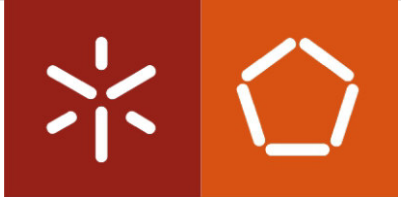
Universidade do Minho
Escola de Engenharia

Bruno Miguel Torres Lopes

Algoritmos de localização com informação histórica e realimentação dos utilizadores

Outubro de 2014

Bruno Miguel Torres Lopes **Algoritmos de localização com informação histórica e realimentação dos utilizadores**
UMinho | 2014



Universidade do Minho
Escola de Engenharia

Bruno Miguel Torres Lopes

Algoritmos de localização com informação histórica e realimentação dos utilizadores

Dissertação de Mestrado
Mestrado Integrado em Engenharia de Comunicações

Trabalho Efetuado sob a orientação do
Professora Doutora Maria João Nicolau
e do
Professor Doutor António Costa

Outubro de 2014

DECLARAÇÃO

Nome: Bruno Miguel Torres Lopes

Endereço Eletrónico: brunomiguel4@gmail.com

Tema da Dissertação:

Algoritmos de localização com informação histórica e realimentação dos utilizadores

Orientadores:

Professora Doutora Maria João Nicolau

Professor Doutor António Costa

Ano de Conclusão:

2014

Designação do Mestrado:

Mestrado Integrado em Engenharia de Comunicações

Universidade do Minho, _____/_____/_____

Assinatura: _____

Agradecimentos

Em primeiro lugar, gostaria de agradecer à Professora Doutora Maria João Nicolau e ao Professor Doutor António Costa por toda a orientação e apoio demonstrado ao longo deste projeto. Agradeço também pela motivação e disponibilidade que me permitiram finalizar com sucesso esta dissertação.

Agradeço a toda a minha família, em especial aos meus pais pela paciência demonstrada. Outra pessoa que sempre demonstrou paciência e me deu força e incentivo foi a minha namorada Filipa, a quem também quero dizer um muito obrigado.

Por fim, agradeço a todos os colegas e amigos que me acompanharam durante todo o percurso académico, em especial ao António, Pedro, João, Louis e Nuno, pelo companheirismo e, acima de tudo, pela amizade incondicional ao longo destes anos.

Resumo

Atualmente, a capacidade de localizar qualquer dispositivo móvel é uma questão de elevada importância, dado que tal informação pode ser usada num vasto conjunto de aplicações, como por exemplo, navegação, orientação de apoio ao turismo, monitorização ou segurança. Com o elevado aumento do número de serviços ou aplicações dependentes da localização, faz sentido pensar em novas formas de localização, preferencialmente mais exatas e precisas.

Em ambientes *outdoor* a localização de dispositivos já é assegurada, principalmente pelos sistemas de localização baseados em GPS (*Global Positioning System*). Em ambientes *indoor* podem ser utilizadas várias tecnologias sem fios para localizar-se dispositivos móveis, como por exemplo, RFID (*Radio-Frequency IDentication*), *Bluetooth*, UWD (*Ultra-Wide-Band*) ou WLAN (*Wireless Local Area Network*). Recentemente, tem-se investido muito em localizar dispositivos móveis numa WLAN, devido a várias razões: já é usada massivamente noutros serviços (principalmente para acesso à Internet), os equipamentos de rede são baratos e, na maioria dos casos, a sua implementação é facilitada através da utilização de uma infraestrutura já existente.

Uma das formas de localização usada nas redes sem fios baseia-se no mapeamento da potência do sinal recebido (RSS - *Received Signal Strength*), utilizando a técnica *fingerprinting*. Esta técnica tem como modo de funcionamento recolher, previamente, amostras de valores de RSS que serão usadas, pelos algoritmos de localização, nas estimativas de posicionamento dos dispositivos. Muitos algoritmos de localização baseados em *fingerprinting* já foram propostos e o mais popular é o algoritmo KNN (K *Nearest Neighbors*). Este algoritmo tem mostrado uma boa exatidão e precisão em vários cenários. No entanto, devido à existência de vários obstáculos dentro dos edifícios, como por exemplo, paredes, mobiliário, pessoas em movimento ou aberturas e fechos de portas e janelas, os sinais eletromagnéticos estão suscetíveis a vários fatores que os degradam. Além disso, em ambientes *indoor* existe maior probabilidade de ocorrerem interferências entre vários sinais eletromagnéticos e, como consequência, provocar uma maior degradação dos mesmos.

Com a utilização de informação histórica neste tipo de algoritmos pode-se minimizar, ou mesmo eliminar estes problemas. Para isso, é necessário sumariar e guardar o histórico dos dispositivos. Outra solução capaz de evitar os problemas referidos é utilizar-se o *feedback* dado pelos utilizadores do sistema. Nada melhor do que eles para avaliar os resultados calculados pelos algoritmos e, através dessa avaliação, melhorar os atuais e futuros resultados. Este documento descreve todo o trabalho realizado no desenvolvimento e implementação de algoritmos de localização que utilizam essa informação, bem como os resultados obtidos.

Abstract

Nowadays, being able to locate any mobile device is a major and important issue since localization information can be used for a broad range of applications such as navigating, tracking and monitoring applications. With the high increase of location services and applications it is pertinent to devise new ways to locate devices, more accurate and precise. For outdoor localization, almost all solutions are GPS based. For indoor localization several technologies are used such as, RFID, UWD and WLAN. The use of WLANs to locate devices has been subject of research in recent times and has increasing importance due to several reasons. It is already widely deployed for other services (being the Internet Access the most important), the equipment is cheap and also the deployment, since it is facilitated by the use of a pre-existing infrastructure.

One of the strategies that can be used to perform Indoor Localization using WLANs is based on the Received Signal Strength (RSS) using the Fingerprinting technique. Fingerprinting can be divided into two phases:an offline phase and an online phase. In the first phase, called the offline phase, RSS values are measured at each point of a given set of locations distributed inside the localization area. The information collected in this phase is afterwards used in the online phase to estimate a device location. In the online phase, RSS values obtaining by the device are compared with those previously observed in the offline phase in order to infer the device's position. Many algorithms, such as deterministic and probabilistic methods and even neural networks, have been proposed to compute device's location in online phase. KNN algorithm is the most popular deterministic one. In this algorithm, the position of mobile user is estimated by averaging the position of the K nearest points in fingerprinting map. The algorithm begins to compute de distance between the current state of the device and each point in fingerprinting map, using the measured RSSs. Then the K points with the lowest distance are chosen and the mean distance between then is computed and presented as the device's position. This algorithm has proven good accuracy in several scenarios, however due to the existence of obstacles in indoor environments, like walls and furniture, the signal propagation in an indoor environment is subject to reflection and diffraction, and usually the K nearest neighbors found are scattered and not close to each other. This causes some significant errors in the algorithm results in some situations.

The use of historical information may minimize or even eliminate this problem. This requires summarize and keep historical information of the devices. Another possible solution to avoid the mentioned problems is to use the users feedback. Nothing better than the system users themselves to evaluate the results provided by the algorithms and through this evaluation improve the current and future results. This document describes all the work done, building and developing several localization algorithms that use this information, as well as the results obtained.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

A-GPS	**A**ssisted-**G**lobal **P**ositioning **S**ystem
AP	**A**ccess **P**oint
AoA	**A**ngle **o**f **A**rrival
CDMA	**C**ode **D**ivision **M**ultiple **A**ccess
DGPS	**D**ifferential **G**lobal **P**ositioning **S**ystem
E-OTD	**E**nhanced **O**bserved **T**ime **D**ifference
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile Communications
HKNN	**H**istorical **K** **N**earest **N**eighbor
KNN	**K** **N**earest **N**eighbors
JDBC	**J**ava **D**ata**B**ase **C**onnectivity
LoS	**L**ine **o**f **S**ight
LTE	**L**ong **T**erm **E**volution
MAC	**M**edium **A**ccess **C**ontrol
PKNN	**P**redicted **K** **N**earest **N**eighbors
RF	**R**adio **F**requency
RFID	**R**adio-**F**requency **ID**entification
RKNN	**R**estricted **K** **N**earest **N**eighbor
RP	**R**eference **P**oint
RSS	**R**eceived **S**ignal **S**trength
RTT	**R**ound **T**rip **T**ime
SQL	**S**tructured **Q**uery **L**anguage
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess
TDoA	**T**ime **D**ifference **o**f **A**rrival
ToA	**T**ime **o**f **A**rrival
UWB	**U**ltra-**W**ide-**B**and

VAKNN	**V**elocity **A**ware **K** **N**earest **N**eighbor
VAP	**V**irtual-**A**ccess-**P**oint
WKNN	**W**eighted **K** **N**earest **N**eighbor
WLAN	**W**ireless **L**ocal **A**rea **N**etwork

Símbolos

f	frequência	Hz
m	metros	m
RSS	potência do sinal recebido	dB
t	tempo	s
v	velocidade	m/s

Capítulo 1

Introdução

1.1 Enquadramento

A utilização generalizada dos dispositivos móveis e das redes sem fios conduziram ao aparecimento de múltiplos e variados serviços, que tiram partido da mobilidade e da informação de localização dos utilizadores destes dispositivos. São os chamados serviços ou aplicações dependentes da localização, de que são exemplo aplicações para orientação de visitantes em museus e espaços públicos, aplicações para localização e monitorização de equipamentos em hospitais, aplicações de apoio ao turismo, aplicações na área da segurança, etc.

Nos últimos anos têm surgido várias soluções para a implementação de serviços de localização, soluções essas que utilizam vários tipos de abordagens e diferentes tecnologias para obter a informação de localização. No entanto, pouco ou nada se tem feito no que toca à descoberta e disponibilização desta informação de forma automática e transparente. Uma das vantagens de ter um serviço de localização suportado pela infra-estrutura de rede é a base de dados daí resultante. O servidor de localização mantém uma visão global de todos os dispositivos localizáveis no perímetro em observação e um histórico das informações de localização associadas. Essa informação inclui tanto os dados efetivamente obtidos (força de sinal) como as estimativas de posição fornecidas (coordenadas). Esse histórico pode ser sumariado e utilizado pelos algoritmos de localização com o intuito de melhorar os resultados. Pode-se, por exemplo, limitar o leque de novas coordenadas a devolver para um dispositivo com base na sua posição e velocidade observada no passado recente, ou definir algumas restrições adicionais como a impossibilidade de atravessar obstáculos. É também possível usar a realimentação dos

utilizadores (*feedback*), tanto de forma positiva como negativa, em relação à exatidão[1] e precisão[2] das estimativas que vão sendo fornecidas e usar essa informação de maneira a melhorar os resultados.

1.2 Objetivos

Este trabalho surge na sequência de um projeto anterior [1], no qual foi desenvolvido um sistema de localização suportado pela infraestrutura de rede. Os algoritmos de localização implementados nesse projeto são baseados na técnica de *localização fingerprinting*. Nas experiências efetuadas no trabalho anterior, e usando o algoritmo com melhores resultados a nível de exatidão, foi obtida uma distância média do erro de 2,71 metros.

Por sua vez, o principal objetivo deste trabalho é estudar a viabilidade da utilização do histórico do servidor de localização e do *feedback* dos utilizadores, de forma a melhorar os resultados obtidos no projeto anterior, quer a nível de exatidão, quer de precisão. Além disso, é importante garantir que os algoritmos desenvolvidos sejam facilmente implementados em qualquer cenário. Para que estas soluções sejam concebidas, é necessário dar resposta a um conjunto de desafios, entre os quais se incluem:

- Identificar formas de sumariar o histórico de informação e o *feedback* dos utilizadores nas estimativas de localização;
- Redefinir os algoritmos de posicionamento, tendo em conta essa informação adicional;
- Implementar os algoritmos propostos em Java;
- Obter resultados em ambiente operacional;
- Comparar os resultados com os algoritmos mais comuns em uso e concluir sobre as melhorias de precisão e exatidão obtidas.

[1] Grau de aproximação entre o valor de uma medição e o valor verdadeiro da grandeza medida
[2] Grau de variação de resultados de uma medição

1.3 Estrutura da dissertação

Esta dissertação encontra-se organizada em sete capítulos. No Capítulo 1 é feita uma introdução ao tema desta dissertação, apresentando-se um enquadramento deste, bem como os respetivos objetivos a alcançar.

O Capítulo 2 descreve o estado da arte dos sistemas de localizações, apresentando técnicas de localização existentes, os sistemas de localização mais conhecidos e, por fim, uma comparação entre eles.

No Capítulo 3 é apresentado o trabalho já realizado, no âmbito da localização em ambientes *indoor*, que utiliza informação histórica ou *feedback* dos utilizadores nos seus algoritmos de localização. Em concreto, são descritos, pormenorizadamente, dois sistemas de localização baseados no histórico do utilizador, bem como os resultados e conclusões obtidos pelos respetivos autores. De seguida, é feita uma comparação entre o uso ou não desse tipo de informação. Por último, é apresentado um sistema de localização que utiliza o *feedback* dos seus utilizadores, de forma a melhorar os resultados dos algoritmos.

Todo o trabalho realizado na conceção dos algoritmos de localização, mais propriamente, as soluções propostas para implementar a informação histórica e o *feedback* dos utilizadores, é abordado no Capítulo 4.

O Capítulo 5 apresenta o trabalho necessário para que fosse possível implementar e colocar em funcionamento os algoritmos de localização. Inicialmente, é apresentada a estrutura da base de dados, de seguida o cenário de teste utilizado, a aplicação de testes e, por último, os procedimentos técnicos realizados para a implementação dos algoritmos.

No Capítulo 6 são descritos os testes experimentais realizados e os respetivos resultados obtidos nos algoritmos implementados. Além disso, são comparados, de uma forma geral, os diferentes resultados obtidos e, também, testes complementares que ajudaram a compreender qual a existência de outros fatores que podem influenciar os resultados dos algoritmos.

Por último, no Capítulo 7, são apresentas as conclusões desta dissertação, resumindo os objetivos cumpridos, as possíveis modificações e sugestões de trabalho futuro.

Capítulo 2

Técnicas e sistemas de localização

Não é tarefa fácil prever o comportamento dos sinais eletromagnéticos em ambientes *indoor*, devido a diversos fatores que podem ocorrer neste tipo de ambiente, tais como, a propagação multipercurso[1] (*multipath*), baixa probabilidade de existência da linha de visão (LoS ou *line of sight*) entre emissores e recetores, existência de objetos em movimento, abertura e fecho de portas, superfícies refletoras, entre outros [2], [3]. Até hoje, não existe nenhum modelo ideal que descreva as características supracitadas, de forma a possibilitar a construção de um sistema de localização 100% exato. Existem apenas técnicas ou algoritmos que permitem uma aproximação aos valores reais, possuindo todas elas algumas limitações. Neste capítulo são apresentadas algumas dessas técnicas e sistemas de localização.

2.1 Técnicas de localização

As técnicas ou algoritmos de localização mencionados podem ser divididos em três tipos [3]: triangulação, análise de cenário (*scene analysis*) e proximidade. Nesta secção vão ser apresentadas e descritas cada uma das diferentes técnicas de localização referidas, assim como as respetivas vantagens e desvantagens das suas implementações em sistemas reais.

2.1.1 Triangulação

A técnica de triangulação baseia-se nas propriedades geométricas dos triângulos, de forma a calcular a localização de um objeto. Esta técnica possui duas variantes, pode

[1] Receber o mesmo sinal por múltiplos caminhos, devido a fenómenos de reflexão, difração ou espalhamento

ser baseada em lateração ou em angulação.

Na primeira derivação, é feita a medição do tempo de propagação do sinal entre o emissor e um ou vários recetores. Com estes valores medidos, e conhecendo-se a velocidade de propagação do sinal, é possível fazer uma estimativa da distância entre um dispositivo a localizar e um objeto de referência (emissor). Neste caso, também é possível fazer medições ao nível da potência do sinal recebido (RSS - *Received Signal Strength*), a fim de estimar a distância percorrida pelo sinal. Com a obtenção de pelo menos três distâncias é possível calcular a posição de um dispositivo. Fazem parte desta técnica os métodos *Time of Arrival* (ToA) e *Time Difference of Arrival* (TDoA), que serão abordados no ponto seguinte.

Já na técnica de angulação, o cálculo da posição de um objeto é efetuado através de uma medição angular, relacionando a direção de propagação do sinal entre o objeto e pontos de referência. O *Angle of Arrival* (AoA) é um método que utiliza esta técnica, e será abordado mais à frente.

2.1.1.1 Lateração

Time of Arrival (ToA)

Este método mede o tempo de propagação de um sinal entre o emissor e o recetor. De seguida, este tempo é multiplicado pela velocidade de propagação do sinal. O ToA pode ser aplicado de duas diferentes formas: medindo o tempo de propagação apenas num sentido (do emissor para o recetor) e, também, medindo o tempo de propagação do sinal a percorrer a distância nos dois sentidos (RTT - *Round Trip Time*) [4]. No RTT, o emissor inicia a contagem ao mesmo tempo que inicia a transmissão e espera pela resposta. Quando esta chega, o emissor pára a contagem do tempo, obtendo, assim, o tempo nos dois sentidos.

Quando se opta por medir o tempo de propagação apenas num sentido (ToA), é indispensável considerar duas situações. Em primeiro lugar, todos os emissores e recetores do sistema devem estar totalmente sincronizados. Além disso, é também necessário que os emissores enviem, nos seus respetivos sinais, informação do instante de tempo em que começaram a transmitir, para que, do outro lado, os recetores possam calcular o tempo de propagação com a máxima exatidão. Em casos em que isto não acontece, um pequeno desvio na medição do tempo na ordem dos μs pode causar erros de exatidão elevados.

