOSED APPROACH

The main objective is to define a pipeline of methods that result in a solution capable of counting objects. The proposed solution is a pipeline consisted of deep learning and computer vision techniques. The pipeline has four main components: object classification, detection, fingerprint, and tracking.

The object classification and detection components work together, and from the results, the detection boxes location with it has given a label.

The object tracking can be one method or a combination or one or more methods such as Kalman Filter with a fingerprint method or the Mean Shift in parallel with the Kalman Filter, the combination of methods is needed because one method is not powerful enough to occlusions or to follow the object, being this the two main problems of tracking.

The object fingerprint problem can be solved by different manners. However, it is not known, which are the best methods for this type of problem. Is proposed a study of fingerprint methods to infer the best one for this type of problem.

The input of the system is a frame of a video (or live recording) given periodically. The output of the solution is the number of people detected, also considering the previous frames given at the input.

For the integration of the pipeline on other systems to explore the results of the pipeline and develop high-level features, the pipeline solution should consist of a library that is used by another primary system, such as a web application.

4

DEVELOPMENT

On this chapter are presented the decisions made for the construction of the algorithm pipeline. A study on fingerprints methods was made being presented the problems on this subject, the work, results, and conclusions.

After defined the algorithm pipeline is presented features that are capable of extracting with the components chosen, presenting the results of the features implemented, and a summary of the results and integration are that can be made.

4.1 DECISIONS

The algorithm for detected and track pedestrians and cars consist of a pipeline of deep learning and computer vision methods. The pipeline consists of four main components: object detection, object classification, object tracking, and object fingerprint. The object detection and classification retrieves the location of an object and a given label, for example, car from the (90,35) upper left coordinate to the (320,125) bottom right coordinate of the detected object.

For object detection, the method chosen was the YOLO algorithm (version 3), this algorithm also performs classification and is the right choice given its fast performance and precise detection in different scenarios. The YOLO algorithm, have some problems, the problems are the high false positives rate, the multiple detections on the same object, this problem is solved with the non-max suppression technique Neubeck and Van Gool (2006) and, less common, the wrong box segmentation, given an enormous detection box size for the detected object.

For performing object tracking, it is considered the location of the objects (and it is history), and it is content, the object image itself. To consider this information, the methods chosen were the combination of the Kalman filter and image fingerprint. The Kalman filter consider the object location and it's future location and the fingerprint method considers the image content, given a equal weights on the two techniques and thresholds to detected if is not the same object an association with potential objects (comparing the objects in

the current frame with past objects), the correct association of objects is made with the Hungarian Algorithm.

The fingerprint methods, although very different it was not possible to know which was the best to apply to this problem. So a fingerprint study comparison was made between the color co-occurrence histogram, the Fourier Transform, SIFT, and Autoencoders. The objective of the study was to conclude which methods were the best for help

4.2 FINGERPRINT STUDY

For the development of the study were created a dataset for pedestrians and vehicles. Each dataset contains 20 instances of the same object and 100 objects. The study compares the similarity between instances of the same object, and it compares with instances from other objects. The techniques used were the color-occurrence histogram, the SIFT, the Fourier Transform, and the Autoencoders.

All the code is available on https://github.com/CarvalhaisMatos/image-fingerprint-study

Dataset

Two datasets were created, one for vehicles and another for pedestrians. Each dataset contains 20 instances of the same object and 100 objects. The datasets were created with detections performed with the YOLOv3 detector and Kalman Filter on each frame on different scenarios. The instances on the dataset have a temporal spacing from at least 50 frames and 1 second. The instances in the dataset have extracted the instances that were all human checked.

Metrics

The metrics compared in this study were the similarity between the same object and with different objects and the computational time of each similarity.

For the similarity between the same object, two graphs are presented, a graph that compares the similarity between the object and his previous occurrence and other a graph that presents the similarity of the first instance of the object with the following instances.

For the similarity between different objects is presented a confusion matrix that compares an object with another instance of himself and with the other objects.

The computational time is given based on an Intel i5-8250U CPU.

Color Co-occurrence Histogram

The Color Co-occurrence Histogram (CCoH) can be applied in different forms considering different types of regions in the image. The comparison between images is made with the

histograms of the regions; the similarity of the histograms is given with the Bhattacharyya distance Cha and Srihari (2002). The image is divided into cells, and each cell has calculated a histogram, which will be compared with the corresponding cell of the image to compare. The similarity value is the mean of the similarity between all the comparisons between cells.

The following results are presented considering two equal regions, tow rows, and one column, this was the best region combination found, and all cells combinations tests are presented in the following table (1).

Table 1: Color Co-occurrence Histogram results in different environments

Rows	Columns	Mean similarity (current with next)	Precision	Recall	F1 measure
1	1	0.819	0.067	0.96	0.124
1	2	0.770	0.055	0.96	0.104
1	3	0.742	0.476	0.97	0.090
2	1	0.823	0.146	0.95	0.253
2	2	0.775	0.108	0.96	0.194
2	3	0.747	0.092	0.96	0.168
3	1	0.820	0.194	0.95	0.323
3	2	0.756	0.167	0.96	0.167
3	3	0.693	0.062	0.96	0.062

The comparison with pedestrians and cars was very similar; the comparisons made with car instances showed better results, being presented the results for pedestrians instances.

The comparison of pedestrians between the current and previous instances of the same object along time on as a mean value of 0.823 and can be visualized on 51.

The comparison of pedestrians between the first instance and next ones of the same object along time had a similarity loss of 15.1%, as shown on 52.

A confusion matrix between objects was made, and an example containing ten instances is presented at 53. A threshold based on the mean with the standard deviation of the diagonal values was defined.

$$threshold = \mu - 2\sigma$$

With the defined threshold, it was reached an F1 measure of 0.253, a precision of 0.146, and a recall of 0.95, an excerpt of 10 objects is presented at 54.

The computational time results are the following:

This technique presents excellent results and solves the problem of similarity between the same object along time with a proper differentiation from other objects.

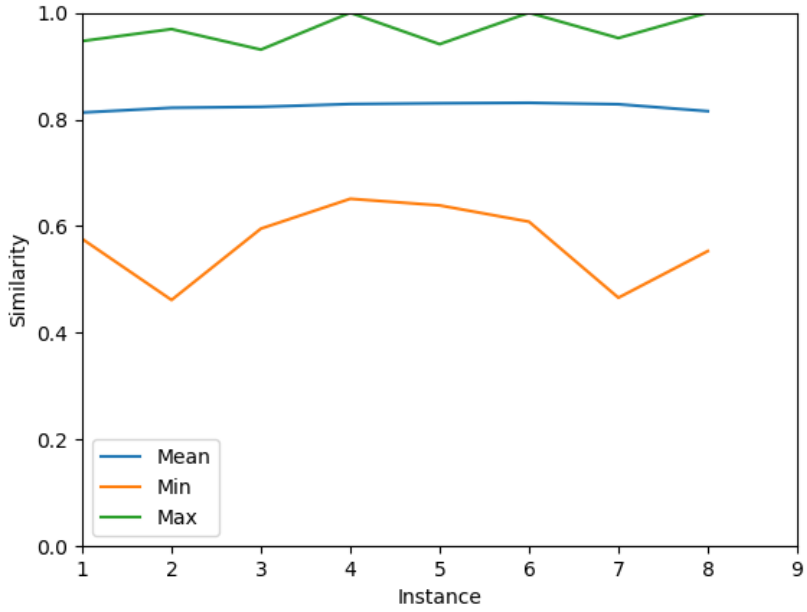


Figure 51: Similarity with the current and next instance of the same object

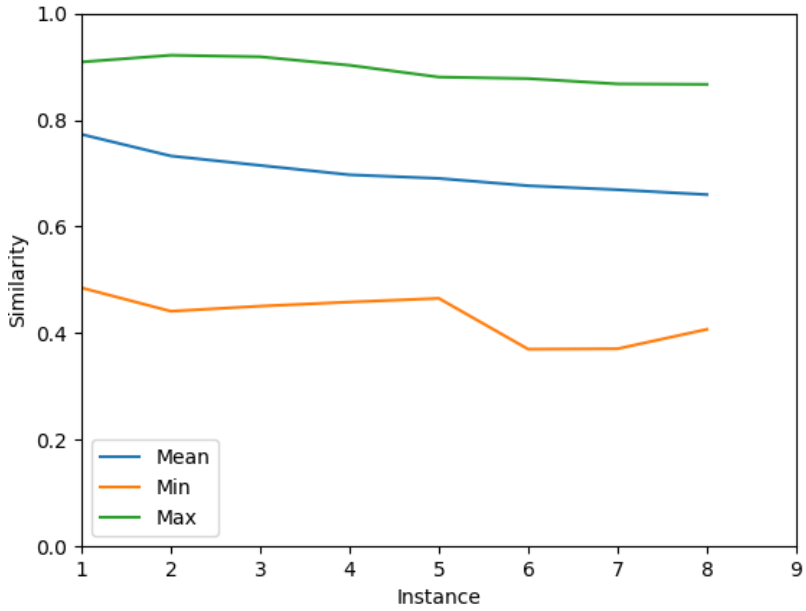


Figure 52: Similarity between the first instance and next ones of the same object along time

SIFT

The similarity between images using SIFT was done considering the keypoints found in the second image and in the first image. The total points of keypoints found in the second image were divided by the points found in the first image, being applied a hyperbolic