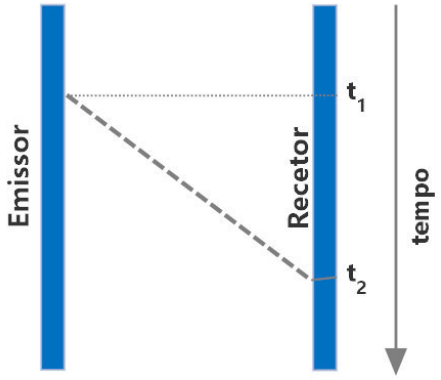


FIGURA 2.1: Cálculo da distância no método ToA

Na Figura 2.1 está representado um exemplo do cálculo da distância entre um emissor e um recetor pelo método ToA. Como se pode verificar, o emissor inicia a transmissão no instante de tempo t_1, e o recetor recebe o mesmo sinal do instante de tempo t_2. A distância percorrida (d) pelo sinal, neste caso, é calculada por $d = v(t_2 - t_1)$, em que v corresponde à velocidade de propagação do respetivo sinal.

Para se poder estimar a posição de um determinado objeto é necessário medir as distâncias em pelo menos 3 pontos de referência não colineares, como se pode verificar pela Figura 2.2.

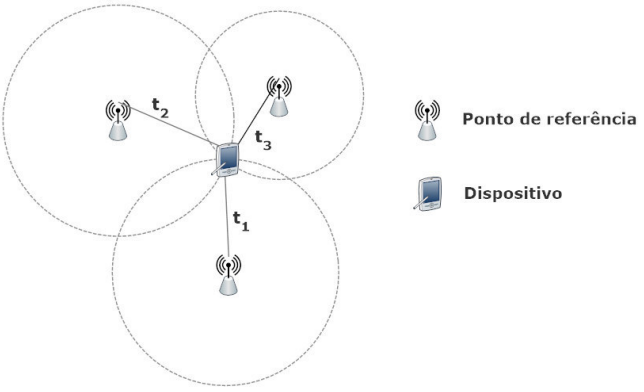


FIGURA 2.2: Cálculo de posicionamento pelo método ToA

O ruído do sinal e a propagação *multipath* podem prejudicar fortemente este método, podendo-se afirmar que esta é a sua principal desvantagem.

***Time Difference of Arrival* (TDoA)**

O método TDoA baseia-se na diferença dos tempos de chegada de dois ou mais sinais a um dispositivo, que foram emitidos simultaneamente. Existem duas técnicas distintas para implementação deste método. Na primeira, é medida a diferença de tempo que um sinal demora a percorrer a distância entre um emissor e três ou mais recetores de referência. Na segunda técnica, é calculada a diferença de tempo que dois sinais de natureza distinta demoram a percorrer a distância entre um emissor e um recetor. Segue-se uma breve explicação do funcionamento destas duas técnicas.

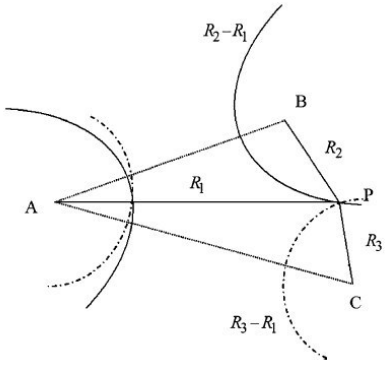


FIGURA 2.3: Localização baseada no método TDoA [3]

Na primeira técnica, em cada medição TDoA é calculada uma hipérbole, que representa a diferença de distâncias entre os dispositivos de referência. A posição de um determinado objeto é indicada através da interceção de duas ou mais hipérboles, como mostra a Figura 2.3. As duas hipérboles são formadas resultantes de três dispositivos de referência (A, B e C), para proporcionarem um ponto de interseção, que se encontra indicado no ponto P.

Como já foi referido, na segunda técnica são transmitidos dois sinais com características diferentes, o que faz com que os dispositivos precisem de estar equipados com hardware com capacidade para transmitir dois sinais em simultâneo. Na Figura 2.4 encontra-se exemplificada esta técnica. Pode-se verificar que o emissor transmite dois sinais em simultâneo e, devido a serem sinais de natureza diferentes, têm velocidades de propagação também diferentes, pelo que chegam ao recetor em instantes de tempo distintos. A distância é estimada por $d = (v_a - v_b)(t_2 - t_1)$, em que t_1 e t_2 representam o instante de tempo de chegada do sinal A e B ao recetor, respetivamente, e v_a e v_b é o valor da velocidade de propagação do sinal A e B, respetivamente.

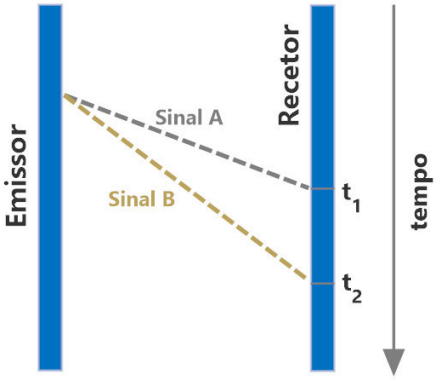


FIGURA 2.4: Cálculo da distância no método TDoA

Nesta secção foram abordadas as técnicas de localização baseadas em lateração mais comuns. No entanto, existem outras técnicas de lateração, tais como, o *Roundtrip Time of Flight* (RTOF) ou a técnica *Received Signal Phase Method* que, também, pode ser denominado por *Phase of Arrival* (PoA) [5].

2.1.1.2 Angulação

***Angle of Arrival* (AoA)**

No método AoA, a localização de um dispositivo é feita através da medição dos ângulos de chegada dos sinais. Uma forma de se obter o AoA, é com o uso de antenas direcionais em nós de referência e calcular o ângulo de chegada dos sinais do dispositivo a localizar. Posteriormente, a localização é conseguida através da interceção de linhas direcionais de pelo menos dois ângulos de referência. Este facto pode ser verificado através da Figura 2.5. Neste exemplo, o AoA utilizou dois pontos de referência (A e B) e foram medidos dois ângulos (θ_1 e θ_2), através dos quais é estimada a localização do dispositivo P.

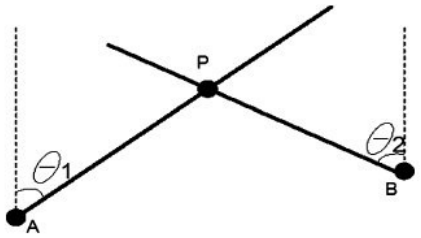


FIGURA 2.5: Localização baseada no método AoA [3]

Uma das vantagens do método AoA é o facto de este precisar apenas de realizar duas medições para estimar uma posição a duas dimensões (2D), ou três medições para

o caso de se pretender efetuar uma localização em 3D. Outra vantagem deste método é não necessitar de sincronização por parte das estações base.

Por outro lado, as desvantagens do AoA passam por este método requerer *hardware* bastante complexo e dispendioso, bem como ser suscetível ao efeito *multipath*. Esta última desvantagem afeta a precisão e exatidão na estimativa de posicionamento dos dispositivos, pois o sinal recebido por estes pode não ser o pretendido, mas sim cópias deste, o que provoca medições angulares erradas dos sinais de chegada.

2.1.2 Análise de cenário

O conceito de análise de cenário [3] refere-se a um tipo de algoritmos que, como o próprio nome indica, consiste em fazer uma análise prévia de um cenário, no qual se pretende implementar o sistema de localização. Nesta técnica, inicialmente é recolhida informação acerca das características do cenário (*fingerprints*) e, só depois, é estimada a localização de um objeto, fazendo-se uma correspondência entre os valores medidos no instante de tempo atual e os medidos previamente (*fingerprints*).

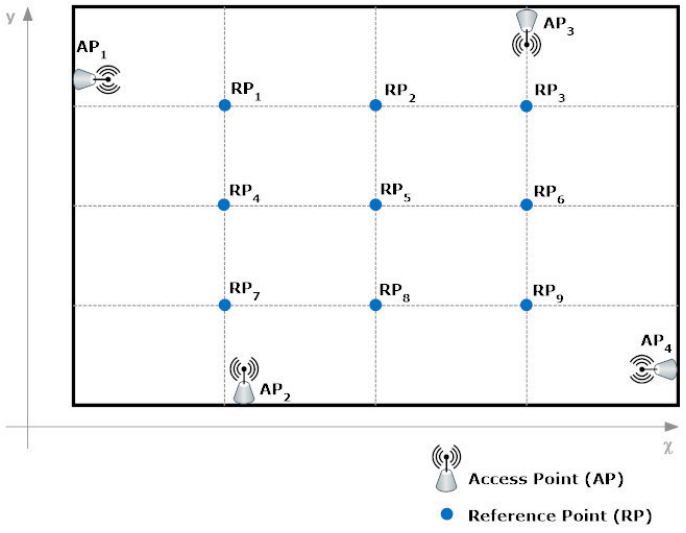


FIGURA 2.6: Cenário de um sistema baseado em *fingerprinting*

A localização baseada em *fingerprinting* consiste numa técnica em que é feito uma "impressão digital" de uma ou várias características de um sinal eletromagnético (geralmente a potência do sinal recebido) em determinadas posições de um cenário.

A técnica de localização baseada em RSS (*Received Signal Strenght*) *fingerprinting* é a mais utilizada pelos sistemas que utilizam o método de análise de cenário, e usa como característica a potência do sinal recebido (RSS). Existem duas fases na localização baseada em RSS *fingerprinting*: a fase *offline* (ou fase de treino) e a fase *online*.

Na fase de treino é feita uma análise do cenário no qual se pretende implementar o sistema de localização, como ilustrado na Figura 2.6. São recolhidas informações em determinados pontos de referência (RPs - *Reference Points*) do cenário, onde se registam os valores da potência do sinal recebido, por um dispositivo, dos vários pontos de acesso (APs - *Access Points*), que fazem parte do cenário. À medida que estes valores vão sendo medidos, são guardados numa base de dados, denominada de base de dados *offline*.

Por outro lado, na fase *online*, o sistema de localização utiliza os valores atuais da potência do sinal recebido do dispositivo a localizar, e os valores recolhidos na fase de treino, comparando-os entre eles. Utilizando-se propriedades de similaridade, pode-se calcular quais os pontos de referência que estão mais próximos do dispositivo a localizar e, posteriormente, estimar a localização desse mesmo dispositivo.

O facto de os sinais eletromagnéticos estarem sujeitos à propagação multipercurso, constitui uma desvantagem para as técnicas de localização baseadas em *fingerprinting*, pois estes fatores podem afetar os valores da potência do sinal recebido.

Existe uma certa variedade de algoritmos, que se utilizam na técnica de localização baseada no método RSS-*fingerprinting*, sendo que nesta secção vão ser abordados dois casos: o *K Nearest Neighbor* (KNN) [1], [3], [6], [7] e os métodos probabilísticos [3], [8].

2.1.2.1 *K Nearest Neighbor*

O algoritmo KNN [1], [3], [6], [7] é um algoritmo baseado em métodos determinísticos, cujo seu funcionamento resume-se em utilizar os valores de RSS medidos num determinado instante de tempo, ou seja, os valores da fase *online*, a fim de encontrar os K pontos de referência (RPs) mais próximos do objeto a localizar. Regra geral, o valor de K é escolhido pelo administrador do sistema. Os valores de RSS, observados nos pontos de referência, encontram-se registados na base de dados da fase *offline*. Os K pontos de referência mais próximos são denominados de K vizinhos mais próximos.

O processo para encontrar os K vizinhos mais próximos consiste em calcular a distância entre o dispositivo a localizar e cada um dos RPs existente no cenário, com base nos valores de RSS. Esta distância é medida através da distância euclidiana[2]. Quanto mais próximo se encontra um RP do dispositivo a localizar, menor será a distância euclidiana entre eles.

O vetor relativo à potência do sinal de um dispositivo num determinado instante de tempo (fase *online*) é representado por $\{RSS_1,\ RSS_2, \ldots,\ RSS_n\}$, onde RSS é a

[2]Distância entre dois pontos

potência do sinal recebido de cada AP (AP_1, $AP_2, \ldots,$ AP_n) e n o número total de APs existentes. Caso um determinado AP esteja fora do alcance de transmissão/receção, no instante de tempo em que é feita uma medição dos valores RSS, o seu valor será NULL. Assim sendo, para calcular a distância euclidiana do utilizador até ao RP_i, é usada a seguinte equação (2.1):

$$D_i = \sqrt{\sum_{j=1}^{n}(RSS_{ij} - RSS_j)^2} \tag{2.1}$$

Onde RSS_{ij} corresponde ao valor de RSS do RP_i relativo ao AP_j, ou seja, o valor que se encontra na base de dados *offline*. RSS_j é o valor de RSS que o dispositivo a localizar está a receber do AP_j, isto é, o valor obtido na fase *online*.

Através da ordenação de todas as distâncias, um certo conjunto de K RPs pode ser escolhido de acordo com as menores distâncias obtidas. O valor K pode ser escolhido entre 1 e o número total de RPs.

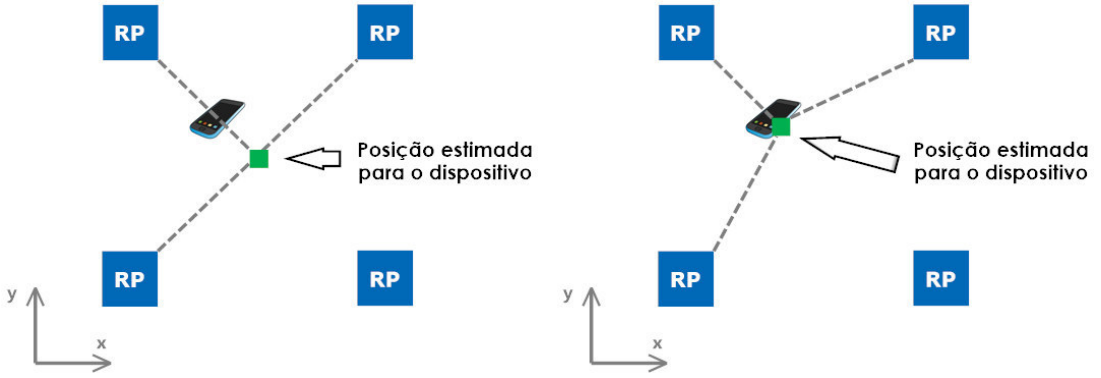
Por fim, a posição de um dispositivo é calculada através da Equação 2.2:

$$(x, y) = \sum_{i=1}^{K} \frac{1}{K}(x_i, y_i) \tag{2.2}$$

Onde (x_i, y_i) são as posições em coordenadas cartesianas dos K vizinhos selecionados e o resultado (x, y) é a posição do dispositivo a localizar, também em coordenadas cartesianas. Ou seja, as coordenadas do dispositivo a localizar são a média aritmética das coordenadas dos K vizinhos mais próximos.

No entanto, numa variação deste algoritmo denominada de *Weighted KNN* (WKNN) [6], [9], as coordenadas do dispositivo a localizar não são calculadas pela média aritmética dos K vizinhos mais próximo. Neste algoritmo, são atribuídos "pesos" consoante o valor da distância euclidiana a que se situa cada vizinho mais próximo. Como o dispositivo a localizar pode não se encontrar à mesma distância de todos K vizinhos escolhidos, logo a sua posição não é propriamente a médias aritmética dessas posições. Assim com o algoritmo WKNN, é atribuído um "peso" conforme a distância entre o dispositivo e os K vizinhos mais próximos, de forma a o aproximar dos que, supostamente, estão a uma menor distância. Na Figura 2.7 encontra-se exemplificado o que se pretende obter com a utilização do algoritmo WKNN em vez do KNN.

O fator "peso", matematicamente, é calculado pela Equação 2.3, em que D_i é a distância euclidiana do RP_i, num total de K RPs. Isto significa, que o fator "peso" dos



(a) No algoritmo KNN (b) No algoritmo WKNN

FIGURA 2.7: Exemplos de localização de um dispositivo com K=3

K vizinhos é atribuído através do cálculo do inverso do quadrado da distância euclidiana de cada K RP, dividindo-a pelo inverso da soma do quadrado das distâncias euclidianas de todos os K RPs.

$$w_i = \frac{\frac{1}{D_i^2}}{\sum_{i=1}^{K} \frac{1}{D_i^2}} \tag{2.3}$$

2.1.2.2 Métodos probabilísticos

A localização baseada em métodos probabilísticos [3], [8] consiste em encontrar, num conjunto de posições, a posição com maior probabilidade para a localização do dispositivo pretendido, tendo em conta uma função de probabilidade treinada com um conjunto de amostras corretas e erradas ou modelos de propagação dos sinais eletromagnéticos (ex: valores RSS observados). Nesta técnica, assume-se que existem algumas posições candidatas em que um dispositivo se pode encontrar, e é construída uma grelha com as probabilidades do dispositivo se encontrar em cada uma dessas posições, consoante os valores de RSS observados. A posição que possuir maior probabilidade acumulada é escolhida como a posição do dispositivo a localizar.

Ao contrário dos métodos determinísticos que, contêm um vetor que representa os valores de RSS observados numa dada posição, os métodos probabilísticos fornecem uma função de distribuição de probabilidades para cada posição. Na maioria dos casos, os métodos probabilísticos envolvem diferentes etapas, algumas bastante complexas, como por exemplo, fase de calibração do sistema, algoritmos de aprendizagem, ou cálculo do erro.

2.1.3 Proximidade

Na técnica de proximidade [3] ou, também, denominada de célula de origem [10], a localização de um objeto é estimada pela sua proximidade com um objeto de referência (ponto de acesso). O seu funcionamento baseia-se na existência de diversos pontos de acesso (por exemplo: antenas) com localizações bem conhecidas pelo sistema. Quando um objeto móvel é detetado por apenas um dispositivo de referência, assume-se que tal objeto se encontra nos seus arredores, mais propriamente dentro do seu raio de alcance, como demonstrado na Figura 2.8. No caso em que um objeto é detetado por vários dispositivos de referência, a localização desse objeto será dada pelo dispositivo cujo valor da potência do sinal recebido pelo objeto é o mais elevado.

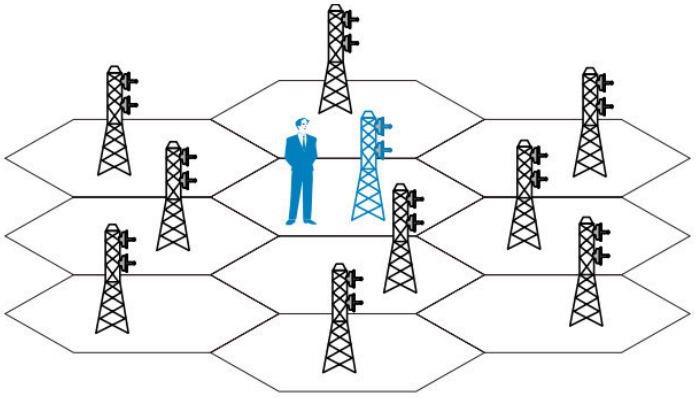


FIGURA 2.8: Localização baseada no método de proximidade [10]

Este método é de implementação relativamente simples, de baixo custo, e possibilita a sua aplicação em diferentes tecnologias de comunicação. É frequentemente implementado em sistemas de comunicação por infravermelhos (IR) e RFID (*Radio Frequency Identification*).

No entanto, esta técnica não possui uma exatidão satisfatória, comparativamente com outras técnicas já abordadas anteriormente, dado que a posição do objeto localizado, poderá ter um raio de erro elevado, consoante o raio de alcance de comunicação do dispositivo de referência. Além disso, em redes com bastantes pontos de acessos, os dispositivos nem sempre se ligam ao que se encontra mais próximo.

2.1.4 Comparação das técnicas de localização

Para uma mais fácil compreensão das vantagens e desvantagens das técnicas de localização apresentadas nesta secção, na Tabela 2.1 são enumerados os prós e os contras destas mesmas técnicas.

Técnica	Vantagens	Desvantagens
ToA e TDoA	- Boa escalabilidade; - Em condições de LoS possui uma elevada exatidão;	- *Hardware* complexo e caro; - Necessária elevada sincronização; - Muito sensível ao efeito *multipath*;
AoA	- Não necessita de sincronização; - Apenas requer 2 medições para localizações a 2D, ou 3 para 3D;	- Muito sensível ao efeito *multipath*; - *Hardware* complexo e caro;
Análise de Cenário	- *Hardware* simples; - Minimiza o efeito *multipath*; - Não necessita de sincronização;	- Necessária calibração do sistema (fase *offline*); - Requer algoritmos complexos; - Pouca escalabilidade;
Proximidade	- Simples implementação; - Baixo custo; - Boa escalabilidade;	- Exatidão relativamente baixa, em comparação com os outros métodos;

Tabela 2.1: Comparação entre técnicas de localização

2.2 Tecnologias e sistemas de localização

Depois de identificadas as técnicas e algoritmos de localização, nesta secção serão abordados alguns sistemas de localização específicos, baseados nas diversas tecnologias existentes e nas técnicas de localização abordadas na secção anterior.

Atualmente, existem diversos tipos de sistemas de localização baseados em tecnologias sem fios. Na Figura 2.9 estão representados os vários sistemas de localização sem fios, bem como a respetiva exatidão média e alcance, podendo este ser classificado como *indoor* e *outdoor*.

Nesta dissertação, vão ser abordados sistemas de localização *indoor*, mais propriamente sistemas baseados em GPS, RFID, redes móveis, UWB, WLAN e *Bluetooth*.

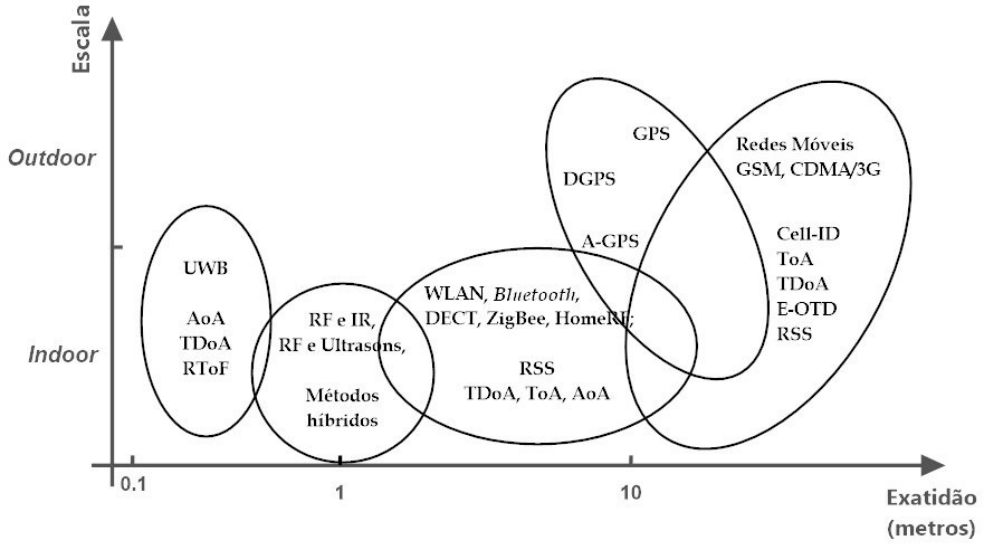


FIGURA 2.9: Sistemas de localização sem fios

2.2.1 Global Position System

O sistema de posicionamento global, também conhecido por GPS (*Global Position System*) é o sistema de posicionamento com maior sucesso em ambientes *outdoor* [11]. O GPS diferencial, ou DGPS (*Differential Global Position System*) [12], é uma evolução do GPS que provê uma melhoria significativa na exatidão e na precisão da localização. O DGPS utiliza um conjunto de antenas terrestres fixas que emitem as diferenças existentes entre as posições calculadas pelos satélites GPS e essas antenas fixas. Através desta correção, consegue-se melhorar os resultados ao nível da exatidão desde 15 metros, valor obtido no GPS tradicional, para cerca de 10cm, obtido nos melhores resultados com DGPS.

No entanto, devido à pouca, ou mesmo nula, cobertura dos sinais de satélite em ambientes *indoor*, a precisão e exatidão dos sistemas baseados em GPS diminui, de tal modo que tornam estes sistemas inoperacionais para utilização neste tipo de ambiente.

Com o intuito de contornar este problema e de fornecer uma solução viável baseada em GPS em ambientes *indoor*, foi desenvolvido um método denominado de GPS assistido, ou *Assisted*-GPS (A-GPS) [14], [13]. O conceito geral do funcionamento deste sistema encontra-se ilustrado na Figura 2.10. O A-GPS utiliza um servidor, que possui um recetor GPS tradicional, de forma a armazenar informações obtidas através dos satélites GPS. O servidor A-GPS encontra-se ligado, simultaneamente, às estações base que, por sua vez, comunicam com os dispositivos móveis em ambientes *indoor*, caso exista cobertura de rede. A comunicação entre um dispositivo móvel e as estações base

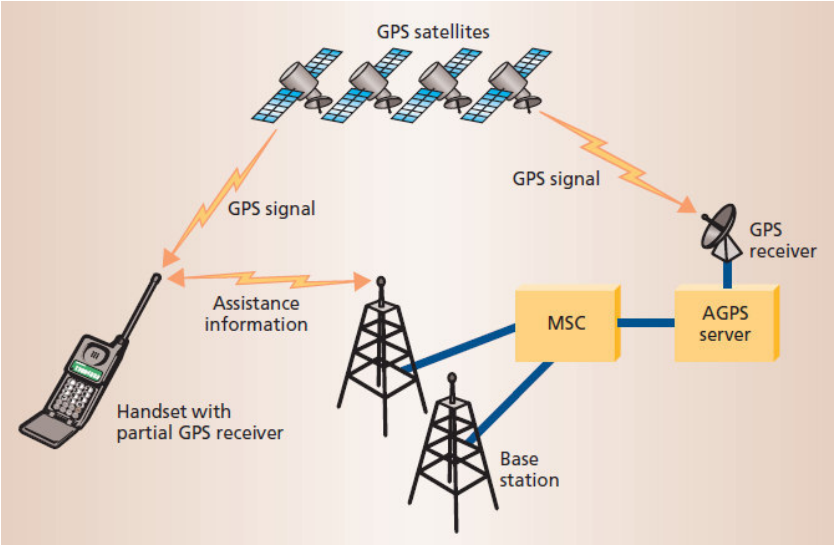


FIGURA 2.10: Funcionamento geral do sistema A-GPS [13]

é efetuada com base nas tecnologias de redes móveis, tais como GSM, CDMA, LTE. Durante esta comunicação são recolhidas informações relativas à propagação dos sinais que, posteriormente, são enviadas para o servidor A-GPS. Este, por sua vez, com base nessa informação recebida das estações base e, também, nos dados recebidos pelos satélites GPS, calcula a posição dos dispositivos móveis. Ou seja, neste sistema, o cálculo da posição é efetuado através da combinação das informações recolhidas dos satélites GPS e das estações base.

O sistema SnapTrack [15] foi o pioneiro na implementação da tecnologia baseada em A-GPS para o cálculo da localização em dispositivos móveis. Este sistema possui uma exatidão média entre os 5 e os 50 metros em ambientes *indoor*.

Mais recentemente, foram anunciados alguns sistemas de localização *indoor* baseados em GPS, como por exemplo, o Locata [16] e [17] que, também, funciona em ambientes *outdoor* e o *Indoor* GPS (iGPS) [18].

2.2.2 RFID

RFID é uma tecnologia que permite identificações automáticas à distância e, ao contrário do que acontece com os códigos de barras, não requer linha de visão [7], [19]. Um sistema RFID possui um conjunto de vários componentes, incluindo um número de leitores RFID, *tags* RFID e a comunicação entre eles. Os leitores RFID têm a capacidade de ler a informação emitida pelas *tags* RFID. Estas podem ser identificadas por uma infraestrutura, pois cada uma possui um identificador único (ID). Tanto os leitores como

as *tags*, para transmitir e receber informação, utilizam comunicações por radio frequência (RF).

As *tags* RFID são divididas em duas categorias: passivas e ativas. As *tags* passivas operam sem a utilização de uma bateria, são pequenas, mais baratas, e geralmente são usadas como substituição dos tradicionais códigos de barras. As *tags* RFID passivas apenas refletem os sinais RF que lhes são transmitidos, adicionando informação ao modular o sinal refletido. Possuem um alcance de apenas 1-2 metros.

Por outro lado, as *tags* RFID ativas são pequenos emissores/recetores que podem, de forma ativa, transmitir informação em resposta a um pedido. Além disso, este tipo de *tags* possui um maior alcance do que as *tags* RFID passivas [3].

Jeffrey Borriello e Roy Want propuseram o SpotON [20], um sistema de localização que utiliza a tecnologia RFID. O SpotON usa um conjunto de algoritmos para estimar localizações 3D, baseando-se na análise da potência dos sinais. No SpotON os objetos são localizados por nós de sensores homogéneos sem um controlo central, isto é, numa rede Ad Hoc. Os objetos a localizar são portadores de SpotON *tags*, com capacidade de calcular valores RSS, de forma a estimar a distância entre *tags*. A densidade de *tags* existente e a correlação com vários valores medidos ao longo do tempo, aumenta a precisão e exatidão do sistema.

Outro sistema para localizações *indoor* com base em *tags* ativas RFID é o LANDMARC [21]. Este sistema é constituído por leitores RFID que operam a uma frequência de 308 MHz, *tags* ativas RFID implementadas nos objetos a localizar, e um servidor que efetua a comunicação com os leitores RFID e realiza operações para o cálculo das localizações. Os leitores RFID têm a capacidade de ler a informação que é emitida pelas *tags* RFID.

De forma a aumentar a exatidão de localização deste sistema sem que sejam introduzidos mais leitores RFID, foram implementadas *tags* de referência, que funcionam como pontos de referência numa determinada posição fixa e que ajudam a calibrar o sistema, como se encontra ilustrado na Figura 2.11. As coordenadas das várias *tags* de referência são conhecidas pela infraestrutura. Quando se pretende localizar um determinado objeto (equipado com uma *tag* RFID), os leitores medem os valores RSS do objeto a localizar e, também, os valores RSS das *tags* de referência mais próximas do respetivo objeto. De seguida, essa informação é enviada para o servidor que, de forma a estimar a localização em que o objeto se encontra, aplica o algoritmo KNN. No entanto,

a exatidão da localização, além de depender do número de *tags* de referência existentes no sistema, irá, também, depender sempre do número de leitores RFID existentes.

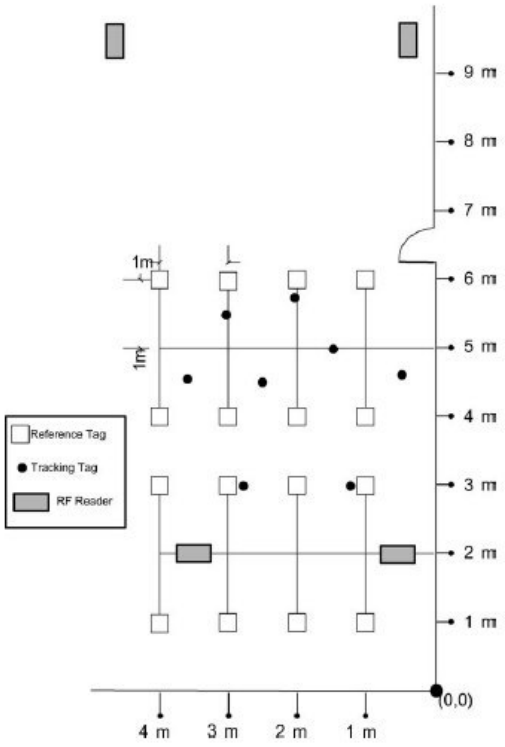


FIGURA 2.11: Disposição dos componentes do LANDMARC [21]

Após a realização de vários testes, onde foi colocado uma *tag* de referência por metro quadrado, os autores referem que foi obtido, em 50% dos casos, um erro de distância inferior a 1 metro, enquanto que o erro de distância máximo é cerca de 2 metros.

2.2.3 Redes móveis

Existem inúmeros sistemas usados nas tecnologias de redes móveis mais comuns (GSM e CDMA) para o cálculo da localização dos dispositivos móveis, que funcionam tanto em ambientes *outdoor*, como também, *indoor*. Geralmente, estes sistemas utilizam o método *Cell-ID* [22] ou o método E-OTD (*Enhanced Observed Time Difference*) [22] e [23]. No entanto, estes métodos possuem exatidões relativamente baixas (entre 50 a 200 metros), que podem variar consoante o tamanho da célula.

Em ambientes *indoor*, a localização baseada em redes móveis é possível se o local onde se encontra o dispositivo a localizar possuir cobertura de várias estações base, ou uma estação base cuja potência de sinal recebido pelo dispositivo seja elevada [3].

Veljo Otsason, Alex Varshavsky, Anthony LaMarca e Eyal de Lara propuseram um exato e preciso sistema para localização dentro de edifícios com vários pisos, baseado na tecnologia GSM [24]. O motivo deste sistema possuir uma exatidão considerável é este usar *fingerprints* com os valores de potência do sinal recebido de várias estações base GSM. Estas GSM *fingerprints*, além de incluírem valores das seis células com maior potência de sinal, tradicionalmente usadas na norma GSM, incluem, também, valores de células adicionais, cujos seus sinais são considerados fracos para serem usados em comunicações eficientes mas, no entanto, possuem potência de sinal suficientes para serem detetados. Tal como no caso da localização baseada em sinais Wi-Fi, os valores referidos em cima são recolhidos numa fase *offline*, com a diferença que neste caso pode ser formado um mapa *fingerprint* de uma grande área geográfica.

Os autores referem que após a realização de testes em três diferentes edifícios de vários pisos e aplicando o algoritmo WKNN, os resultados adquiridos mostram que o sistema de localização, por eles implementado, tem a capacidade de diferenciar, eficientemente, localizações entre os vários pisos. De uma forma geral, eles alcançaram uma exatidão média de 2,5 metros em localizações num único piso.

2.2.4 UWB

UWB é o acrónimo de *Ultra-Wide-Band*, e consiste em toda a tecnologia que utiliza comunicações sem fios com uma largura de banda superior a 500 MHz [3]. Tipicamente, o UWB baseia-se no envio de pulsos de pequena duração e com um *duty cicle* muito baixo.

Nos sistemas UWB, a forma de onda dos pulsos de pequena duração permite uma determinação precisa do tempo de chegada da transmissão de um pulso de um emissor para o recetor correspondente. A localização baseada em UWB explora as características do tempo de sincronização das comunicações UWB, em que é possível alcançar exatidões na ordem dos 20 cm [25]. Este valor é bastante satisfatório para soluções de alta exatidão de sistemas de localização em 2D e em 3D. No caso de um sistema de localização a 3 dimensões, utiliza-se duas medidas: primeiro, através da técnica TDoA, calcula-se a diferença do tempo de chegada de um pulso UWB a vários sensores recetores, e por fim é calculado o respetivo ângulo de chegada dos sinais pela técnica AoA.

A vantagem da utilização de ambas as técnicas de localização, em conjunto, é o aumento da exatidão do sistema. Além disso, a localização pode ser determinada usando

apenas dois sensores, diminuindo, assim, o requisito de uma densidade elevada de sensores existentes nos sistemas que utilizam apenas o TDoA ou o AoA como técnica de localização. A desvantagem é que este sistema é de implementação bastante difícil e requer que os dispositivos localizáveis estejam equipados com *hardware* bastante complexo e caro.

Atualmente, existem vários sistemas de localização precisos baseados em UWB. O Ubisense [26] é uma plataforma de localização unidirecional baseada em UWB, com controlo de acesso TDMA (*Time Division Multiple Access*).

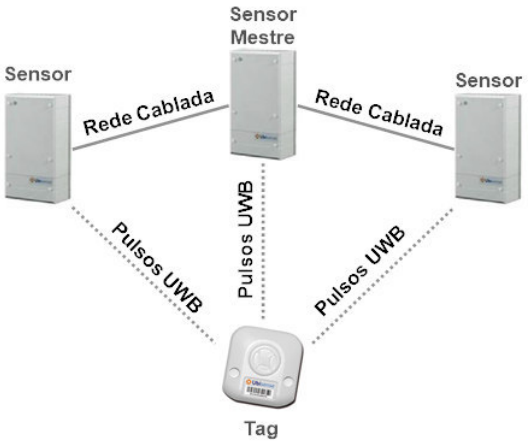


FIGURA 2.12: Componentes do Ubisense

O Ubisense é constituído por três componentes, *tags* ativas, sensores de referência e uma plataforma de *software* chamada *Ubisense Smart Space*, ilustrados na Figura 2.12. As *tags* enviam pulsos UWB para a rede de sensores existente, em resposta a pedidos feitos pelos sensores. Estes sensores estão organizados numa rede de células, cada uma delas composta por, pelo menos, quatro sensores, que dão cobertura a uma determinada área. Cada célula possui um sensor mestre, cuja sua função é coordenar o acesso ao meio das *tags* dentro da sua célula. Os vários sensores detetam o ângulo de chegada AoA dos sinais emitidos pelas *tags* e, caso estejam sincronizados entre si, é possível calcular o TDoA dos sinais, trazendo, deste modo, vantagens para o sistema, já acima referidas.