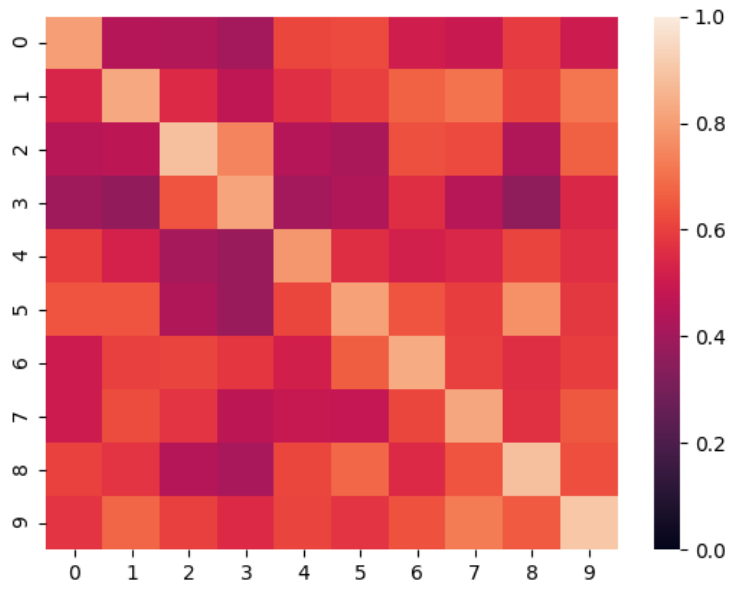


Figure 53: Confusion matrix co-occurrence color histogram, limited to ten objects

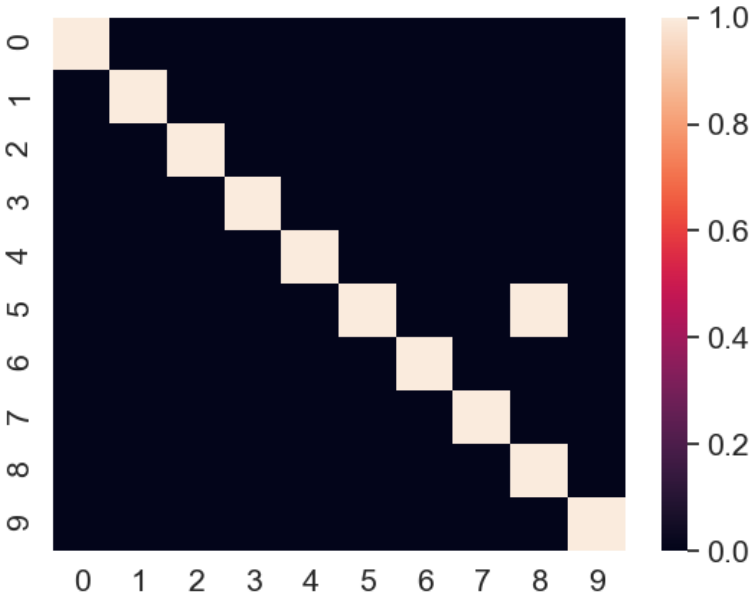


Figure 54: Confusion matrix co-occurrence color histogram with threshold, limited to 10 objects

tangent function to the percentage of points matched. The hyperbolic tangent function applied was:

$$f(x) = \tanh(7x)$$

Table 2: Time measurement of using CCoH for image similarity between two images

Mean	Standard Deviation	Min	Max
36 ms	33 ms	10 ms	232 ms

The comparison with pedestrians and cars was very similar; the comparisons made with car instances showed better results, being presented the results for pedestrians instances.

The comparison of pedestrians between the current and previous instances of the same object along time on as a mean value of 0.512 and can be visualized on 55.

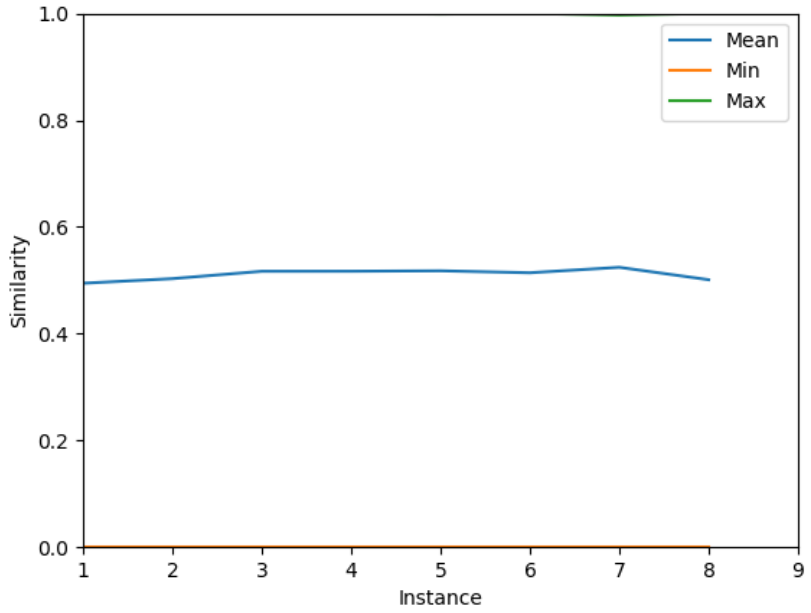


Figure 55: SIFT similarity with the current and next instance of the same object

The comparison of pedestrians between the first instance and next ones of the same object along time had a similarity loss of 32.5%, as shown on 56.

A confusion matrix between objects was made, and an example containing ten instances is presented at 57. A threshold based on the mean with the standard deviation of the diagonal values was defined.

$$threshold = \mu - \sigma$$

With the defined threshold, it was reached an F1 measure of 0.538, precision of 0.438, and a recall of 0.70, an excerpt of 10 objects are presented at 58.

The computational time results are the following:

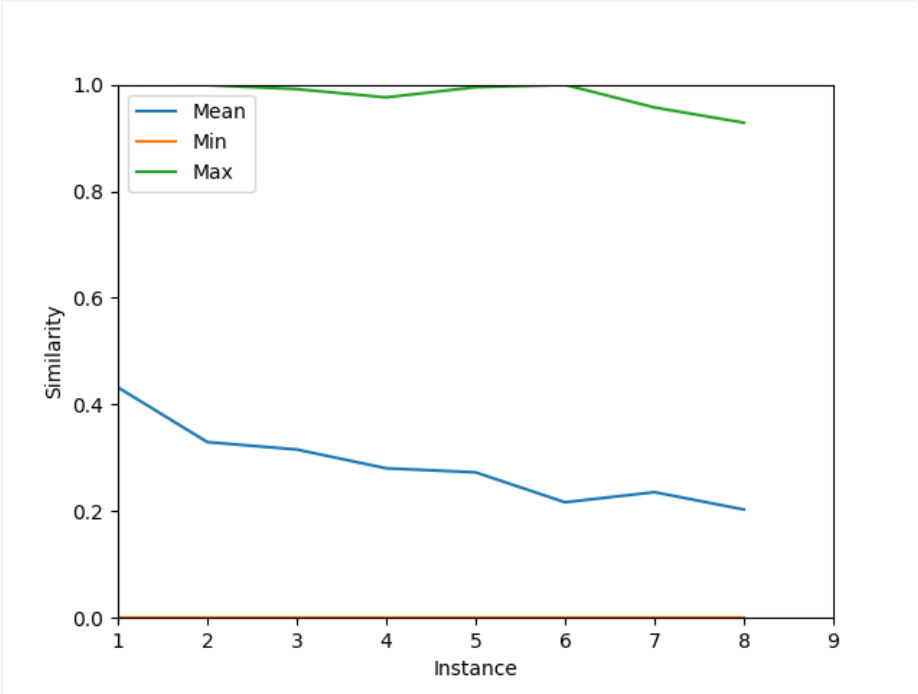


Figure 56: SIFT similarity with the current and next instance of the same object

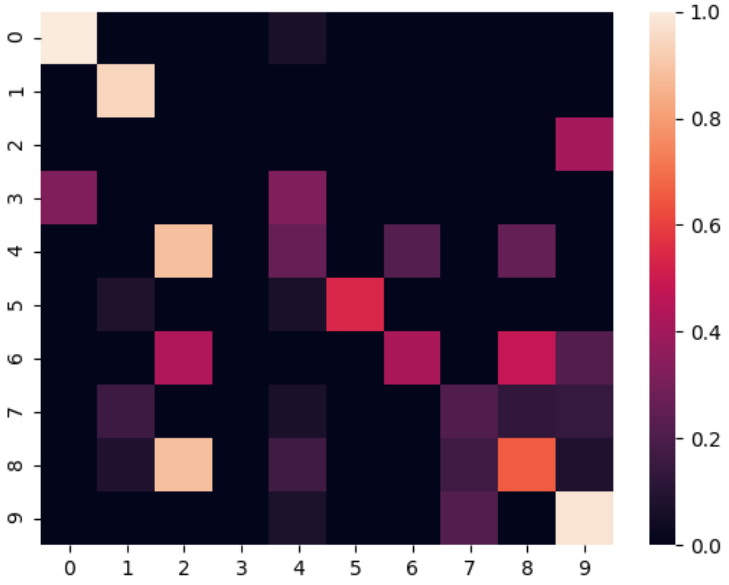


Figure 57: Confusion Matrix between objects using SIFT

This technique presents consistent results. The similarity between the same object varies between 0 and 1, making it very fickle with no guarantee that they detect with minimal similarity. This technique also could not distinct instances from other objects instances.