Toda a informação adquirida pelos sensores é enviada para a plataforma de *software* que, por sua vez, processa toda esta informação de modo a calcular a localização das *tags*, bem como apresentar as correspondentes localizações num modo gráfico. Segundo os autores, com a combinação das duas técnicas de localização, foi obtida uma exatidão de cerca de 15cm em 95% das experiências, proporcionando assim um elevado nível de exatidão, relativamente a sistemas baseados em outras tecnologias já abordadas.

2.2.5 WLAN

A norma IEEE 802.11 é uma norma utilizada nas redes WLAN, que opera com uma frequência de 2.4 GHz. O IEEE 802.11 é, hoje em dia, a norma dominante nas redes locais sem fios e, portanto, é muito apelativo para implementações de uma infraestrutura para localização *indoor* baseada neste tipo de redes. A exatidão de um sistema de localização numa WLAN baseada em medições RSS varia, aproximadamente, entre 3 a 30 metros, com uma taxa de atualizações na ordem dos poucos segundos [3].

Paramvir Bahl e Venkata N. Padmanabhan propuseram um sistema de localização e monitorização *indoor* de utilizadores, denominando-o de RADAR [27]. Este sistema é formado por um conjunto de APs, e a estimativa da localização é realizada através de uma análise do cenário. Inicialmente, na chamada fase *offline*, é feita a calibração no cenário onde se pretende implementar o sistema, criando assim um mapa com vários pontos de referência em determinadas posições, cada um com valores RSS correspondentes à sua posição. Na fase *online*, os valores RSS captados por um dispositivo a localizar são comparados com o mapa criado, de forma a estimar a localização do respetivo dispositivo.

Os autores propuseram dois tipos de abordagens para determinação da localização de um dispositivo, sendo eles um método empírico e outro um método determinístico. O primeiro utiliza os dados empíricos resultantes da fase *offline*, de forma a aplicar o algoritmo KNN para o cálculo da localização de um dispositivo.

No segundo método, são aplicados modelos de propagação dos sinais, que constituem uma alternativa à necessidade de dados empíricos para a aplicação do algoritmo KNN. Devido à obstrução que um sinal pode sofrer, os autores adaptaram um modelo que considera o efeito que obstáculos podem causar aos sinais eletromagnéticos entre o emissor e o recetor. Desde modo, eles propuseram o *Wall Attenuation Factor* (WAF), que considera o número de paredes (obstáculos) existentes. Através do WAF, é determinada a atenuação causada por uma parede, e este valor é usado em conjunto o com número total de paredes existentes entre o emissor e o recetor, de forma a compensar a atenuação existente. A exatidão do RADAR pode variar entre 2 e 3 metros.

Outro sistema conhecido, implementado no contexto da norma IEEE 802.11, é o Horus [28], proposto por Moustafa Youssef e Ashok Agrawala. A conceção do Horus visa satisfazer dois objetivos, primeiro ser um sistema de localização bastante preciso e, por último, possuir requisitos computacionais baixos. Este sistema identifica as diferentes

causas para as variações existentes nos sinais sem fios e tenta ultrapassa-las, de forma a alcançar uma elevada exatidão na localização. Para minimizar os requisitos computacionais, este sistema funciona com o uso da técnica de *clustering*. Para determinação da localização são aplicados métodos probabilísticos, que foram referidos anteriormente. Os autores referem que, durante as experiências efetuadas, foi obtida uma exatidão de 1,40 metros em 90% dos casos.

2.2.6 *Bluetooth*

A tecnologia *Bluetooth* opera na banda de frequência ISM de 2.4 GHz, e possui um alcance de até 100 metros. A localização baseada nesta tecnologia é apelativa, pois praticamente todos os dispositivos móveis, hoje em dia, estão equipados com *Bluetooth*.

O Topaz [3] é um sistema de posicionamento local, que estima a localização de *tags* *Bluetooth* e outros dispositivos equipados com esta tecnologia (telemóveis, PDAs, etc.). Possui uma exatidão média de 2 a 3 metros e é capaz de estimar a localização de vários dispositivos em simultâneo. Este sistema é constituído por três elementos: um ou mais servidores de localização, pontos de acesso e dispositivos *Bluetooth*. Segundo os autores, o Topaz possui uma exatidão de cerca de 2 metros em 95% das experiências, no entanto, está sujeito a um atraso de localização entre os 15 e os 30 segundos.

2.2.7 Comparação entre sistemas de localização

Nesta secção foi apresentada uma visão global sobre as tecnologias e sistemas de localização existentes, através de uma comparação entre os sistemas de localização referidos anteriormente, abordando as suas tecnologias, os algoritmos de localização, a exatidão e precisão e o custo.

Observando a Tabela 2.2, pode-se verificar que cada sistema de localização possui as suas particularidades bem claras e, na hora de escolher qual empregar, é importante ter algumas considerações. Deste modo, a escolha do melhor sistema de localização será sempre influenciada pelas características do cenário onde irá ser implementado, quais os tipos de dispositivos, bem como o custo e outras circunstâncias de cada caso.

Sistema de Localização	**Tecnologia**	**Algoritmos de Localização**	**Exatidão**	**Precisão**	**Custo**
GSM *Fingerprinting*	Redes móveis GSM (RSS)	WKNN	5m	10m em 80%	Médio
Horus	WLAN RSS	Métodos probabilísticos	2m	1,40m em 90%	Baixo
LANDMARC	*tags* RFID ativas	KNN	<2m	1m em 50%	Baixo
RADAR	WLAN RSS	KNN	2 a 3m	2,50m em 50% e 5,90m em 90%	Baixo
SnapTrack	A-GPS e TDoA	n.a.	5m a 50m	25m em 50%	Médio
SpotON	*tags* RFID ativas	lateração *Ad-Hoc*	Depende do tamanho do *cluster*	n.a.	Baixo
Topaz	*Bluetooth* RSS	n.a.	2m	2m em 95%	Médio
Ubisense	unidirecional UWB, TDoA + AoA	Mínimo desvio quadrado	15cm	0,30m em 99%	Médio alto

Tabela 2.2: Comparação entre sistemas de localização

Capítulo 3

Localização com informação histórica e *feedback* dos utilizadores

A localização, baseada em informação histórica ou *feedback* dos utilizadores, tem como objetivo melhorar a precisão e exatidão do cálculo da localização nos sistemas de localização. Neste capítulo vão ser apresentados alguns sistemas concluídos e trabalhos elaborados, no âmbito da localização em ambientes *indoor*, e cujos algoritmos de localização utilizam informação histórica ou *feedback* dos utilizadores no processo de cálculo das posições dos dispositivos localizáveis. Primeiramente, vai ser abordado o sistema de localização baseado em RSS *Fingerprinting* [6], apresentado por Khodayari, Maleki e Hamedi. Mais à frente, será descrito o sistema proposto por Zhen *et al.*, baseado na incorporação de informação geométrica nos algoritmos de localização [29]. Por último, será explicado um algoritmo que aplica o *feedback* atribuído pelos utilizadores do sistema [30].

3.1 RSS *Fingerprinting* baseado em dados históricos

Neste trabalho, Khodayari, Maleki e Hamedi propõem um sistema de localização baseado em RSS *Fringerprinting*, utilizando informação histórica [6]. Nesta secção será descrito, pormenorizadamente, todo este sistema.

Um dos algoritmos de localização determinísticos mais utilizados em sistemas *Fingerprinting* é o *K Nearest Neighbors* (KNN). No entanto, como o KNN apenas utiliza os

K vizinhos mais próximos, em alguns casos pode não ser obtida uma precisão e exatidão satisfatória, devido ao facto de os sinais eletromagnéticos estarem sujeitos a degradações, tais como o efeito *multipath*, atenuações, ruídos, interferências e perdas em espaço livre.

Com o objetivo de melhorar a precisão e exatidão dos sistemas de localização baseados em *RSS Fingerprinting*, os autores propuseram um novo algoritmo denominado *Predicted K Nearest Neighbors* (PKNN). Este novo algoritmo é uma variação do KNN, e determina a localização de um utilizador em movimento, não só utilizando os K vizinhos mais próximos, como também utilizando a velocidade e posicionamentos anteriores dos respetivos utilizadores a localizar.

Para o desenvolvimento e testes deste sistema, foi criado dentro de um edifício uma infraestrutura WLAN coberta com quatro *Access Points* (APs). Os utilizadores do sistema, ou seja, os utilizadores que pretendem ser localizados dentro do edifício, estão equipados com *tag's* RFID que possuem a capacidade de calcular e enviar os valores da potência do sinal recebido (RSS) de cada AP para o servidor de localização. O intervalo de cálculo e envio dos valores de RSS por cada *tag* foi definido como sendo de 1 segundo.

3.1.1 Arquitetura proposta

Como referido anteriormente, este sistema é baseado no método *Fingerprinting*. Tal como todos os sistemas baseados neste método, este também se encontra dividido em duas fases, sendo elas a fase *online* e fase de *offline* (fase de treino). Na fase de treino é construído um mapa com as medições da potência do sinal recebido de cada AP, em posições pré-definidas ao longo de uma determinada área. Estas medições são denominadas de RPs, e possuem o seguinte formato: $\{RSSV_i \quad RP_ix \quad RP_iy\}$. Onde RP_ix e RP_iy representam as coordenadas da localização do RP_i, em que $i = (1, 2, \ldots, R)$ corresponde ao número dos R RPs existentes. Já $RSSV_i$ é o vetor de RSSs $\{RSS_{i1}, RSS_{i2} \ldots RSS_{ij}, \ldots, RSS_{in}\}$, ou seja, os valores da fase *offline* referente ao RP_i, em que RSS_{ij} representa a média dos valores de RSSs observados no RP_i dos AP_j, onde $j = (1, 2, \ldots, n)$ é o número de n APs existentes.

Na fase *online*, para que um utilizador seja localizado numa dada posição L, o sistema mede os RSSs de todos os APs disponíveis em L. Depois, todos estes valores medidos são comparados com os valores guardados na base de dados da fase *offline* e, em conformidade com um algoritmo de localização, é calculado o posicionamento do utilizador.

O algoritmo que os autores usaram inicialmente foi o WKNN, uma variante do algoritmo KNN, que atribui "pesos" aos K vizinhos escolhidos conforme a distância a que eles se encontram do dispositivo a localizar. No entanto, devido à distância existente entre um dispositivo num determinado local e todos os APs existentes, alguns APs podem não estar no alcance desse mesmo dispositivo. Sendo assim, alguns vetores RSS podem não incluir valores da potência do sinal recebido de todos os APs existentes, o que faz com que alguns RPs vizinhos possam ter vetores RSS semelhantes. No algoritmo KNN original, todos os RPs existentes no mapa estão suscetíveis a serem escolhidos como vizinhos mais próximos sem se considerar este facto. Por outro lado, os vizinhos encontrados pelo algoritmo KNN podem até estar dispersos pelo mapa, ou seja, na realidade não serem vizinhos. Isto deve-se ao facto de a atenuação dos sinais de cada AP não depender unicamente da distância, mas também de outros fatores que já foram referidos anteriormente.

Deste modo, os autores propuseram um novo algoritmo de localização denominado de PKNN, que complementa o algoritmo WKNN e tem em consideração os problemas mencionados no parágrafo anterior. Para tal, são usadas informações históricas relativas ao passado dos dispositivos a localizar. No ponto seguinte é explicado o funcionamento deste novo algoritmo.

3.1.1.1 *Predicted K Nearest Neighbors*

Em primeiro lugar, o algoritmo PKNN introduz uma técnica que visa melhorar o desempenho relativo aos algoritmos KNN ou WKNN. Este encontra os vizinhos mais próximos filtrando os RPs que não têm vectores RSS semelhantes, reduzindo assim o tempo e a complexidade computacional. Na prática, significa que nos dispositivos a localizar, se a potência do sinal recebido (RSS) de um AP não possuir o valor NULL (o AP está no seu alcance), então o valor do sinal recebido desse mesmo AP nos vetores RSS do RPs selecionados, também não podem ter o valor NULL. Usando este filtro, é criado um subconjunto de RPs, a que os autores chamam de *Filter-RPset*, que possuem uma maior probabilidade de serem selecionados como vizinhos mais próximos.

Depois disto, a localização de um dispositivo é calculada com base no seu vetor RSS e no *Filter-RPset*, através da execução de algumas iterações que serão descritas de seguida.

Inicialmente, a localização de um dispositivo é calculada utilizando-se o algoritmo WKNN. Em segundo lugar, devido ao facto de as *tag's* RFID deste sistema atualizarem periodicamente (a cada 1 segundo) os valores da potência do sinal recebido de cada AP, é possível prever a sua localização, baseando-se no seu comportamento prévio, isto é, nas suas respetivas localizações anteriores.

Então, se a distância entre a posição de um dispositivo, calculada pelo algoritmo WKNN, e a sua posição anteriormente calculada for superior ao deslocamento máximo permitido (*max_displacement*), pode-se concluir que a posição calculada não é de confiança, pois a distância percorrida pelo utilizador, calculada pelo algoritmo, é elevada demais para um curto espaço de tempo. Portanto, para estes casos, usando um histórico de localizações dos dispositivos, é possível prever uma localização possível para estes. Assim sendo, são feitos os seguintes cálculos: assume-se que (x_{pre1}, y_{pre2}) e (x_{pre2}, y_{pre2}) são as coordenadas das duas localizações anteriores de um dispositivo. A próxima posição desse mesmo dispositivo (x_{next}, y_{next}) é calculada por:

$$\begin{cases} x_{next} = x_{pre2} \pm \Delta X \\ y_{next} = y_{pre2} \pm \Delta Y \end{cases} \tag{3.1}$$

Onde ΔX e ΔY representam os deslocamentos possíveis nas direções x e y, respetivamente. Estes valores são calculados por:

$$\begin{cases} \Delta X = D \cos\left(\tan^{-1} m\right) \\ \Delta Y = D \sin\left(\tan^{-1} m\right) \end{cases} \tag{3.2}$$

Onde, neste caso m representa a taxa de variação do movimento, considerando as duas últimas posições de um dispositivo. D é o deslocamento de um dispositivo em coordenadas cartesianas. São calculados, respetivamente, pela Equação 3.3 e pela Equação 3.4.

$$m = \frac{y_{pre2} - y_{pre1}}{x_{pre2} - x_{pre1}} \tag{3.3}$$

$$D = Vt \tag{3.4}$$

Onde V representa a velocidade atual de um dispositivo. Para o cálculo deste valor, os autores definem V_{human} como sendo o valor da velocidade média de um humano (cerca de 1.5 m/s), e têm em consideração a velocidade anterior desse mesmo dispositivo nas suas duas últimas posições, como é demonstrado de seguida.

$$V = \frac{(V_{human} + V_{pre})}{2} \tag{3.5}$$

onde,

$$V_{pre} = \sqrt{\frac{(x_{pre2} - x_{pre1})^2 + (y_{pre2} - y_{pre1})^2}{t}} \tag{3.6}$$

Como foi referido, o intervalo de tempo definido para cálculo das posições dos dispositivos é de 1 segundo, logo o valor t na Equação 3.4 e na Equação 3.6 é assumido com o valor 1.

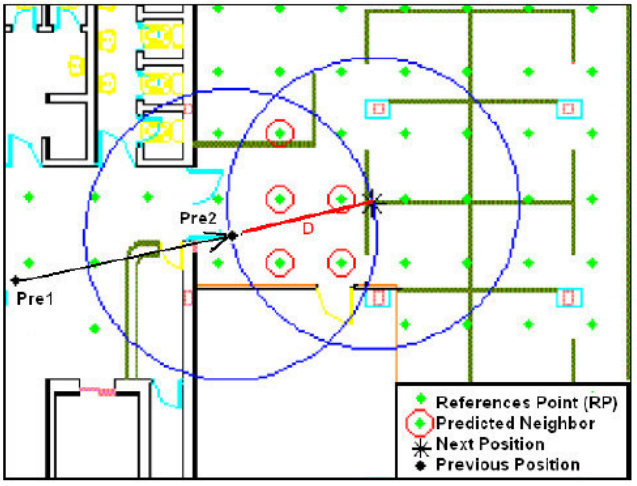


FIGURA 3.1: Cálculo dos *Predict-Neighbors* [6]

Através de várias observações realizadas pelos autores, eles concluíram que a zona onde se obtém maior precisão e exatidão na localização de um dispositivo é algures entre a posição anterior (x_{pre2}, y_{pre2}) e a próxima posição calculada (x_{next}, y_{next}). Então, como se pode verificar pela Figura 3.1, são "desenhadas" duas circunferências de raio D centradas na posição anterior e na próxima posição calculada. Os RPs que se encontrarem no espaço de interceção entre estas duas circunferência são chamados de *Predict-Neighbors*. De seguida, é calculada a média das posições dos *Predicted-Neighbors* através da equação (2.2), obtendo-se, assim, a denominada posição prevista.

Por último, é feita a média aritmética entre a posição prevista calculada anteriormente e a posição dada pelo algoritmo WKNN. Finalmente, é enviada em coordenadas cartesianas a localização calculada através do algoritmo PKNN.

Caso não haja nenhuma informação acerca das posições anteriores de um dispositivo, a sua posição prevista não pode ser calculada e a posição dada pelo algoritmo WKNN

é assumida como sendo sua posição. Um resumo de todos os passos realizados pelo sistema de localização proposto pelos autores é apresentado de seguida.