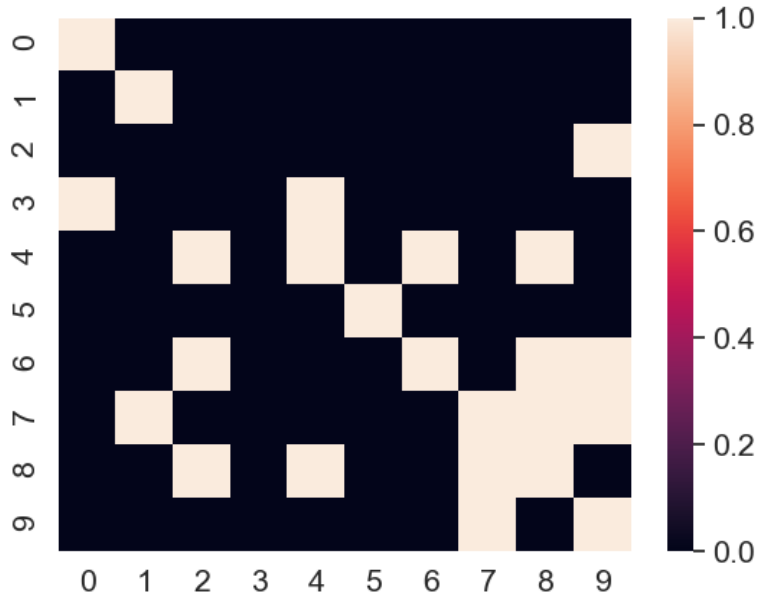


Figure 58: Confusion matrix co-occurrence color histogram with threshold, limited to 10 objects

Table 3: Time measurement of using SIFT for image similarity between two images

Mean	Standard Deviation	Min	Max
7 ms	5 ms	0.9 ms	43 ms

Fourier transform

The similarity between images using the Fourier Transform was made applying the Discrete Fourier Transform (DFT) to the image and calculate the magnitude spectrum. The similarity value is the mean comparison of each magnitude spectrum for each color channel.

The comparison with pedestrians and cars was very similar; the comparisons made with car instances showed better results, being presented the results for pedestrians instances.

The comparison of pedestrians between the current and previous instances of the same object along time on as a mean value of 0.936 and can be visualized on 59.

The comparison of pedestrians between the first instance and next ones of the same object along time had a similarity loss of 2.6%, as shown on 60.

A confusion matrix between objects was made, and an example containing ten instances is presented at 61. A threshold based on the mean with the standard deviation of the diagonal values was defined.

$$threshold = \mu - \sigma$$

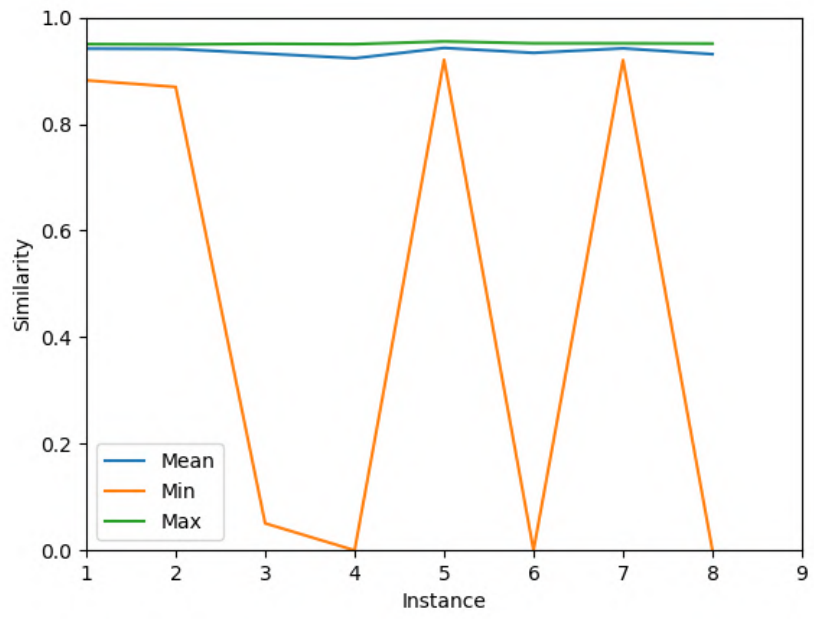


Figure 59: Fourier transform similarity with the current and next instance of the same object

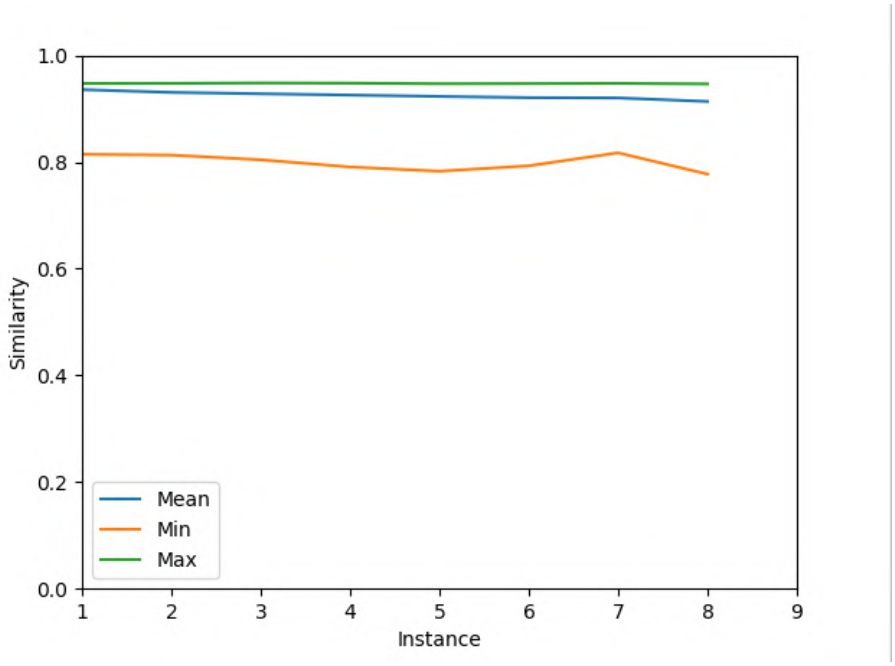


Figure 60: Similarity with the current and next instance of the same object

With the defined threshold, it was reached an F1 measure of 0.667, precision of 0.529, and a recall of 0.90, an excerpt of 10 objects is presented at 62.

The computational time results are the following:

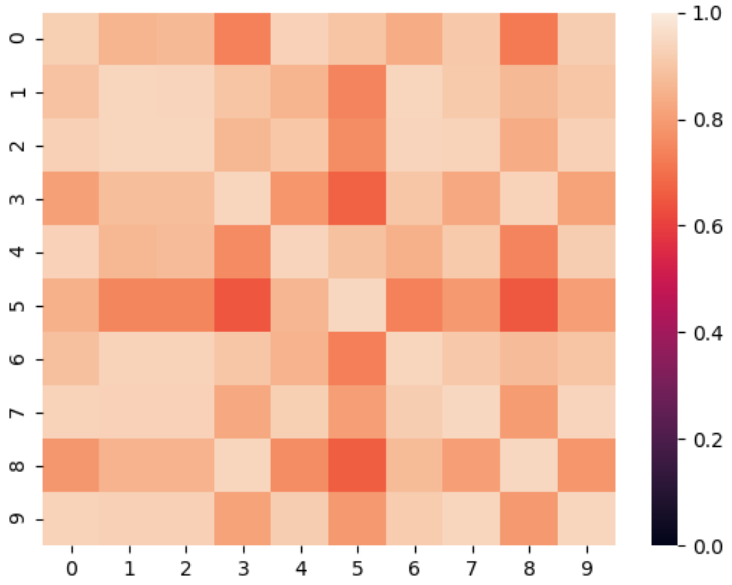


Figure 61: Confusion Matrix between objects using Fourier Transform

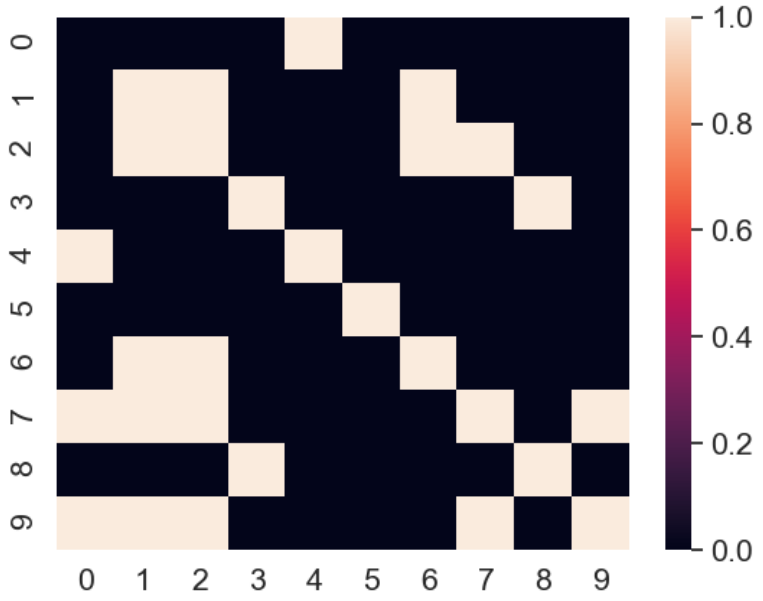


Figure 62: Confusion Matrix between objects using Fourier Transform

This technique presents consistent results. The similarity between the same object does not vary significantly. This technique also could not distinguish clearly instances from other objects instances.

Table 4: Time measurement of using Fourier Transform for image similarity between two images

Mean	Standard Deviation	Min	Max
36 ms	33 ms	10 ms	232 ms

Autoencoders

The idea beyond calculating a similarity between images using autoencoders is to compare the activation nodes of the encoded layer of both images. This similarity consists of comparing the vectors with a euclidean distance and using Principal Component Analysis (PCA) to find a different distance.

The model was trained with the dataset created for pedestrians and cars, and a previous model was defined with the most dataset.

When comparing the similarities, the results do not permit to establish a correlation of similarities. With the model of the MNIST dataset (that reached a loss of 0.14), the low similarity extracted between the same numbers of MNIST was identical to not equals numbers, being the distances 11.4 and 11.2 respectively using the PCA. These results showed that three main problems could be happening, the model may be wrong trained, the similarity method is not suitable, or the model learns the features in a way that difficult his comparison.

Works with autoencoders such as Theis et al. (2017), Gondara (2016), and Vincent et al. (2008) use this technique to have good denoising results and a reasonable compression data rate, being used only the output layer and not the encoding layer. The behavior of the network to encode and decode seems to fundamentally compress the information from the input image, not learning great relevant features of the image and reconstruct it itself. Although the lack of good results, the use of autoencoders as a high feature retrieval and image generation as a generative model is still open to discussion.

Conclusion

All methods presented different types of results.

SIFT method is irregular detecting similarity between the same object and could not distinct instances from other objects instances.

The Fourier transform detects high similarity between the same object but also detects significant similarity comparing with different objects.

The autoencoder showed a lack of good results. This technique tends to compress the data and learn to perform the invert operation. To achieve the results pretended, the model would have to create a high feature retrieval model (encoding phase) and image generation

as a generative model (decoding phase). However, to achieve the pretended model, more research on the model architecture is needed.

The color co-occurrence histogram method presents good results, being capable of having a high similarity between the same object along time and be capable of having a useful distinction comparing an object with different ones.

The use of more than two methods is not seen as a definite advantage. The color co-occurrence histogram is the best method tested considering two rows and only one column for pedestrians and vehicle images.

4.3 LIBRARY

The library allows extracting features from a video. The video processing analysis is permitted via a video, an IP camera, or a connected camera. The video analysis visualization when running the library is available. The main libraries for drawing were the python pillow package and OpenCV.

4.3.1 *Features*

The features extracted from this library are what differentiate this work from similar works, in addition to the pipeline application features are extracted that accelerate and promote the use of this library.

The features are the following:

- Detection boxes;
- Counter;
- Area definition;
- Line intersection;
- Heatmap;
- Collision;
- Object recovery;
- Dataset creation;
- Panorama Stiching.

How they work is described will be described individually on the next sub sections.

Detection Boxes

A fundamental feature is the creation of the detection boxes. The detection boxes from this work are different from the simple detection boxes given by a detector. This feature uses the pipeline and gives the location of the detection box, the label, the ID, and the prediction location. An example can be view on 63.

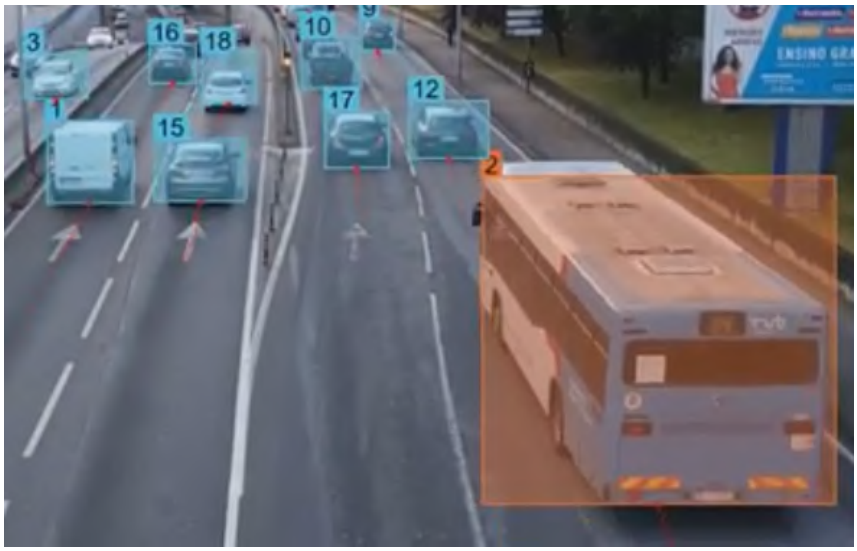


Figure 63: Detection boxes library feature

Counter

The counter feature is a feature that permits to count the number of objects presented in real-time. Options were developed, such as unique count objects, count along time, and which type of objects. The data about this feature is made available by the library and is also presented in the video.

Area definition

The area definition feature associates the counter feature in a polygon defined by the user. This permits us to know if objects in the determined area and how many, the polygons can be a circle, a rectangle, or a polygon with n values. This feature shares all features from the counter feature and enables a smart city environment to analyze parking lot spaces, how many people are on a game of court tennis, among others. The data about this feature is made available by the library and is also presented in the video.

Line intersection

The line intersection feature enables to create a line that divides the scenario and registers when the line is intersected and in which direction. This feature results in two counters, one for each type of intersection. The data about this feature is made available by the library and is also presented in the video.

Heatmap

A heatmap with the location of the object is permitted. The creation of this feature, seen on 64, it was saved the locations of the objects and displayed on matrix. To each location, it was considered the center bottom dot as the location of the object with a radius of 5-pixel units. These units were summed to a matrix with the size of the image initialized at 0.

To display the heatmap, a matrix of the number of locations is mapped to a color matrix based on his frequency, and these colors are merged with the original image.

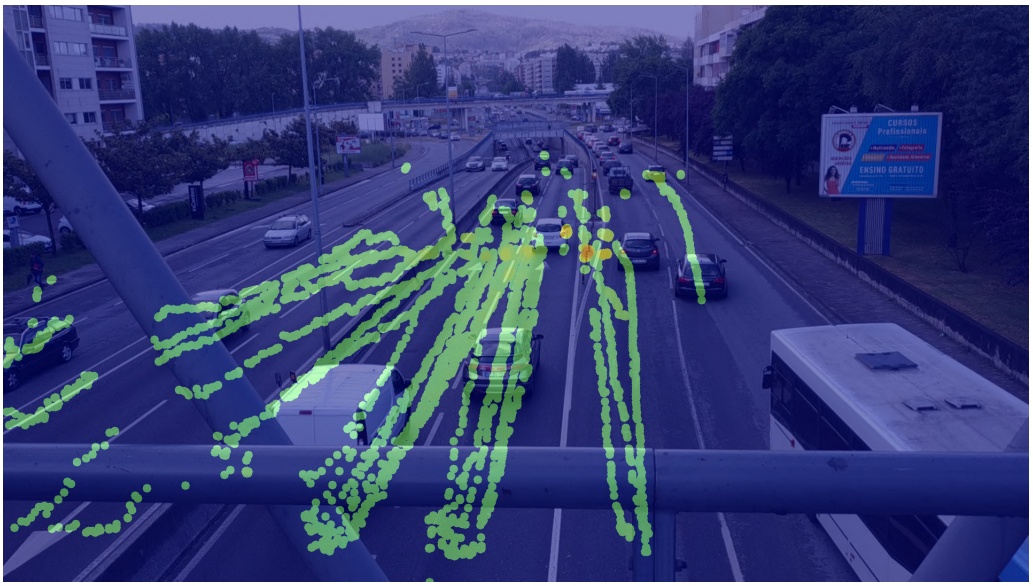


Figure 64: Heatmap library feature

Collision

The collision feature permits to detect object collisions. This feature detects the collision based on the object speed and route along time. If a significant speed drop is found, followed with box intersection of two objects is requested a collision. It is also possible to detect a probability of collision, and this option is made based on the speed of the objects and his route intersection. An example of an application in a smart city environment is the detection of vehicles collision. This feature is made available by the library and is also presented in the video.

Object Recovery

The object recovery feature is a feature that permits to detect an object that was previously detected, considering a long period. For this feature, an area can be defined, and all the objects that disappear in this area are stored. When a new object appears in this area, this object is compared with the previous with a color co-occurrence comparison, a threshold similarity was defined by default and can be changed via function argument on the code.

Panorama Stitching

If two cameras are monitoring the same scene is possible to have a panorama view of the scene, connecting the two cameras' input with the developed library.

The pipeline of the given feature is presented at 66.

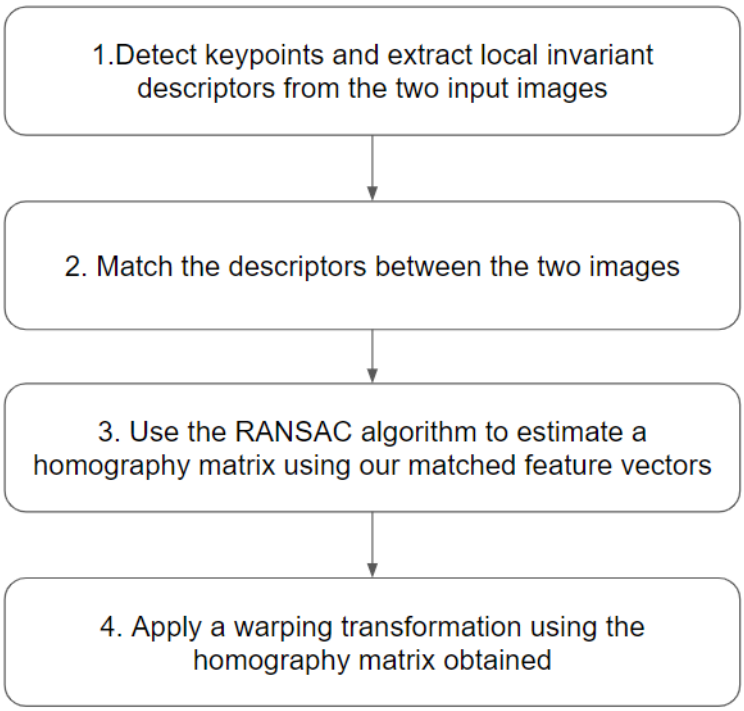


Figure 65: Panorma stiching pipeline

The pipeline starts with the detection of keypoints with SIFT, and the invariant descriptors from the two input images are extracted. Next, the RANSAC algorithm **?** is performed to estimate a homography matrix using the matched features, the match feature vectors. To stitch the video is with the matched vectors is applied a warping transformation using the homography matrix obtained.