1. Filtrar os RPs de acordo com os seus vetores RSS, formando o *Filter-RPset*.
2. Encontrar os K vizinhos mais próximos do *Filter-RPset*, através da Equação 2.1.
3. Calcular da posição de uma *tag* usando o algoritmo WKNN, pela Equação 2.2.
4. Se a distância entre a posição obtida e a posição anterior for maior do que o deslocamento máximo (*max_displacement*):
   (a) Calcular a velocidade atual e a anterior através da Equação 3.5 e da Equação 3.6.
   (b) Calcular a próxima posição, em conformidade com o deslocamento possível, a direção e a velocidade da *tag*, através da Equação 3.1.
   (c) Calcular os *Predict-Neighbors*.
   (d) Calcular a posição prevista de uma *tag*, realizando a média dos *Predict-Neighbors* encontrados, através da Equação 2.2.
   (e) Calcular a média aritmética entre a posição obtida pelo algoritmo WKNN e a posição prevista. O resultado obtido é a posição atual da *tag*.

3.1.2 Testes e resultados

Depois de o sistema estar operacional, os autores realizaram diversos testes e compararam os resultados obtidos, com e sem as modificações que eles implementaram. Em todos os testes, o número de vizinhos mais próximos do algoritmo WKNN foi definido como sendo 20 (K=20), e o valor do deslocamento máximo (*max_displacement*) é de 3 metros. Isto significa que o algoritmo de previsão PKNN só é utilizado pelo sistema caso a distância entre a posição calculada pelo algoritmo WKNN e a posição anterior de uma *tag* for superior a 3 metros.

O primeiro teste apresentado consiste em avaliar a exatidão da localização, com o uso do algoritmo WKNN (sem o módulo de previsão) e com o uso do algoritmo PKNN (com o módulo de previsão). Os resultados estão representados na Tabela 3.1, onde são apresentados os valores da distância do erro médio estimado e as distâncias máximas e mínimas dos erros obtidos. As distâncias são calculadas através da distância euclidiana e estão expressas em metros. Pela comparação dos resultados, conclui-se que

no algoritmo PKNN o erro médio obtido é menor e, além disso, os valores dos erros de distância máxima e mínima são bem mais satisfatórios em comparação com o algoritmo WKNN.

Algoritmo	**Erro Médio**	**Erro Máximo**	**Erro Mínimo**
WKNN	3.95	14.42	0.39
PKNN	2.65	7.28	0.33

Tabela 3.1: Resultados obtidos pelos algoritmos WKNN e PKNN

Outro teste realizado permite analisar o efeito do filtro *RPset* nos algoritmos WKNN e PKNN. Utilizando-se este filtro, apenas os RPs que possuem vectores RSSs similares são selecionados, de forma a melhorar o desempenho dos algoritmos, quer a nível de exatidão, como de capacidade computacional. No algoritmo WKNN, os autores obtiveram um erro médio de 3,05m usando o filtro *RPset*, e de 3,95m sem o filtro. Para o algoritmo PKNN, obteve-se um erro médio de 2,65 com filtro e de 3,35 sem filtro. Como consequência destes resultados, conclui-se que o desempenho de ambos os algoritmos é melhorado quando se utiliza o método do filtro *RPset*. Além disso, os autores referem que o tempo e a capacidade computacional são inferiores quando se utiliza este método.

3.2 Incorporação de informação geométrica

A incorporação de informação geométrica nos algoritmos de localização tem como principal objetivo melhorar a precisão e exatidão da localização. Zhen *et al.* propuseram um sistema para controlo de iluminação dentro de edifícios [29], em que fazem uso de um algoritmo de localização que, por sua vez, utiliza informação histórica, de forma a melhorar a exatidão do algoritmo. Para tal, os autores utilizam informação geométrica do cenário no qual pretendem aplicar o seu sistema, que consiste num piso amplo de um edifício. Este edifício, que serviu como ambiente de teste, possui quatro divisões separadas por paredes de vidro, para permitirem que os sinais RF ultrapassem as paredes sofrendo atenuações mínimas. O sistema é baseado na tecnologia RFID, e é construído por sensores e *tags* RFID, como em outros sistemas já abordados. Cada uma das quatro divisões existentes encontra-se dividida em três regiões, formando-se, assim, doze regiões no total, como ilustra a Figura 3.2.

O cálculo da localização das *tags* RFID encontra-se dividida em dois passos. Primeiro, de forma a localizar cada *tag* dentro de uma das doze regiões, é utilizado um

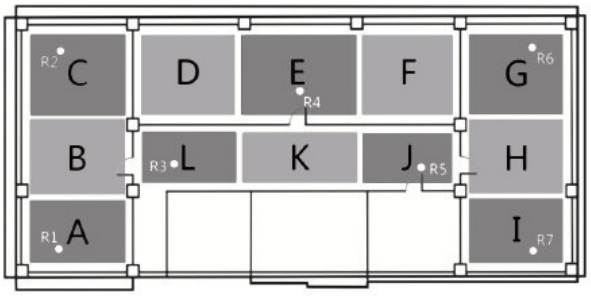


FIGURA 3.2: Ambiente de teste do sistema

algoritmo de localização baseado em RSS-Fingerprinting. Detalhes sobre esta técnica de localização podem ser consultados na secção 2.1.2. Posteriormente, é usada a informação histórica e relações geométricas, com o objetivo de contornar erros de comunicação que possam surgir, valores de RSS lidos incorretamente, e de melhorar a exatidão da localização estimada pelo algoritmo de localização. Todo este processo é explicado de seguida.

Considere-se o caso em que uma pessoa equipada com uma *tag* RFID se encontra na região C e que esteja de pé, encostada à parede que divide esta região com a região D, e outra pessoa que se encontre na região D encostada à parede que separa esta região com a região C. Nesta situação, os valores dos dois vetores RSS medidos pelos sensores RFID são muito idênticos, e o resultado do cálculo das suas localizações pode variar entre a região C e D. Este facto não é viável, uma vez que uma pessoa não consegue mover-se rapidamente entre estas duas regiões, num período de atualização da localização.

A solução encontrada pelos autores foi restringir o resultado da localização dentro das regiões vizinhas. Nas experiências realizadas, as atualizações de localização são efetuadas uma vez por segundo. Calculou-se, que em condições normais, os ocupantes do sistema não ultrapassam os 5 m/s. A área de cada uma das regiões varia entre 5 a 6 m^2. Deste modo, conclui-se que a solução encontrada pelos autores é viável apenas em condições normais. No entanto, em condições de emergência, ou em outras condições, os ocupantes a localizar podem mover-se mais rapidamente que o normal, de uma dada região para outra que não seja vizinha. Quando estas situações ocorrem, devido às restrições impostas, um ocupante é localizado na região vizinha da sua última localização que, na maioria dos casos, também é região vizinha da região onde ele realmente se encontra. Então, na próxima atualização de localização, o ocupante é localizado na sua região onde ele realmente está, que por sua vez é uma região vizinha da localização anterior.

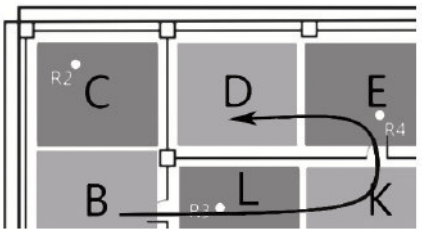


FIGURA 3.3: Exemplo de uma situação em que a localização pode ficar "presa" [29]

No entanto, se um ocupante mantém um movimento muito rápido por um longo período de tempo, a sua localização pode ficar "presa" numa determinada região. Na Figura 3.3 está ilustrado um exemplo desta situação. Suponha-se que um utilizador move-se rapidamente da região B para a região D, atravessando L, K e E. Até ao momento em que ele pára, em D, o cálculo da sua localização pode ficar "preso" na região L. Uma vez que D não é vizinho de L, o algoritmo não vai localizar o ocupante na região D. L apenas tem como vizinhos a região B e K, embora a região D esteja próxima (do outro lado da parede).

Por forma a resolver este problema, o sistema verifica, depois de localizar o utilizador, em que região ele se encontrava à n segundos atrás, e quais as regiões vizinhas dessa região. Neste caso, depois de localizar o utilizador na região D, se se verificar que o utilizador esteve na região E à n segundos atrás, e que esta possui K como região vizinha, é feito o "shif" da localização para a região correta, ou seja, da região L, onde a localização estava "presa", para a região D.

O valor de n é um *tradeoff* entre um sistema robusto e um sistema sensível. Um valor pequeno de n tem como resultado um algoritmo mais sensível, tornando possível localizar corretamente um ocupante, mesmo que ele se mova rapidamente. Por outro lado, resulta num sistema menos robusto, em que pequenas alterações do meio ambiente (ex: abrir/fechar portas, janelas, outros ocupantes) afetem os resultados da localização. O contrário, resulta num sistema mais robusto mas menos sensível. O valor de n é, normalmente, calculado por experiências.

3.3 Localização com informação histórica versus não histórica

Como se pode verificar pelos resultados obtidos em trabalhos anteriores, a utilização de informação, baseada em dados históricos nos algoritmos de localização, traz

benefícios à performance dos sistemas de localização que implementam este tipo de algoritmos. Desta forma, é possível que se consiga um melhoramento da exatidão, precisão e, também, do nível de estabilidade dos cálculos das localizações. Como retratado nos trabalhos apresentados nesta secção, com informação histórica é possível restringir o leque de posições possíveis para os dispositivos a localizar, com base na sua velocidade de deslocamento e nas suas posições do passado. Além disso, é possível criar a impossibilidade de um dispositivo "atravessar" certos obstáculos, como as paredes.

Por outro lado, os algoritmos de localização baseados em informação histórica provocam um atraso no tempo necessário para o cálculo de uma nova posição, de um dispositivo em movimento, comparativamente com os algoritmos baseados em dados não históricos. Isto deve-se ao facto de os algoritmos baseados em informação histórica serem mais complexos e de necessitarem de maiores recursos computacionais, pois para além de utilizarem a mesma informação que os algoritmos não históricos, ainda usam informações adicionais, como a velocidade de deslocamento de um dispositivo e os dados sobre as suas respetivas posições no passado.

3.4 Aplicação de *feedback* do utilizador

Uma possível forma de corrigir e prevenir os mesmos erros no futuro, por parte dos sistemas de localização, é usar um procedimento *online* com base no *feedback* dos utilizadores. Bhasker, Brown e Griswold apresentaram um algoritmo de localização [30], em que os utilizadores do sistema podem "corrigir" possíveis erros de exatidão, de forma a evitá-los nas próximas utilizações. Este algoritmo foi aplicado num sistema RSS *fingerprinting* numa rede sem fios 802.11b.

Inicialmente, a posição de um utilizador é calculada por um algoritmo básico e, quando este apresenta um resultado insatisfatório, o utilizador está autorizado a corrigir a posição que lhe é apresentada. Para tal, este clica no mapa na posição onde realmente se encontra. Depois desta ação, na posição onde o utilizador clicou é criado um ponto de acesso virtual (VAP - *Virtual Access Point*), usado para cálculos futuros.

Um VAP guarda dois tipos de informação: a posição correta do utilizador e os valores de RSS, observados naquela posição pelo dispositivo de rede que o utilizador possui. Esta informação, em conjunto, é denominada pelos autores de correção (*correction*) e é armazenada na base de dados, juntamente com o ID do utilizador de quem criou o VAP, o instante de tempo e o modelo do dispositivo que recolheu os dados.

Quando um utilizador se desloca para uma área onde já existem correções, o algoritmo, que aplica o *feedback* dos utilizadores, vai utilizar essas correções.

Em qualquer sistema que utiliza *feedback* dos utilizadores, surgem sempre algumas questões sobre a instabilidade do sistema, que poderá ser causado pelos utilizadores ao darem *feedback* errado. De forma a contornar este problema, os autores definiram as seguintes políticas, baseadas na similaridade dos valores RSS observados.

1. Se uma nova correção dada por um utilizador é similar a correções já existentes adicionadas por outros utilizadores, essas correções já existentes são definidas como privadas. Correções privadas apenas são visíveis por quem as adicionou. Assim sendo, a nova correção fica disponível para todos os utilizadores, enquanto que as correções existentes ficam apenas para quem as adicionou. Através desta política, o comportamento malicioso dos utilizadores é abordado de forma relativamente simples. Basta apenas um utilizador corrigir o *feedback* malicioso dado por outro utilizador para escondê-lo. No entanto, o algoritmo de localização irá usar esse *feedback* errado no cálculo das posições do utilizador que o deu.

2. Se uma nova correção é similar a correções adicionadas pelo mesmo utilizador, as correções existentes são apagadas. Esta política permite aos utilizadores remediar uma correção errada ou obsoleta (ex: a mobília é mudada de sítio).

3. Se uma correção é adicionada, esta é marcada como pública.

3.5 Localização com *feedback* dos utilizadores versus sem *feedback* dos utilizadores

A utilização do *feedback* dos utilizadores nos algoritmos de localização pode trazer várias vantagens. Nada melhor do que os utilizadores para avaliar os resultados calculados pelos algoritmos, quer seja uma avaliação positiva ou negativa. Com a introdução dessas avaliações na base de dados utilizada nos cálculos de posicionamento, consegue-se realimentar o sistema, com o objetivo de melhorar os futuros, ou mesmo atuais, cálculos de localização.

Nos algoritmos que não têm em consideração o *feedback* dos utilizadores, quando existe alguma alteração no cenário em que o sistema está implementado, como por exemplo, mudança de mobiliário, é necessário fazer-se uma nova calibração do sistema

para que os valores de RSS nos RPs não se tornem obsoletos. Todavia, uma nova recolha de amostras *offline* é sempre uma tarefa a evitar, dado que se trata de um procedimento bastante trabalhoso e demorado. Com a utilização do *feedback* dos utilizadores, em certas situações em que ocorra uma alteração no cenário, é possível que o sistema se adapte ao longo do tempo e de forma dinâmica, evitando-se assim os problemas referidos neste parágrafo.

Por outro lado, *feedback* fraudulento pode prejudicar bastante a exatidão e precisão dos algoritmos de localização, uma vez que estes podem ser enganados por informação que não está correta. De forma a evitar esta situação, é fundamental definir medidas ou políticas de prevenção contra estas situações.

Capítulo 4

Algoritmos de localização propostos

Um dos aspetos fundamentais de um sistema de localização é providenciar uma estimativa da posição com elevado grau de certeza, isto é, quanto maior for a exatidão da localização, mais robusto é o sistema. Nos sistemas baseados em *fingerprinting*, o grau de exatidão e precisão depende, em grande parte, dos algoritmos de localização. A correta leitura dos valores de RSS efetuada pelo *hardware* e a interferência nos sinais eletromagnéticos possuem, também, impacto na exatidão e precisão da localização. No entanto, através do algoritmo é possível atenuar, ou mesmo eliminar essas influências. Deste modo, os algoritmos assumem um papel crucial para se garantir uma elevada exatidão e precisão num sistema de localização. Além disso, é fundamental que os algoritmos executem, de modo minucioso, o processamento de todos os dados relativos à informação de localização, de forma a serem efetuados corretamente os cálculos de posicionamento dos dispositivos localizáveis. Na Figura 4.1 está ilustrado o papel dos algoritmos de localização num sistema baseado no método *fingerprinting*.

Neste trabalho, foram propostos cinco diferentes algoritmos para melhorar os algoritmos mais comuns e os propostos num projeto anterior [1], sendo que três deles fazem uso da informação histórica dos utilizadores, e outros dois usam o *feedback* dos utilizadores. Além disso, foi também desenvolvido um algoritmo bastante básico, denominado soma das distâncias, que foi usado como ponto de partida e os seus resultados foram comparados com os restantes algoritmos. Neste capítulo é apresentado todo o trabalho realizado na conceção dos algoritmos de localização propostos, mais propriamente, soluções propostas para melhorar os algoritmos já existentes, a sequência de processos

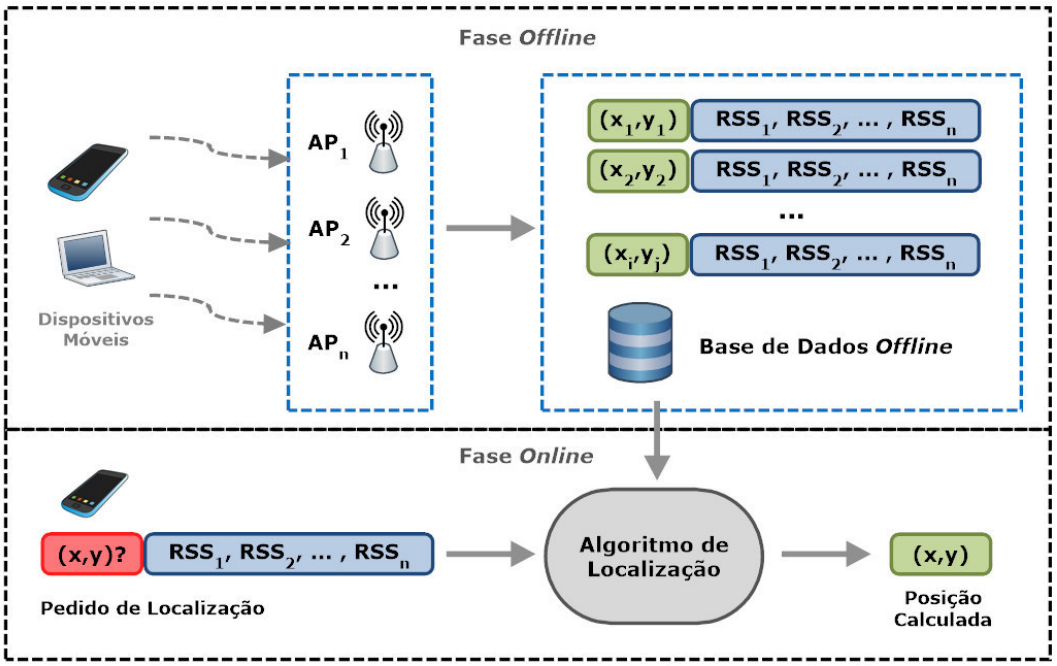


FIGURA 4.1: Cálculo da localização dos dispositivos móveis na técnica RSS *fingerprinting*

até chegar ao resultado final, como são efetuados os cálculos de posicionamento e os diagramas de atividades. De seguida, são apresentados, detalhadamente, os métodos de funcionamento de todos os algoritmos propostos.