This feature can only be applied to fixed cameras because performing keypoint detection, local invariant description, keypoint matching, and homography estimation is a computa-



Figure 66: Panorma stiching example; on the top row the two images to stich; on the middle row the matching descriptors; on the bottom row the panorama stich

tionally expensive task. Applying this pipeline on almost every frame for real-time results, it is only possible for computers with extreme resources.

This feature opens more use case scenarios for the application of the library. One specific scenario studied was the application of the library in a bus for monitoring in which bus station people exit from the bus given its entrance. For this problem, the library with the two cameras setup features permits to have an unobstructed panorama of the bus, using the library's main pipeline to detect when the people begin the journey, track its position and detect when they leave the bus. This gives the opportunity of instead have a range of 140 degrees the possibility to achieve 180 degrees.

Despite the various positions that can be taken by the cameras, choosing the placement of the cameras in front of the exit door seemed to be the best starting point for the study, as seem in 67.

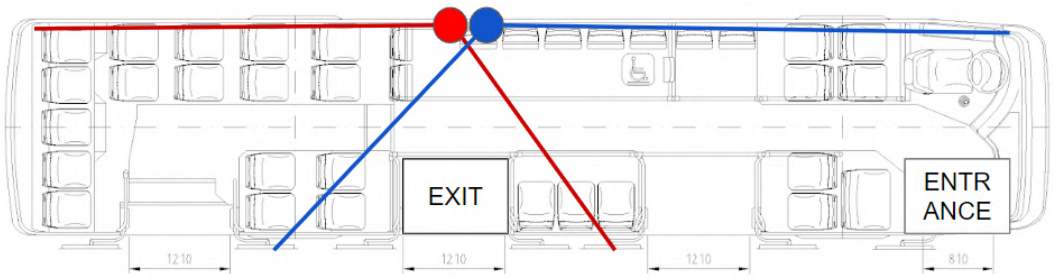


Figure 67: Two cameras setup on a bus for a panorama stiching

Dataset Creation

It was created functionality to create a dataset with the objects detected. This feature was used to create the fingerprint study dataset, creating one for pedestrians and another for vehicles.

This functionality organizes the objects detected by folder, and on each folder are the instances of each occurrence of the object detected. It is possible to limit the number of occurrences of each object, and if the object were detected more than the limit given, detections would be passed to arrange only the number of objects wanted. For example, given a limit of 20 occurrences and detecting an object 40 times, the dataset will contain the 20 instances by an interval of 2 detections to better diversify the dataset.

4.3.2 *Project Structure*

The project structure was defined accordingly to the facade design pattern and is presented at 68.

4.3.3 *Library distribution*

The library is available via the python package manager (PyPi) at `https://pypi.org/project/o4/`, and it is a public repository on GitHub at `https://github.com/CarvalhaisMatos/o4`.

For the development of the library, some python libraries were used, the libraries are presented above and are available on `https://pypi.org/project/pip/`.

4.4 WEB APPLICATION

A web application was built using the library created. This web application analyses real-time videos and presents the information processed mainly in graphs. All content presented is oriented to the library features. The users of the application are the video owners that define which type of features are available. These users can see all the data that are available for each feature and can choose if the data is available for non-registered users for public purposes.

The authentication method chosen was token-based authentication.

This section presented the architecture decisions and model, the chosen technologies, and the features available by the application.

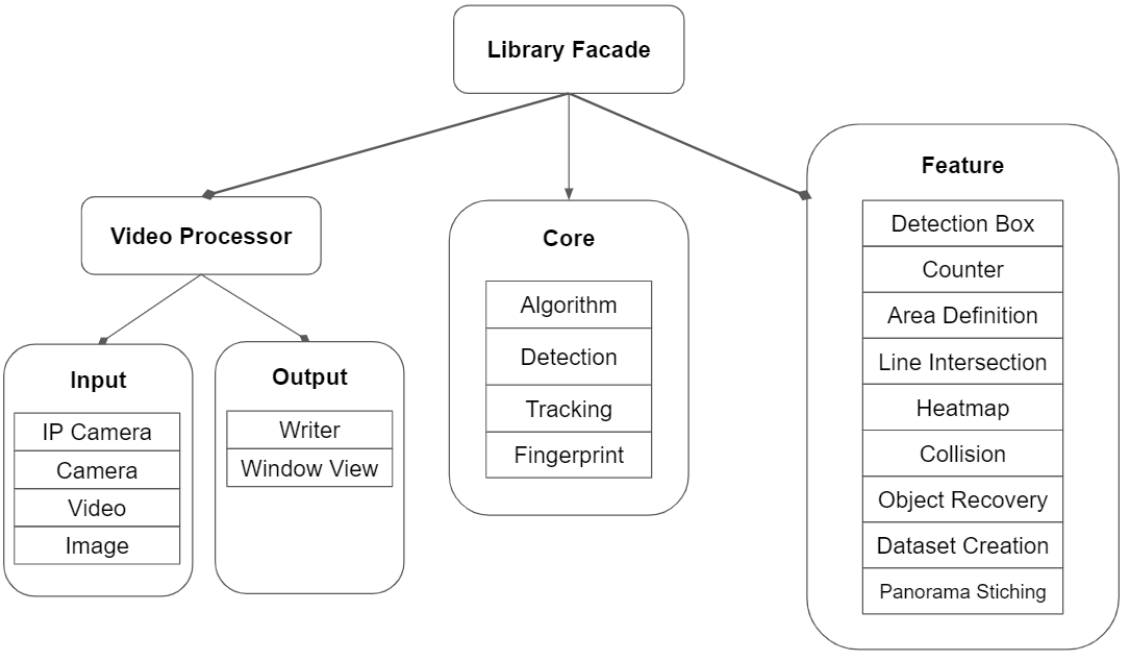


Figure 68: Library structure

Architecture

The architecture chosen for this application was a microservice architecture Sill (2016). This architecture choice was due to the high performance needed to process the video, having the necessity to divide the processing video logic with the data management logic.

The architecture consists of an API and a frontend application. The API contains two microservices and an API gateway. One of the microservices has a database. The frontend server is just a server that returns the web content of the application. The video camera content is published to the API.

From the two microservices, one has the focus to process the video and another the focus to manage the data access, and this last microservice could be divided into three microservices, one for authentication, one for pulling and publishing the video data and another for business logic. However, the choice of not decouple is given to the very low complexity given by each of these components and the not prospection of grown in complexity. Another main reason is the non-expectancy to have much traffic on this microservice.

The architecture diagram is presented at 75.

The authentication of the system relies on token authentication Burr et al. (2004). A token is sent in the "Authorization" header of the HTTP protocol in each request. This token is created when the login operation is performed, and this operation requires the user and password to be sent.

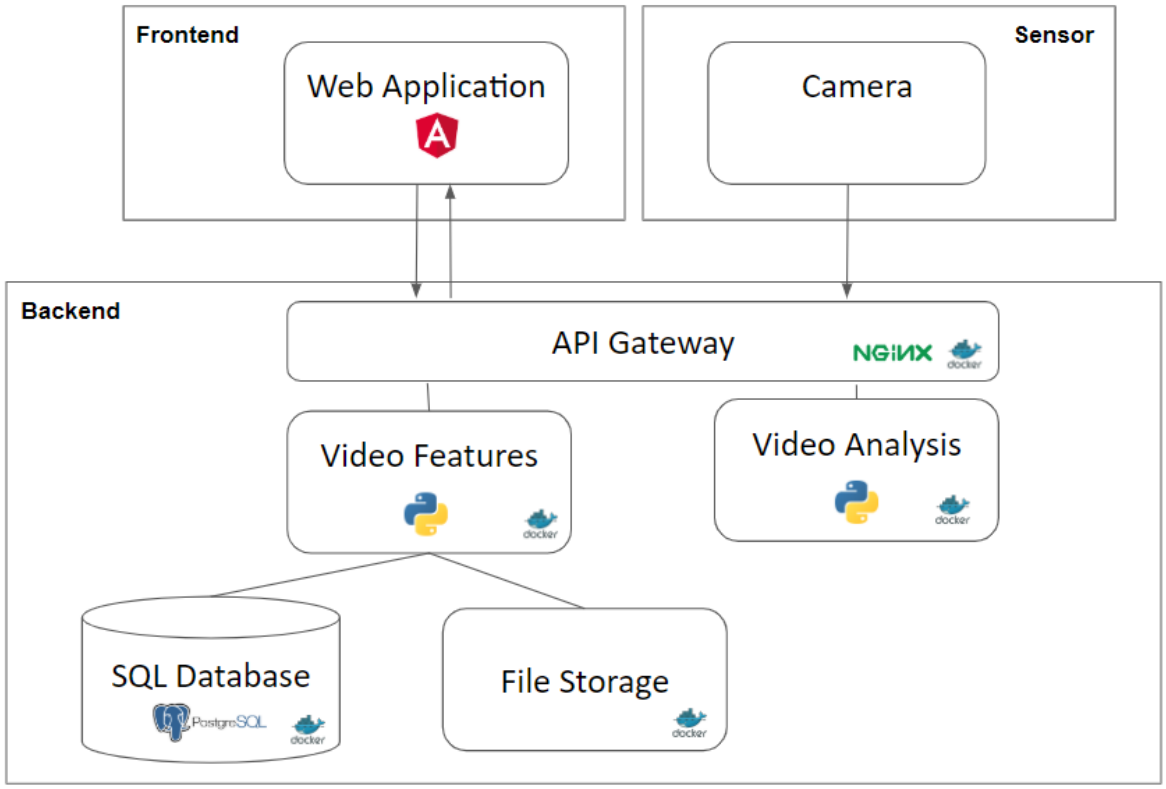


Figure 69: Web application architecture diagram

The system deployment has the main focus to be deployed on a client infrastructure and the connecting to the camera be to an IP camera via HTTP pulling within a time interval of 300ms.

Technologies

The technologies used on this project were chosen based on the productivity that these technologies can offer and the low dependency level of other libraries.

The main technology used for each component are presented in the following table:

Each main technology component required other technologies. The following technologies were used:

- HTML, used for the structural design of the application;
- CSS, used to give style to the HTML defined;
- Javascript, this programming language was used in all the frontend logic;
- Python, this programming language was used on the backend for all the business logic;

- Docker, this technology was used to containing each component in a virtualized container. This enables clear modularity of each component and as an advantage, an easy setup of the system.

On the frontend were used libraries that enriched the content design and user interaction. These libraries were the following:

- Bootstrap, for component design;
- Jquery, used as a simple API to interact with HTML;
- Chartjs, permitted the creation of charts and graphs.

Features

The features of the web application are all oriented to the library features being the features available, the object counter along time and on the scene, the option to select an area, the line interceptor, the object collider, the object speed, and the object recognizer.

More high level features for the context of a smart city were developed, such as traffic conditions, this indicates the level of traffic condition (low, medium, or high), other features added was the possibility to see the history of detections based on the weekdays and hour of the day.

For all the features is possible the see the correspondent value in a time-frame until two months.

The graphical types available are bar charts, color table charts, and line graphs.

The actions performed by the user are presented on a use case diagram available on 70.

Database

The database models are presented at 71.

4.5 SUMMARY

The algorithm pipeline is the following.

The detection algorithm chosen was the YOLO version 3 detector. Other detectors were considered like the single shot detector, the faster-RCNN, the retina net, and the mobile net. This detector was chosen given its accuracy and relatively fast inference compared with other detectors. The detector classification is based on a convolutional neural network. When a frame is read, it is capable of retrieving the detections boxes and the given label. This detector is not perfect, has lots of false positives, detects the same object multiple times on a frame, and has some minor problems with shadows, however with some adjustments and configurations, this detector has reliable results.

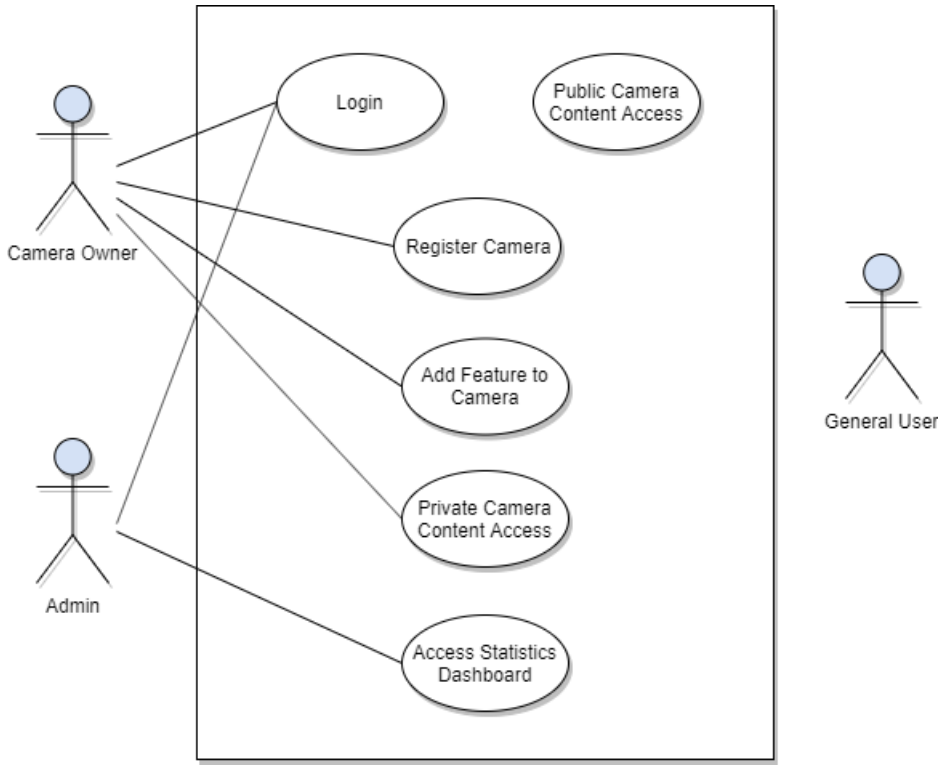


Figure 70: Use Case diagram of the web application

The first adjustment is the box size correction. The library considers a box size minimum and maximum given the scenario. This little adjustment improves the false positives on some frames, eliminating big boxes candidates that lead to a group of objects inference like 'person' on ten persons with significant distance from each other and also eliminating small boxes that are correlated with classification uncertainty.

The second adjustment is the elimination of the multiple object detection of the same object. To solve this problem is used the Non-Maxima Suppression (NMS) technique Neubeck and Van Gool (2006). The NMS technique considers the classification label confidences and the area of intersection of the multiple boxes and chooses the right detection box for the object.

For tracking, two methods were developed to reinforce the robustness of the algorithm considering the object movement and the object content image. It was used the Kalman Filter (KF) for predicting the object location; it was given to the KF that the detection box has some noise because the detector not makes a perfect box adjustment to the object. It was used the color co-occurrence histogram (CCoH) also to consider the object image content. The CCoH divides the image into cells and extracts a color histogram from each color channel cell; other techniques were consider are presented at the fingerprint study of this work with its benchmark.

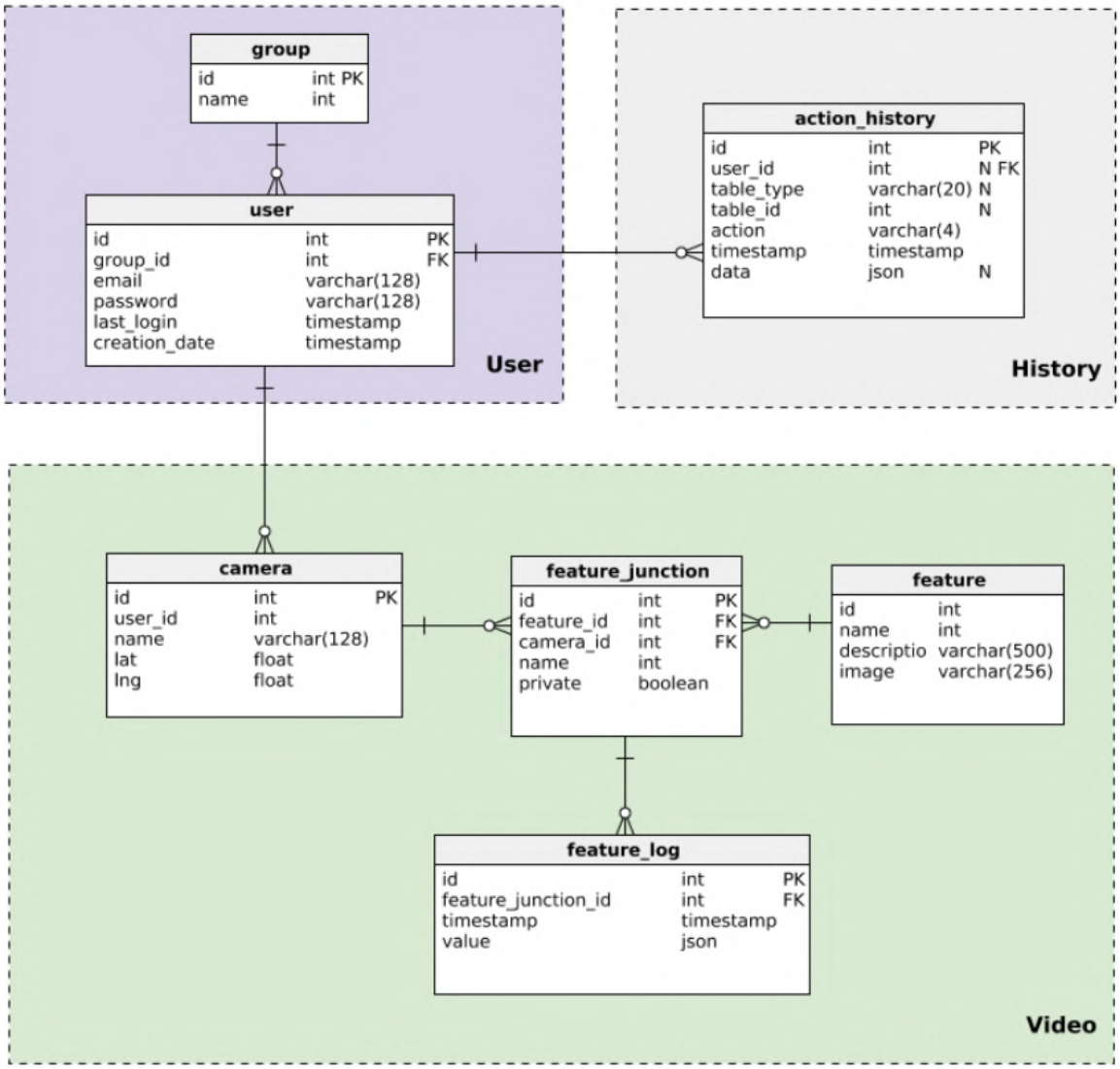


Figure 71: Database diagram of the web application

When a new frame is processed and after the detection boxes and labels are inferred arises the problem of how to assign the old objects to the new objects. These new objects are compared with all the old objects with the Kalman Filter and the CCoH. Thresholds were defined in the Kalman Filter not to permit the CCoH comparison if it is the same object. The similarity function is defined by:

$$similarity = K * C$$

,

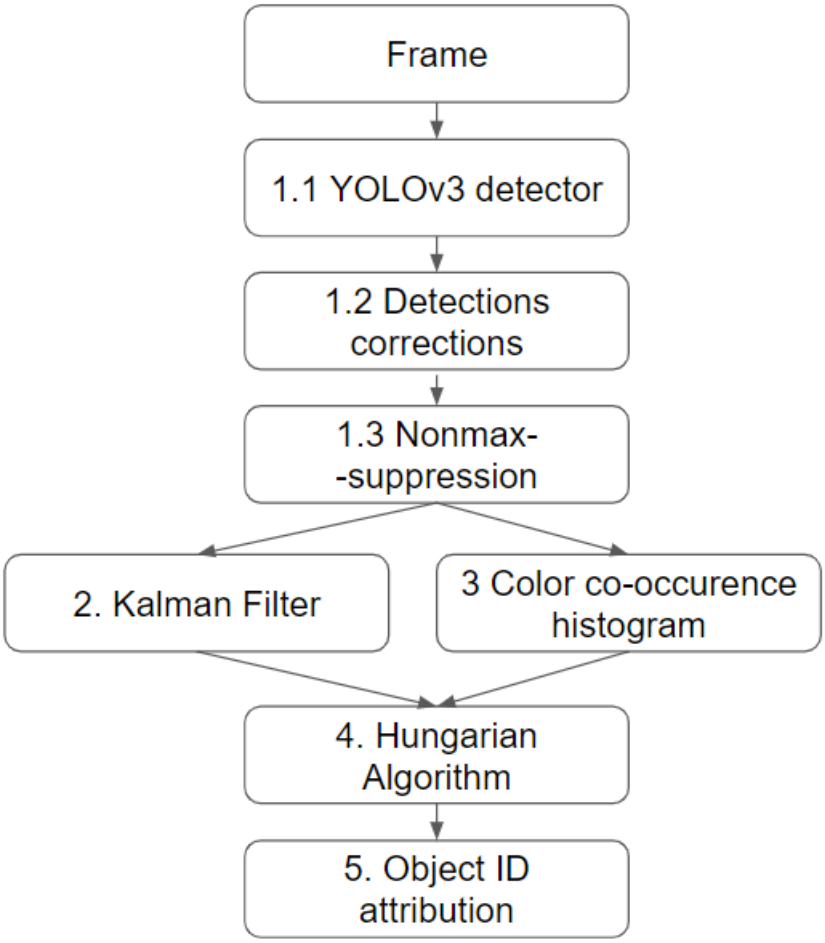


Figure 72: Algorithm pipeline

where K is the distance to the Kalman Filter prediction, and C is the CCoH similarity value. This results in a matrix of similarity. If the object passes the threshold defined for the K or C the object is considered a new object.

To assign each object to the right object is used the Hungarian Algorithm Kuhn (1955). This algorithm permits the correct assignment given the similarity matrix.

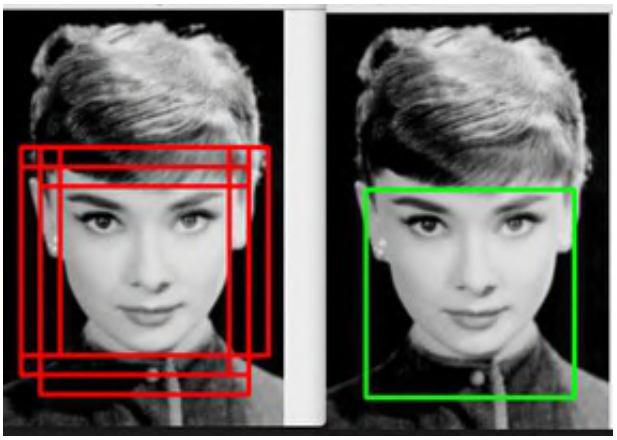


Figure 73: Non-Maxima Suppression example
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors/yolo.html

Table 5: Python lisbrary dependencies and version

Library	Version
absl-py	0.7.1
asn1crypto	0.24.0
astor	0.7.1
certifi	2019.6.16
cffi	1.12.3
chardet	3.0.4
cryptography	2.7
cycler	0.10.0
gast	0.2.2
grpcio	1.16.1
h5py	2.8.0
idna	2.8
imutils	0.5.2
Keras	2.2.2
Keras-Applications	1.0.4
Keras-Preprocessing	1.0.2
kiwisolver	1.1.0
Markdown	3.1.1
matplotlib	3.1.0
mkl-fft	1.0.12
mkl-random	1.0.2
munkres	1.1.2
numpy	1.16.4
olefile	0.46
Pillow	6.0.0
protobuf	3.6.0
pycparser	2.19
pyOpenSSL	19.0.0
pyparsing	2.4.0
PySocks	1.7.0
python-dateutil	2.8.0
pytz	2019.1
PyYAML	5.1
requests	2.22.0
scipy	1.2.1
six	1.12.0
tensorboard	1.10.0
tensorflow	1.10.0
termcolor	1.1.0
tornado	6.0.2
urllib3	1.24.2
Werkzeug	0.15.4
win-inet-pton	1.1.0
wincertstore	0.2

Table 7: Web application components

Component	Technology	Version
Web application	Angular	7.2.11
API Gateway	Nginx	1.16
Video Features Microservice	Django	2.1
SQL Database	PostgreSQL	11.5
Video Analysis Microservice	Django	2.1

5

RESULTS

In this section are presented the results of the pipeline developed in this work to cars and pedestrian videos. The videos chosen for this task are videos for object detection and related tasks. This experiment is expected to extract the accuracy of the pipeline, the difference of the methods applied, and the average framerate.

5.1 EXPERIMENT SETUP

All the video analyses were produced on a computer with an Intel Core i5 8250U and NVIDIA GeForce MX150.

Two different videos were chosen to test the pedestrian and the car accuracy being in a total of 4 videos.

5.2 RESULTS

For each video, it experimented with the extraction of the number of objects (pedestrians and cars) with three solutions: the first with the Kalman Filter, the second with the fingerprint method, and the third one with the full pipeline.

Table 8: Results on pedestrians videos

Video	KF Error	Fingerprint Error	Full Pipeline Error
oxford	9.4%	13.9%	7.3%
university	10.1%	17.0%	5.8%

The frame rate achieved with the full pipeline was 4 FPS. The error rate is primarily due to false positives detections by the detector along with the scene, detecting more pedestrians than the expected. Some occlusions with a significant time interval (superior to 5 seconds)

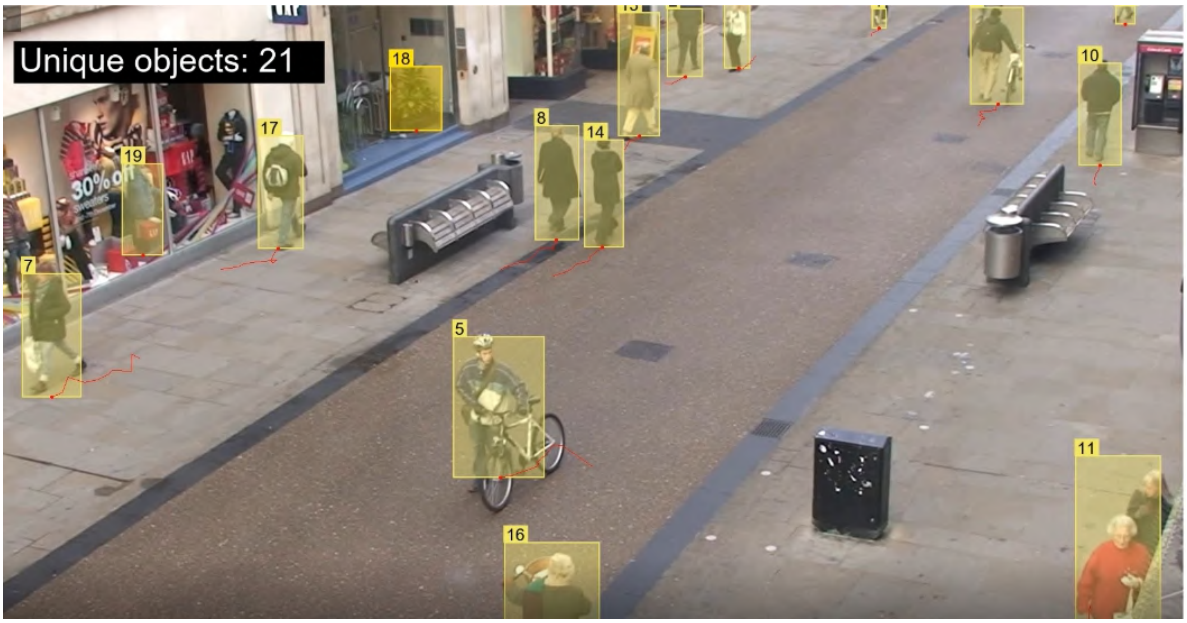


Figure 74: Results on oxford pedestrians video

also increase the pedestrians detected, considering that the occluded pedestrian for a long time is new.