4.1 Soma das distâncias

O primeiro algoritmo de localização implementado foi o algoritmo denominado de soma das distâncias. É o menos complexo dos algoritmos implementados, e baseia-se na distância entre dois pontos. Neste caso, pretende-se calcular a distância a que o dispositivo a localizar se encontra em relação a cada ponto de referência do cenário. Este cálculo é feito através dos valores de RSS medidos pelos dispositivos móveis num determinado período de tempo, e os valores de RSS presentes na base de dados *offline*. Deste modo, a distância D_i a que se encontra cada ponto de referência é calculada pela Equação 4.1:

$$D_i = \sum_{j=1}^{N} |RSS\_offline_{APj} - RSS\_online_{APj}| \tag{4.1}$$

Onde $RSS\_offline_{APj}$ representa os valores de RSS observados do AP_j na fase *offline* e RSS_online_{APj} representa os mesmos valores, no entanto, durante a fase *online*.

Este cálculo é feito para cada RP, num total de R RPs, em que cada um destes pontos de referência possuem registos dos valores de RSS observados nos N APs do cenário.

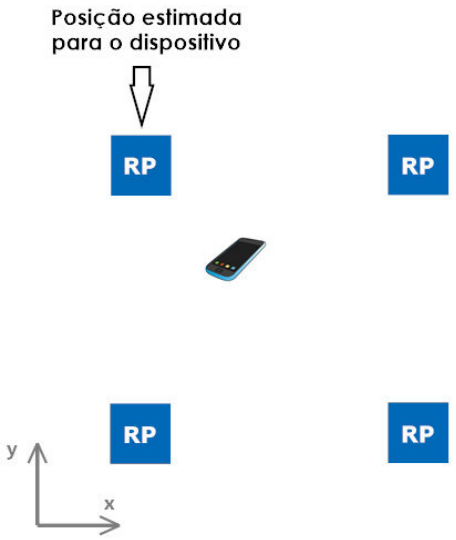


FIGURA 4.2: Exemplo de posição estimada para um dispositivo no algoritmo soma das distâncias

As coordenadas do ponto de referência que possuir a menor distância calculada, isto é, o ponto de referência mais próximo, são apresentadas como sendo as coordenadas da posição do dispositivo a localizar. Assim sendo, a posição do dispositivo a localizar é estimada através da Equação 4.2.

$$(x, y) = \min \left\{ \sum_{i=1}^{R} D_i(x_i, y_i) \right\} \tag{4.2}$$

Aqui reside uma desvantagem deste algoritmo, uma vez que ele apenas pode estimar posições para os dispositivos a localizar, as mesmas posições dos pontos de referência existentes, como exemplificado na Figura 4.2. Caso o dispositivo se encontre desviado do ponto de referência mais próximo, a sua posição calculada pelo algoritmo irá ser sempre as coordenadas desse ponto de referência.

Na Figura 4.3 encontra-se ilustrado o diagrama de atividades do algoritmo de localização soma das distâncias.

FIGURA 4.3: Diagrama de atividades do algoritmo soma das distâncias

4.2 Algoritmos que utilizam informação histórica

De seguida, irá ser apresentado, detalhadamente, o trabalho desenvolvido no âmbito da utilização da informação histórica nos algoritmos de localização, com o intuito de melhorar os resultados. Serão apresentados os diferentes métodos utilizados para sumariar a informação histórica, bem como a sua utilização no cálculo da localização dos dispositivos.

4.2.1 *Historical K Nearest Neighbor*

O primeiro algoritmo desenvolvido que utiliza a informação histórica dos utilizadores foi o *Historical* K *Nearest Neighbor* (HKNN). Este algoritmo tem como objetivo aumentar a exatidão e precisão do cálculo das localizações, através da tentativa de minimizar, ou mesmo corrigir, certos erros que podem ocorrer, como por exemplo, uma leitura errada dos valores de RSS.

Nos algoritmos soma das distâncias, KNN ou WKNN as localizações dos dispositivos são calculadas sem ter em conta as respetivas posições anteriores. Este algoritmo

foi proposto para considerar este facto. De uma forma geral, este algoritmo utiliza informação do passado recente dos utilizadores, de forma a evitar grandes deslocações em curtos espaços de tempo.

No entanto, este algoritmo tem como ponto de partida o resultado de localização dado por um outro algoritmo, sem dados históricos e baseado no conceito de *Nearest Neighbors*. Neste caso, vai ser usado como exemplo o algoritmo WKNN.

Em primeiro lugar, é definido um deslocamento máximo permitido durante um certo intervalo de tempo para os dispositivos a localizar. Quando o deslocamento máximo permitido entre a posição inicialmente calculada pelo algoritmo WKNN e a posição do mesmo dispositivo no instante de tempo anterior (última posição conhecida) é ultrapassado, o algoritmo calcula uma possível próxima posição para dispositivo, consoante o movimento que ele realizou nas suas duas posições anteriores. Esta posição calculada é denominada de Próxima Posição e, de seguida, com a última posição conhecida (calculada pelo WKNN) e com a Próxima Posição, são escolhidos os Possíveis Vizinhos, que consistem nos RPs que se encontram na interceção de duas circunferências traçadas com origem nestas duas posições, como exemplificado na Figura 4.4. Seguidamente, é feita uma média entre os Possíveis Vizinhos, de forma a obtermos uma só posição em coordenadas cartesianas, denominada de Posição Prevista. Por último, a posição devolvida como sendo a posição do dispositivo a localizar é a média aritmética das coordenadas da Posição Prevista e da posição anteriormente calculada pelo algoritmo sem dados históricos (WKNN).

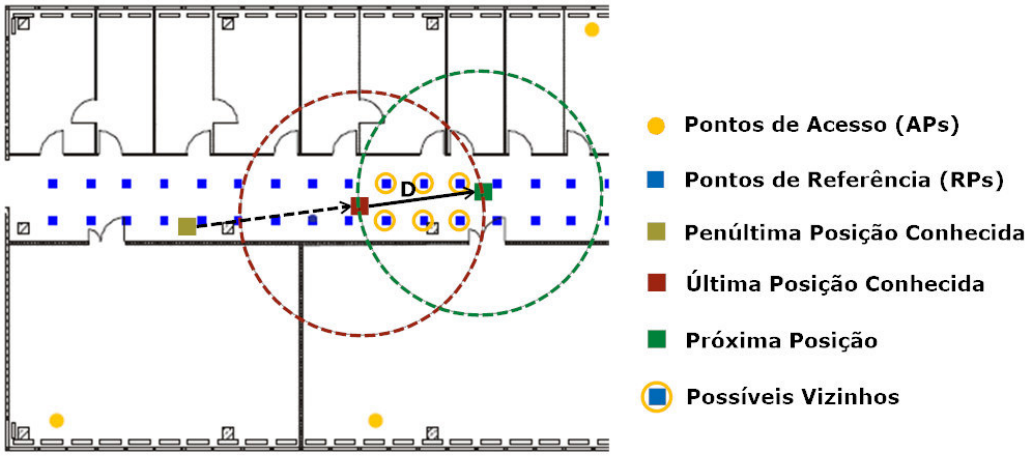


FIGURA 4.4: Escolha dos Possíveis Vizinhos no algoritmo HKNN

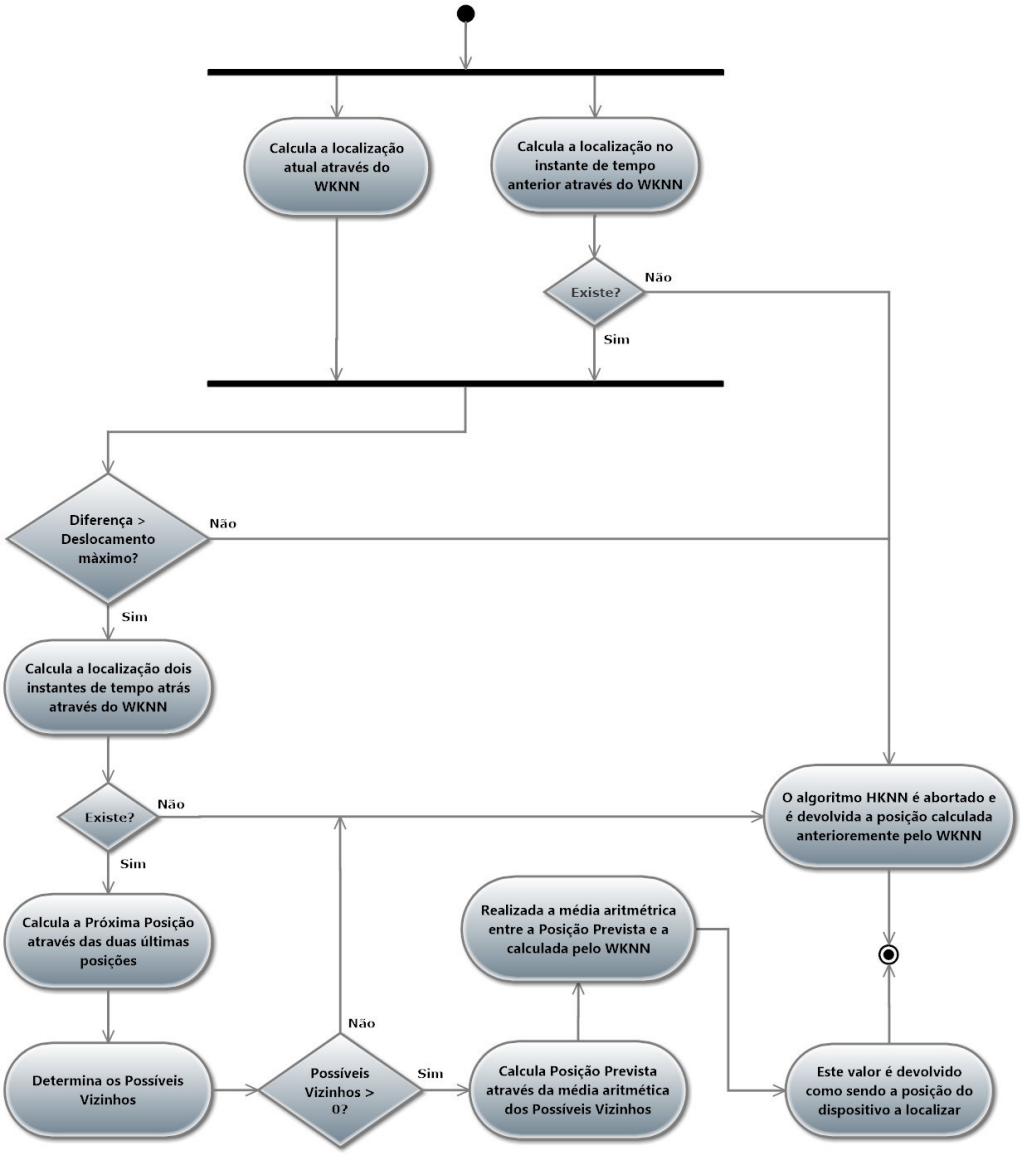


FIGURA 4.5: Diagrama de atividades do algoritmo HKNN

Todavia, caso não existam, pelo menos, as duas posições anteriores conhecidas, o algoritmo é abortado, e a posição do dispositivo será a posição inicialmente calculada pelo algoritmo WKNN. Esta anulação também acontece caso não sejam encontrados nenhuns Possíveis Vizinhos no espaço de interceção das duas circunferências. A Figura 4.5 ilustra o diagrama de atividades do algoritmo HKNN desenvolvido.

Alguns cálculos deste algoritmo baseiam-se no algoritmo PKNN, proposto por Khodayari, Maleki, e Hamedi [6], explicado na secção 3.1.1.1. Assumindo que (x_{ant1}, y_{ant1}) e (x_{ant2}, y_{ant2}) são a posição do dispositivo a localizar no instante de tempo anterior e dois instantes de tempo anteriores, respetivamente, a Próxima Posição prevista calculada pelo algoritmo é dada pela Equação 4.3

$$\begin{cases} x_{prox} = x_{ant1} \pm \Delta X \\ y_{prox} = y_{ant1} \pm \Delta Y \end{cases} \tag{4.3}$$

Em que ΔX e ΔY representam os deslocamentos possíveis nas direções x e y, respetivamente. Estes valores são calculados através das Equações 3.2, 3.3, 3.4, 3.5 e 3.6, referidas no capítulo anterior na secção 3.1.1.1.

Com este algoritmo, os utilizadores vão ser localizados nas zonas onde se obtém maior exatidão e precisão, isto é, na zona de interceção entre as duas circunferências traçadas. Deste modo, com a utilização de informação histórica, espera-se que os resultados, em relação aos algoritmos falados anteriormente, sejam melhorados.

4.2.2 *Velocity Aware K Nearest Neighbor*

O algoritmo abordado na secção anterior, o HKNN, comparativamente com os mais básicos, é bastante mais complexo em termos computacionais e, como consequência, é mais lento ao determinar as localizações dos dispositivos. Este facto advém de serem efetuados, durante o funcionamento do algoritmo HKNN, bastantes cálculos até ser determinada a posição atual de um dispositivo, como por exemplo, determinar as suas duas posições anteriores, taxa de variação de movimento, deslocamentos, etc. De forma a diminuir este problema, foi proposto um novo algoritmo que utiliza informação histórica no cálculo da localização dos dispositivos, mas que possui uma menor exigência a nível de recursos computacionais e, por isso, é mais rápido a providenciar as localizações.

Foi proposto o algoritmo *Velocity Aware K Nearest Neighbor* (VAKNN). Este algoritmo começa por calcular a posição dos dispositivos no instante de tempo em que ela é pedida, e no instante de tempo anterior (última posição conhecida). O cálculo destas posições é realizado recorrendo-se a um algoritmo sem dados históricos baseado em *Nearest Neighbors*. Durante esta explicação, usou-se como exemplo o algoritmo WKNN.

De seguida, é calculada a velocidade média de deslocação do utilizador, através da distância e do tempo entre a última posição conhecida e a posição atual calculada. Caso esta velocidade seja superior a 1,5 m/s, são selecionados todos os pontos de referência que se encontram no espaço de interceção entre as duas circunferência com raio igual à distância entre a posição calculada e a posição anterior. Este processo encontra-se ilustrado na Figura 4.6.

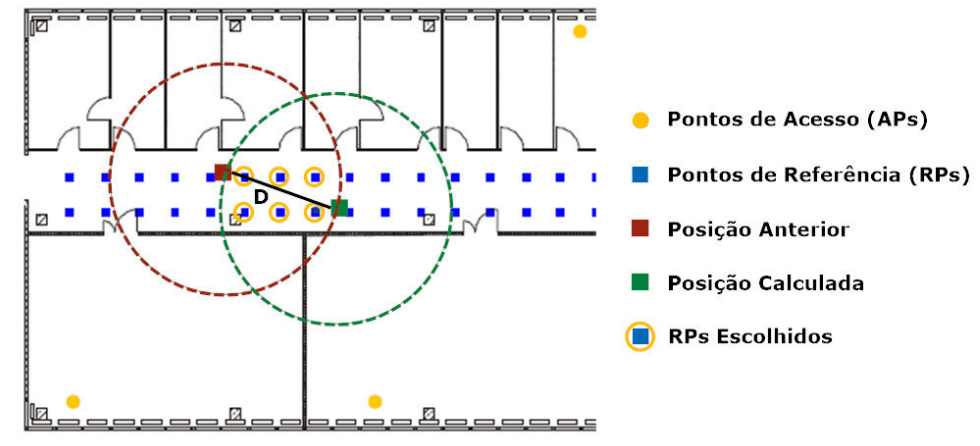


FIGURA 4.6: Escolha dos possíveis vizinhos mais próximos no algoritmo VAKNN

Caso não exista nenhum RP no espaço de interseção das duas circunferências, a posição do dispositivo será a posição calculada pelo algoritmo WKNN. No caso de existirem um ou mais RPs, é feita uma média aritmética entre todos eles, onde resultará as coordenadas locais de um ponto. Este cálculo é efetuado através da Equação 4.4.

$$(x, y) = \frac{\sum_{i=1}^{K}(x_i, y_i)}{K} \tag{4.4}$$

Por último, o algoritmo devolve o resultado desta equação como sendo as coordenadas locais do dispositivo a localizar. Para mais fácil compreensão deste algoritmo, na Figura 4.7 encontra-se representado o seu diagrama de atividades.

Com a utilização do algoritmo VAKNN, pretende-se que os resultados a nível de exatidão e de precisão não sejam muito inferiores aos obtidos no algoritmo HKNN. O principal objetivo do VAKNN era utilizar o histórico dos dispositivos no cálculo das suas localizações, mas de modo a usar-se menos cálculos no funcionamento do algoritmo, para que este determine as localizações atuais mais rapidamente. Uma das soluções encontradas para resolver este prolema foi usar apenas a posição anterior do dispositivo. Além disso, ao contrário do HKNN, não necessita de calcular a próxima posição prevista.