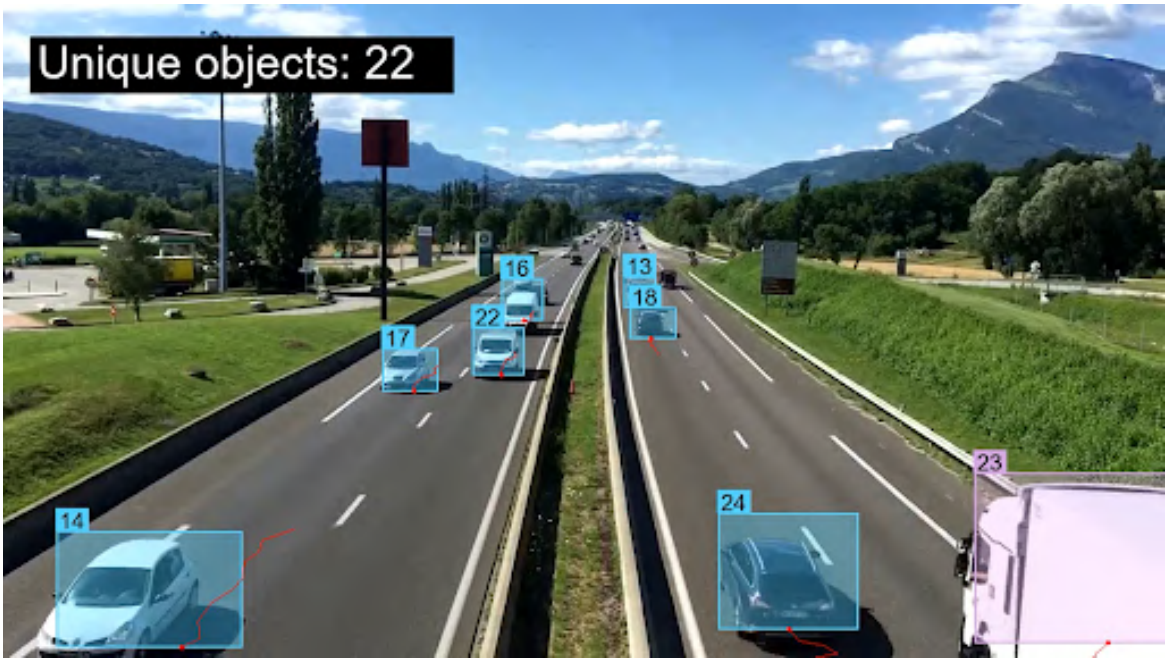


Figure 75: Results on highway vehicles video

The frame rate achieved with the full pipeline was 4 FPS. The results demonstrate that the Kalman Filter is the main responsible for the results given that there are no occlusions

Table 9: Results on car videos

Video	KF Error	Fingerprint Error	Full Pipeline Error
Highway	1.5%	8.2%	1.5%
Road	2.3%	3.9%	2.3%

and intersections in the videos, which is an expected result. The error rate is primarily due to lack detections by the detector, being some trucks the objects that were not detected the fingerprint.

5.3 SUMMARY

On the videos of the cars, the good results are justified by the lack of occlusions and easy prediction, the lack of some detections such as large trucks is the main reason not to achieve 100% positive results, being the Kalman Filter the leading cause for the success of the pipeline. On pedestrian's videos, the fingerprint method evidence it is usefulness in the pipeline and demonstrates that enrich the pipeline, especially after occlusions moments.

The framerate is very poor with the current setup with an average of 3 FPS, being the detection algorithm the leading cause of the low framerate, despite with a better GPU and processor it is possible to achieve 30 FPS, the 3 FPS in a real-time scenario is sufficient to perform object counting.

6

CONCLUSION

This chapter presents the conclusion and, on future work, which can improve scientifically given the problems explored, are presented.

6.1 CONCLUSIONS

The focus of this work was the development of a library to count pedestrians and cars in a video for a smart city environment. The library contains an algorithm that consists of a pipeline of computer vision and deep learning methods that serves as a fundamental base for the application of features to video monitoring. The features are all focused on a smart city environment.

The pipeline algorithm that detects and tracks objects was developed, defining a pipeline of methods that includes object detection, classification, tracking, and fingerprint. A study was made comparing techniques and presenting the results to find the best fingerprint methods. The considered methods for this study were the SIFT, the Fourier Transform, the Color co-occurrence histogram, and the autoencoders. The technique with the best results for similarity comparison between different and the same object was the color co-occurrence histogram. The fingerprint technique in the algorithm pipeline permits a more robust object tracking and the development of unique library features.

The final pipeline consists of using a convolutional neural network-based detector,the YOLOv3, with some minor improvements to reduce the false positive rate, for the tracking are used two techniques, the mean shift and the combination of the Kalman filter, that gives the location prediction of the object, with the color co-occurrence histogram that considers the content of the image.

The library was developed in python, and his installation is permitted via the python package manager (PyPi). This work also resulted in two public repositories available on Github, the library repository, and the fingerprint study repository.

A web application was made using the developed library. The architecture and the solutions for the main problems are presented. The application will be used in real-world scenarios to contribute to the evolution of a more smart city environment.

6.2 Prospect for future work

The use of autoencoders as a technique for image similarity didn't show good results, the behavior of the network to encode and decode seems to fundamentally compress the information from the input image, not learning great relevant features of the image and reconstruct it. However, the use of autoencoders as a high feature retrieval and image generation as a generative model can have the potential for this purpose, seemed like a favorable deep learning technique for this type of problem.

The main improvement to the algorithm pipeline would consist of a development of the object detection module. The object detection module is highly dependent on a convolutional neural network-based object classifier, that throws errors, meanly true positives. The state of the art detectors does not consider the whole image and relations between objects that consist of information loss and a not perfect classification in some scenarios. Another problem of the detector is the amount of data needed to train and the difficulty to find the right neural network architecture given that is computationally expensive in time and resources, although this is mainly convolution neural networks problems and some solutions like augmented dataset creation and the use of recurrent neural networks to find a good architecture these solutions also bring issues that are open to future works.

BIBLIOGRAPHY

Detecting humans in dense crowd images. URL http://crcv.ucf.edu/projects/crowd/.

Object fingerprinting. Master's thesis, University of Groningen, the Netherlands, 2009.

Alaa E Abdel-Hakim and Aly A Farag. Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978–1983. IEEE, 2006.

Gidudu Anthony, Hulley Greg, and Marwala Tshilidzi. Classification of images using support vector machines. *arXiv preprint arXiv:0709.3967*, 2007.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

William Burr, Donna Dodson, and W Polk. Electronic authentication guideline. Technical report, National Institute of Standards and Technology, 2004.

Xianbin Cao, Changxia Wu, Pingkun Yan, and Xuelong Li. Linear svm classification using boosting hog features for vehicle detection in low-altitude airborne videos. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2421–2424. IEEE, 2011.

Sung-Hyuk Cha and Sargur N Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370, 2002.

Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, volume 2, page 8, 2011.

COCO. URL http://cocodataset.org/.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Pierre Del Moral. Non-linear filtering: interacting particle resolution. *Markov processes and related fields*, 2(4):555–581, 1996.

Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

Fortune. http://fortune.com/2018/10/28/in-china-facial-recognition-tech-is-watching-you/. URL http://fortune.com/2018/10/28/in-china-facial-recognition-tech-is-watching-you/.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

Bernard Ghanem, Tianzhu Zhang, and Narendra Ahuja. Robust video registration applied to field-sports video analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 241–246. IEEE, 2016.

Daniel Gordon, Ali Farhadi, and Dieter Fox. Re^3: Re al-time recurrent regression networks for visual tracking of generic objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

Chengquan Huang, LS Davis, and JRG Townshend. An assessment of support vector machines for land cover classification. *International Journal of remote sensing*, 23(4):725–749, 2002.

Nusrah Hussain, Asifüllah Khan, Syed Gibran Javed, and Mutawarra Hussain. Particle swarm optimization based object tracking using hog features. In *Emerging Technologies (ICET), 2013 IEEE 9th International Conference on*, pages 1–6. IEEE, 2013.

ImageNet. URL http://www.image-net.org/.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

Yan Ke and Rahul Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2:II–II, 2004a.

Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004b.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

Christoph Palm. Color texture classification by integrative co-occurrence matrices. *Pattern recognition*, 37(5):965–976, 2004.

Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

Deva Ramanan, David A Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):65–81, 2007.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Alan Sill. The design and architecture of microservices. *IEEE Cloud Computing*, 3(5):76–80, 2016.

Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection using gabor filters and support vector machines. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 2, pages 1019–1022. IEEE, 2002.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

NY Times. Inside china's dystopian dreams: A.i., shame and lots of camer. URL https://www.nytimes.com/2018/07/08/business/china-surveillance-technology.html.

Arti Tiwari, Pradeep Kumar Singh, and Sobia Amin. A survey on shadow detection and removal in images and video sequences. In *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pages 518–523. IEEE, 2016.

Marin Van Heel. Similarity measures between images. *Ultramicroscopy*, 21(1):95–100, 1987.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

PASCAL VOC. URL http://host.robots.ox.ac.uk/pascal/VOC/.

Yongzheng Xu, Guizhen Yu, Xinkai Wu, Yunpeng Wang, and Yalong Ma. An enhanced viola-jones vehicle detection method from unmanned aerial vehicles imagery. *IEEE trans Intell Transp Syst*, 18(7):1845–1856, 2017.

Ruiqin Yan. Researches on hybrid algorithm for moving target detection and tracking in sports video. *Cluster Computing*, pages 1–10, 2018.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.