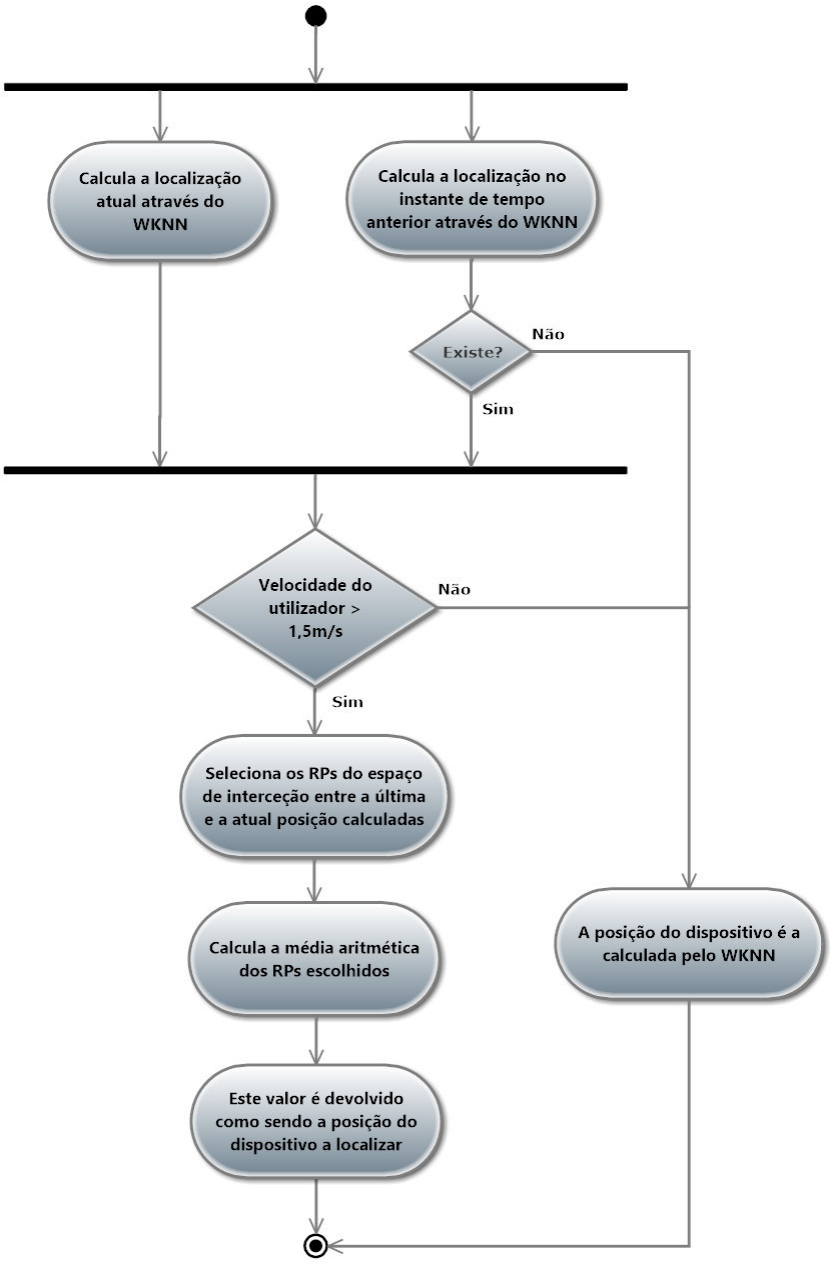


FIGURA 4.7: Diagrama de atividades do algoritmo VAKNN

4.2.3 *Restricted K Nearest Neighbor*

Outro algoritmo proposto que utiliza dados históricos dos utilizadores para o cálculo das respetivas posições é o *Restricted K Nearest Neighbor* (RKNN). Contudo, ao contrário do que acontece no HKNN e no VAKNN, em que os dados históricos apenas são implementados caso ultrapassado um limite de distância ou velocidade nas posições anteriores, no RKNN a informação histórica é implementada em todas as situações, exceto quando não existem dados que o permitam.

Este algoritmo consiste, durante o cálculo das posições através dos K vizinhos mais próximos, em fazer uma restrição destes relativamente ao passado recente do respetivo utilizador. Ou seja, os K vizinhos mais próximos escolhidos apenas podem ser pontos de referência próximos da última posição conhecida do utilizador, e não pontos que se encontram dispersos ou a distâncias consideráveis relativamente à posição anterior.

Quando é feito um pedido de localização num determinado instante de tempo, esta é calculada através da comparação do vetor de RSS obtido na fase *online*, pelo respetivo dispositivo, e os valores existente na base de dados *offline*. Neste algoritmo, esta posição obtida é denominada de Posição Estimada. Em paralelo ao cálculo da Posição Estimada, é também calculada a posição do mesmo dispositivo no instante de tempo anterior, graças à informação histórica existente na base de dados *online*. Esta posição é denominada de Posição Anterior.

Posto isto, o algoritmo foca-se agora apenas na Posição Anterior. Nesta posição, são escolhidos os pontos de referência mais próximos. Estes pontos de referência irão ser os que têm a possibilidade de serem escolhidos como K vizinhos mais próximos durante o cálculo da posição no instante de tempo seguinte (posição atual). Ou seja, são selecionados todos os RP que se encontrem num raio de 1,5 metros referente à posição anterior, e que se denominam de Possíveis Vizinhos, como representado na Figura 4.8.

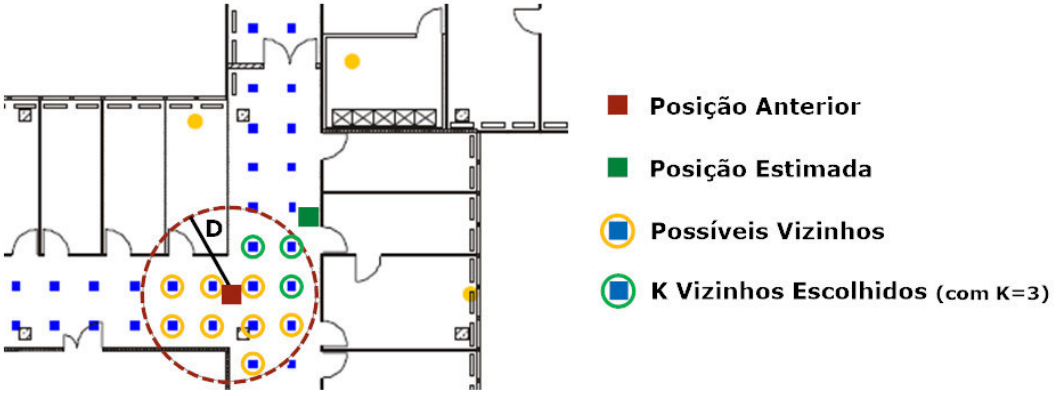


FIGURA 4.8: Seleção de três vizinhos no conjunto dos Possíveis Vizinhos

A restrição da seleção de apenas os possíveis vizinhos, evita que sejam escolhidos K vizinhos dispersos no mapa, ou pontos que na realidade não são os pontos mais próximos de uma posição. Deste modo, limita-se os possíveis vizinhos escolhidos, de acordo com a distância que um utilizador pode percorrer durante um determinado instante de tempo.

De seguida, são determinadas as coordenadas (x, y) referentes à posição atual do dispositivo a localizar, através da Equação 4.5, em que os K vizinhos mais próximos

escolhidos (x_i, y_i) têm de pertencer ao conjunto dos Possíveis Vizinhos. Onde D_i é a distância euclidiana do vizinho i, num total de V vizinhos. À semelhança do que acontece no algoritmo WKNN, no RKNN, também são atribuídos "pesos" consoante o valor da distância a que se encontra cada K vizinho. Para os casos em que não é encontrado nenhum Possível Vizinho ou não exista informação histórica referente ao dispositivo a localizar, a posição do utilizador é a posição estimada, calculada inicialmente.

$$(x, y) = \sum_{i=1}^{K} \frac{\frac{1}{D_i^2}}{\sum_{v=1}^{V} \frac{1}{D_i^2}} (x_i, y_i) \tag{4.5}$$

Com a utilização deste algoritmo, é possível corrigir a escolha de pontos de referência distanciados mas com valores de RSS semelhantes, ou até erros de leituras erradas dos valores de RSS durante a fase *online* e, como consequência, evitar a escolha de K vizinhos que se encontrem à mesma distância da posição calculada mas em direções diferentes, como ilustrado na Figura 4.9.

FIGURA 4.9: Dois vizinhos opostos mas a iguais distâncias da posição de referência

Por outro lado, o algoritmo RKNN pode trazer desvantagens nos casos em que os utilizadores do sistema se movam muito rapidamente. Nestes casos, a posição calculada irá ser sempre próxima da posição anterior, o que pode não acontecer quando o utilizador vá a correr sempre na mesma direção.

Para melhor compreensão do funcionamento deste algoritmo, na Figura 4.10 está representado o seu diagrama de atividades.

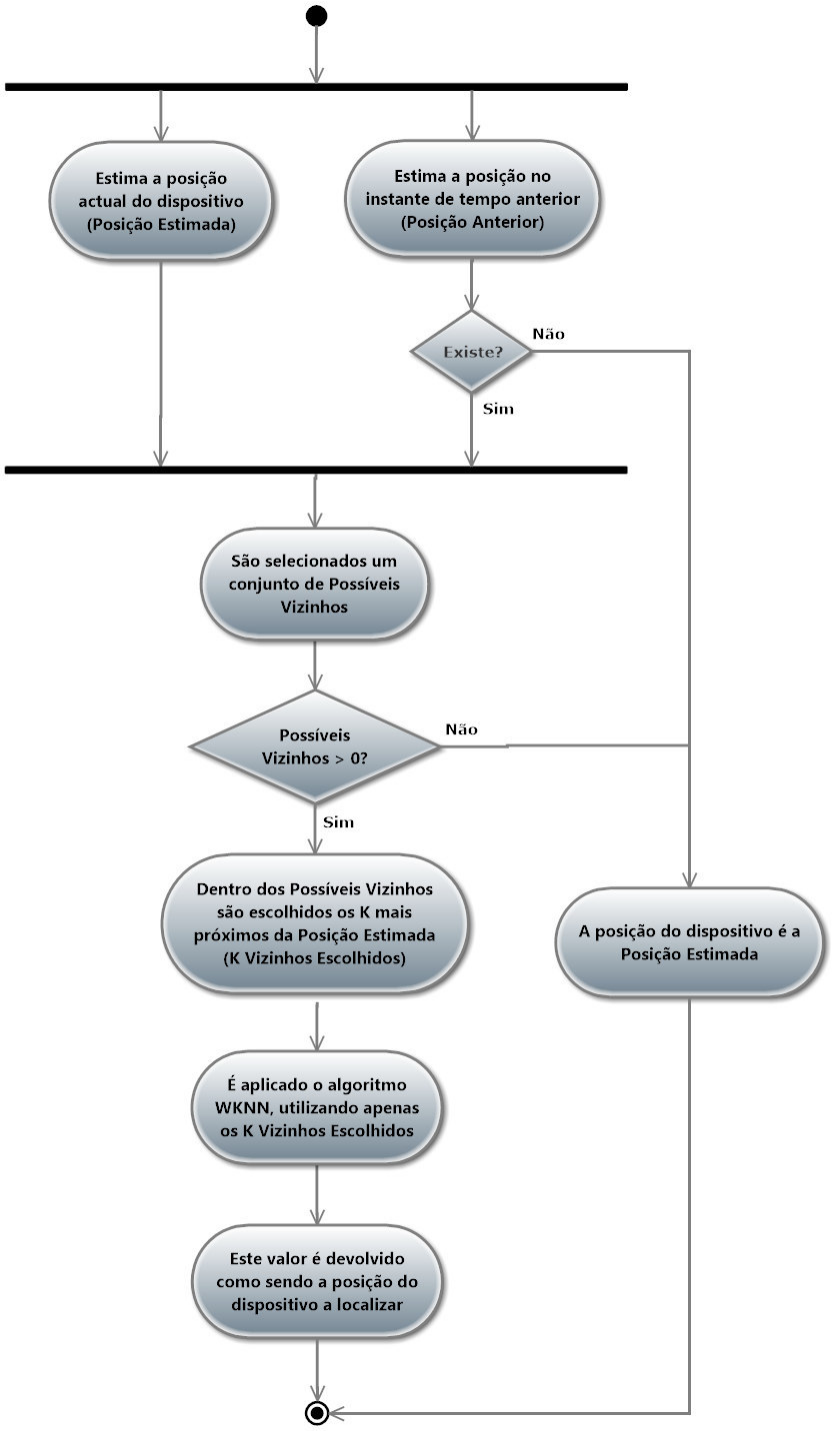


FIGURA 4.10: Diagrama de atividades do algoritmo RKNN

4.3 Algoritmos que utilizam *feedback* do utilizador

Uma solução que visa melhorar os resultados das localizações é a utilização do *feedback* dado pelos utilizadores do sistema. Este *feedback* pode ser positivo, no caso de os utilizadores confirmarem que a posição calculada pelo sistema está correta ou muito

próxima disso. Pelo contrário, quando os resultados não são satisfatórios, pode ser dado um *feedback* negativo.

Após alguma análise sobre a forma como implementar o *feedback* dos utilizadores, concluiu-se que num sistema de localização real, o *feedback* pode ser implementado usando duas diferentes técnicas: de modo *offline* ou *online*. O *feedback offline* pode ser assegurado através da instalação de um conjunto de equipamentos de rede, em posições pré-definidas do cenário, em que a sua única tarefa é ler, constantemente, os valores de RSS observados na sua posição e enviar estas leituras para a base de dados *offline*. Deste modo, iremos estar sempre a atualizar os valores de RSS da base de dados *offline* nas posições onde os equipamentos se encontram. Este conjunto de equipamentos deve ser colocado em posições onde os resultados de localização devolvidos pelos algoritmos mais tradicionais são menos satisfatórios.

A outra técnica referida, o *feedback online*, pode ser implementada pelos próprios utilizadores do sistema, ou seja, todos os utilizadores que sejam calculadas as suas posições podem dar respostas acerca do resultado da sua localização, podendo ser um *feedback* positivo ou negativo. Desta forma, depois de um período de tempo de utilização do sistema, pode-se avaliar onde é que este possui maiores falhas. Esta avaliação por parte dos vários utilizadores é, posteriormente, enviada para a base de dados *offline*, com o objetivo de melhorar os cálculos em situações futuras.

4.3.1 Localização Recalculada

Neste algoritmo foi implementada a solução descrita no parágrafo anterior, onde os utilizadores do sistema, durante a sua utilização, vão dando *feedback* sobre os resultados dos cálculos das suas posições e, com isso, enriquecendo o sistema (*feedback online*). Contudo, caso estas avaliações sejam dadas pelos utilizadores de forma errada, quer seja propositadamente ou não, irá fazer com que a exatidão do sistema venha a piorar. Por forma a evitar isto, foi pensada numa solução que previna *feedback* fraudulento, e este não prejudique os restantes utilizadores do sistema.

A solução construída consiste em utilizar o *feedback*, dado por um utilizador/dispositivo, apenas para o cálculo do posicionamento desse mesmo utilizador/dispositivo, evitando-se assim, que um ou vários utilizadores possam prejudicar os restantes. O *feedback* atribuído é identificado pelo endereço MAC dos dispositivos.

Quanto ao processo de cálculo do posicionamento dos dispositivos, este algoritmo atua quando um utilizador não fica satisfeito com a sua localização que lhe é apresentada. Este, imediatamente, pode atribuir um *feedback* negativo ao resultado da sua posição. De seguida, o utilizador terá de indicar qual a área/célula do cenário em que ele se encontra. Estas áreas/células são grandes subdivisões do cenário, que contêm dentro de si apenas o conjunto de APs que garantem os melhores resultados de localização na respetiva área. Depois de o utilizador indicar em qual das áreas ele se situa, a posição é recalculada, utilizando-se este novo algoritmo.

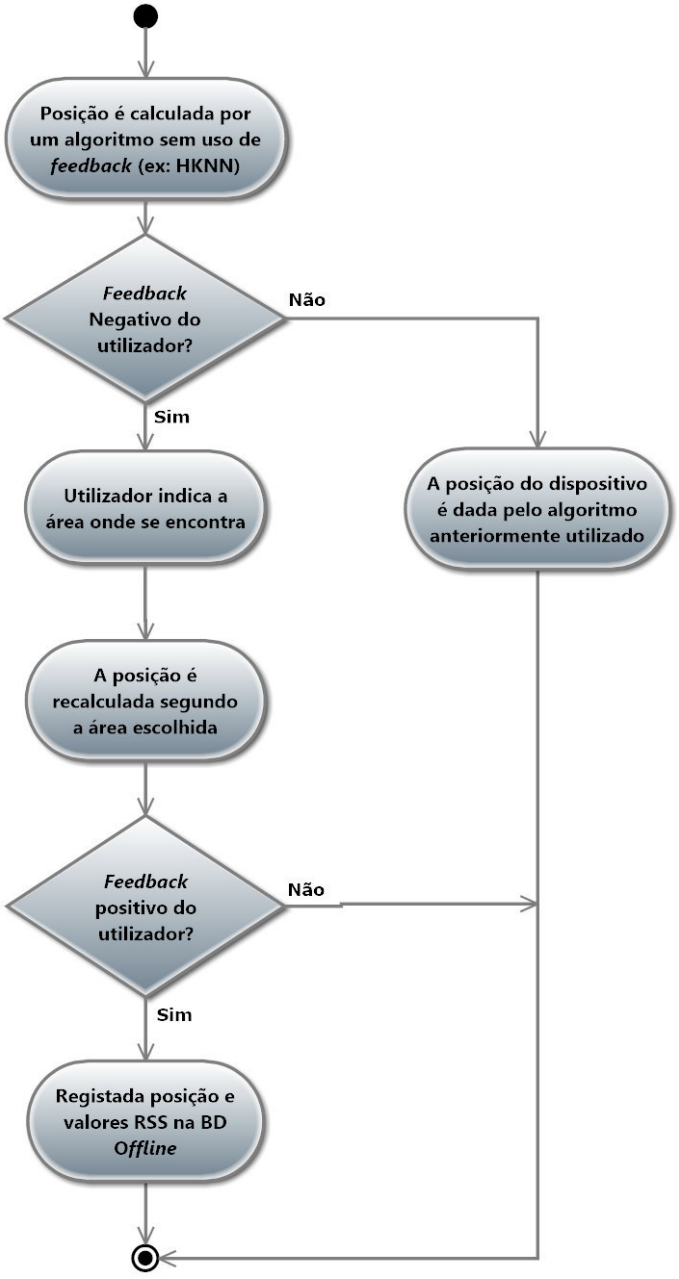


FIGURA 4.11: Diagrama de atividades do algoritmo usando Localização Recalculada

A nova estimativa da posição é efetuada através do algoritmo WKNN (ou outro implementado) e, tal como já referido, apenas são utilizados os APs que devem ser considerados na área escolhida. Caso a nova posição estimada seja aprovada pelo utilizador, isto é, desta vez ele atribuí um *feedback* positivo, são enviadas as informações dos valores de RSS observados de todos os APs do cenário, em conjunto com as coordenadas da posição calculada, para a base de dados *offline*. Esta informação apenas irá ser usada pelos algoritmos para o cálculo da posição do respetivo dispositivo que a submeteu, evitando-se assim que avaliações erradas dos utilizadores prejudiquem os restantes.

Quando é enviada nova informação para a base de dados *offline*, em primeiro lugar, é verificado se essa posição (num raio de 1 metro) já foi previamente corrigida pelo utilizador. Caso esta situação se verifique, os respetivos valores de RSS já registados serão atualizados.

Como consequência do registo dos novos dados na base de dados *offline*, a utilização deste algoritmo, ao longo do tempo, faz com que sejam criados vários pontos de referência, que poderão estar em permanente atualização. Na Figura 4.11 está representado o diagrama de atividades deste algoritmo.

4.3.2 Eliminação de *Outliers*

Os *outliers* são observações que apresentam um grande afastamento das restantes ou são inconsistentes [31]. Os *outliers* também são denominados, por vários autores, de observações *anormais*, *contaminantes*, *estranhas*, *extremas* ou *aberrantes*.

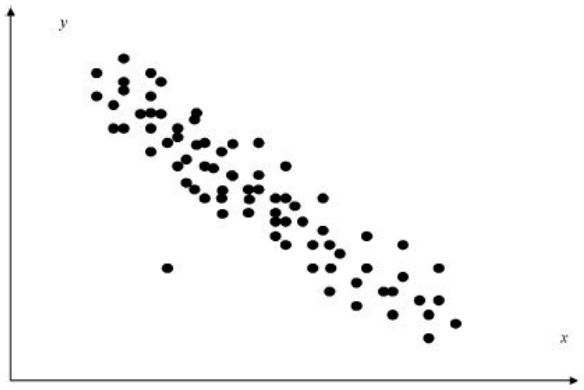


FIGURA 4.12: Deteção de um *outlier* no espaço 2D [32]

Um simples exemplo da deteção de um *outlier* é apresentado na Figura 4.12, onde, claramente, o ponto observado em baixo do lado esquerdo é um *outlier*.

Este conceito pode ser aplicado nos algoritmos de localização, mais propriamente nos algoritmos baseados no conceito de *Nearest Neighbors*. Na realidade, e devido a diversos fatores já abordados, a escolha dos K vizinhos mais próximos nem sempre é a correta, sendo que, por vezes, são escolhidos K pontos que não são os mais próximos.

A deteção de *outliers* ajuda na escolha dos K vizinhos da seguinte forma: o algoritmo escolhe os K pontos que acha que são os mais próximos e, de seguida, nessa amostra de observações verifica-se se existem *outliers*. Em caso afirmativo, um ponto que seja considerado um *outlier* é substituído pelo ponto seguinte mais próximo calculado pelo algoritmo de localização, e assim sucessivamente.

Estatisticamente, um *outlier* é detetado através da medição da sua distância em relação ao centro de distribuição médio de todas as observações. Esta distância é denominada de distância de *Mahalanobis* [32]. Distingue-se da distância euclidiana, dado que a distância de *Mahalanobis* tem em conta as correlações do conjunto dos dados.

Dadas n observações de um conjunto de dados, designa-se por $\overline{X}_n$ o vetor médio das amostras e por V_n a matriz de covariância, onde:

$$V_n = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)(X_i - \overline{X}_n)^T \tag{4.6}$$

A distância de *Mahalanobis* M_i para cada amostra i, $i = 1, ..., n$, é dada por:

$$M_i = \left(\sum_{i=1}^{n}(X_i - \overline{X}_n)^T {V_n}^{-1}(X_i - \overline{X}_n)\right)^{1/2} \tag{4.7}$$

De acordo com os autores, as observações com uma grande distância de *Mahalanobis* são indicadas como sendo *outliers*.

Como se pode verificar, este algoritmo possui uma elevada quantidade de cálculos, inclusive com matrizes, para se estimar o posicionamento de um dispositivo. Este facto torna o algoritmo bastante mais complexo e exigente computacionalmente, em comparação com os restantes algoritmos. De forma a melhorar o desempenho e rapidez nos cálculos das localizações, o processo de deteção de *outlier* apenas é aplicado quando os utilizadores dão *feedback* negativo. Assim sendo, as posições dos dispositivos são, inicialmente, calculadas por um algoritmo baseado em *Nearest Neighbors*, como por exemplo o KNN. Caso o utilizador não fique satisfeito com a localização que lhe é apresentada, este deve informar o sistema de tal facto, para que o algoritmo de eliminação de *outliers*

seja aplicado. Na Figura 4.13 encontra-se representado o diagrama de atividades deste algoritmo.

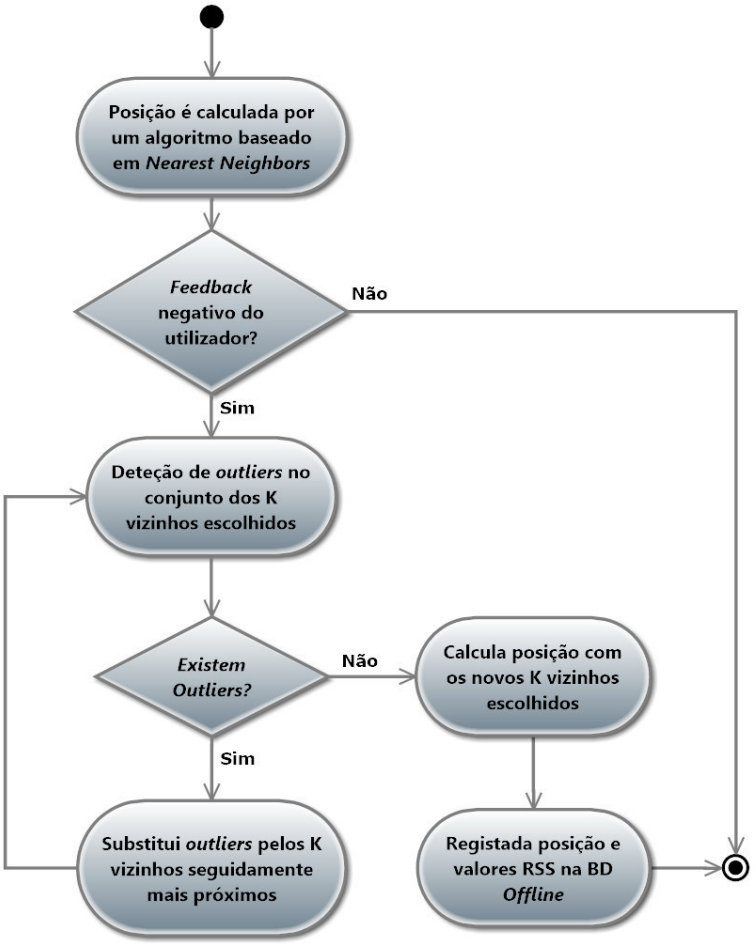


FIGURA 4.13: Diagrama de atividades do algoritmo de eliminação de *Outliers*

Tal como o algoritmo Localização Recalculada, explicado na secção anterior, o algoritmo Eliminação de *Outliers* regista na base de dados o *feedback* atribuído pelos utilizadores, de forma a melhorar os cálculos de posicionamento futuros. Para tal, é guardado na base de dados *offline* um novo ponto de referência que apenas é usado no cálculo das localizações do dispositivo que o criou.

Capítulo 5

Implementação

Depois de desenvolvidos os algoritmos, é necessário coloca-los em funcionamento. Para isso, é necessário ter um conjunto de ferramentas de acordo com uma estrutura pré-definida, por forma a aplicar os algoritmos. O objetivo deste trabalho era implementar os algoritmos propostos no protótipo existente na sequência de um projeto anterior [1]. No entanto, devido a fatores que serão explicados ao longo deste capítulo, decidiu-se aplicar os algoritmos num outro cenário diferente e, como consequência, foi necessário desenvolver uma nova base de dados. A necessidade de se desenvolver uma nova base de dados advém de a anterior estar limitada a cenários com apenas 3 APs, que é o número de APs existente no cenário do projeto anterior. Deste modo, foi desenvolvida uma nova base de dados com capacidade para guardar os dados de qualquer cenário, independentemente do número APs ou RPs que esta possua.

Além disso, foi também desenvolvida uma aplicação de testes, de forma a aplicar e testar os algoritmos no cenário escolhido. Neste capítulo é descrito todo o trabalho efetuado na implementação da base de dados e na aplicação de testes desenvolvidas, assim como a descrição detalhada do cenário de teste utilizado, as decisões tomadas nas configurações necessárias dos algoritmos e como estes foram implementados.

5.1 Base de dados

Neste projeto, os algoritmos de localização serão desenvolvidos de modo a funcionarem num serviço de localização suportado pela infraestrutura de rede, evitando-se assim, uma elevada utilização dos recursos do equipamento do utilizador. Além disso,

evita-se diferente leituras de valores de RSS, como consequência de os vários dispositivos utilizarem diferentes componentes de hardware, que podem usar diferentes escalas de valores nas suas medições. Isto faria com que o sistema não fosse homogéneo, e consequentemente poderia trazer falhas. Deste modo, a base de dados, existente na infraestrutura, torna-se num componente essencial para o funcionamento de todo o sistema de localização.

No projeto, a base de dados utilizada foi desenvolvida na linguagem SQL (*Structured Query Language*). É nela onde se encontram todas as características dos cenários onde são implementados os sistemas de localização, ou seja, os registos e valores de cada mapa *fingerprint*. Caso se pretenda utilizar os algoritmos desenvolvidos e implementados neste projeto num qualquer cenário, todos os valores daí resultantes devem ser submetidos nesta base de dados, respeitando todo o seu formato, que será explicado de seguida.

Na base de dados deste projeto foram definidas 4 tabelas: Calibração, Offline, Online e Metadados. A tabela Calibração, como o próprio nome indica, é a tabela onde são guardados os dados relativos à calibração do cenário, isto é, os dados que são originados nas recolhas efetuadas em cada ponto de referência, durante a fase *offline*. É registada a informação da posição, em coordenadas locais, de cada ponto de referência existente, bem como o valor de RSS observado nos diferente APs. A estrutura definida para esta tabela encontra-se ilustrada da Figura 5.1. O campo da tabela indicado como *idCalibração* é o valor utilizado para identificar cada entrada desta tabela, e é preenchido por incrementação automática. Por sua vez, os campos *xx*, *yy* e *zz* possuem os valores das coordenadas locais x, y e z, respetivamente, de cada RP. Por fim, o campo *RSSI* corresponde aos valores de RSS observados no AP, cujo endereço MAC (*Media Access Control*) se encontra no campo designado por *macAP*.

FIGURA 5.1: Formato da tabela Calibração

Na Figura 5.2 está ilustrada a estrutura da tabela Offline. Esta tabela contém os dados referentes à média dos valores dos APs por ponto de referência. Enquanto que na tabela Calibração são registados todos os valores das recolhas efetuadas pelos dispositivos durante a fase *offline*, na tabela Offline é resumida essa informação por ponto de referência. Isto significa que, por cada ponto de referência existente, é feita a média dos valores de RSS observados por cada AP. Esse valor é registado no campo *avgRSSI* e as respetivas coordenadas locais são introduzidas nos campos xx, yy e zz. O campo *idOffline* identifica cada registo desta tabela. Durante este trabalho, a tabela Offline é, muitas vezes, denominada de base de dados *offline*. A vantagem de se utilizar esta tabela, em vez dos dados presentes na tabela Calibração, é o facto de não ser necessário realizar os cálculos na aplicação. Desta forma, são consumidos menos recursos e o processo de cálculo de posicionamento não é atrasado. Além disso, nesta tabela também é registado o endereço MAC dos dispositivos, quando estes atribuem um *feedback* positivo (campo *macFeedback*), criando assim um novo RP. De uma forma geral, pode-se dizer que a tabela Offline é um "resumo" da tabela Calibração, com acréscimo do endereço MAC relativo ao *feedback*, caso este exista.

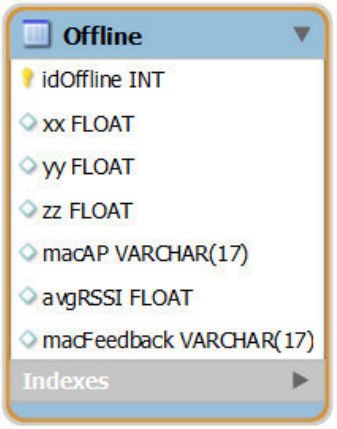


FIGURA 5.2: Formato da tabela Offline

O formato da tabela Online encontra-se representado na Figura 5.3. Esta tabela é responsável por armazenar os dados que são recolhidos pelos dispositivos a localizar, quando estes fazem um pedido de localização ao sistema. Os dados armazenados são os valores de RSS (campo *RSSI*) observados em cada AP (campo *macAP*), durante o instante de tempo de captura (campo *dataHora*). Os campos xx, yy, zz são facultativos, e podem ser preenchidos com as coordenadas locais reais correspondentes, caso sejam conhecidas. As coordenadas locais reais servirão para o cálculo da distância do erro

entre as posições estimadas pelos algoritmos de localização e as respetivas posições reais dos dispositivos. Esta tabela é, também, chamada de base de dados *online*.

FIGURA 5.3: Formato da tabela Online

Por último, na Figura 5.4 está representada a tabela metadados. Esta é uma tabela informativa, cujos dados inseridos não tem influência nos processos de localização. Nela está contida informação complementar acerca do mapa *fingerprint*, a que a base de dados se aplica. Contém informações das dimensões do cenário (campo *dimensoes*), do número de APs (campo *numAPs*) e RPs (campo *numRPs*) existentes, a data das recolhas efetuadas (campo *data*), uma breve descrição acerca do cenário (campo *descricao*) e um prefixo que identifica o *dataset* (campo *prefixo*).

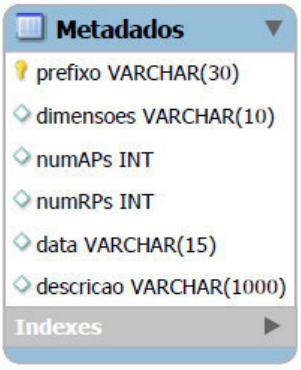


FIGURA 5.4: Formato da tabela Metadados

De forma a resumir e clarificar o fluxo de dados que ocorre na base de dados, foi construído o esquema ilustrado na Figura 5.5. Neste, os valores observados dos APs na fase *offline* são armazenados da tabela Calibração e, posteriormente, resumidos na tabela

Offline. Por outro lado, os valores observados durante a fase *online* são armazenados na tabela Online.

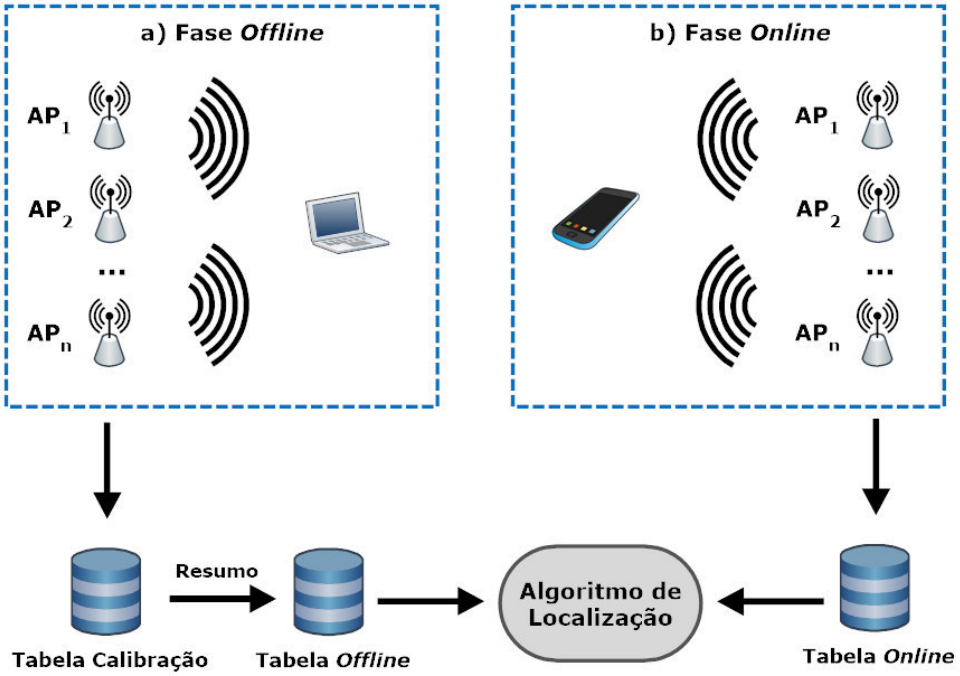


FIGURA 5.5: Fluxo de dados na base de dados

Os valores de RSS da fase *online*, assim como os respetivos instantes de tempo, devem ser armazenados na base de dados para que, durante os cálculos dos algoritmos com informação histórica, sejam calculadas as posições do passado dos dispositivos. Regra geral, os algoritmos que utilizam informação histórica necessitam de conhecer a posição dos dispositivos no seu passado recente. Esta solução, poderia ser, também, implementada guardando apenas as coordenadas locais calculadas pelos algoritmos e os respetivos instantes de tempo. No entanto, com esta solução era necessário estar constantemente a calcular e a guardar o posicionamento dos dispositivos. Pelo contrário, é mais prático guardar somente os valores de RSS e os instantes de tempo. Além disso, guardando os valores de RSS observados, permite que cada posição possa ser calculada por diferentes algoritmos nos instantes de tempo em que exista informação para tal, o que não aconteceria se apenas fossem guardadas as coordenadas calculadas, pois só existiria informação dada pelo algoritmo utilizado. No entanto, de forma a evitar um elevado crescimento da base de dados, o que poderia baixar o desempenho do algoritmo, os dados registados na tabela Online têm uma duração máxima de 1 dia.

Nesta secção foi descrita a estrutura da base de dados utilizada. Qualquer cenário no qual se pretenda pôr em prática os algoritmos de localização implementados, apenas

é necessário "injetar" os dados correspondentes ao seu mapa *fingerprint* nas tabelas da base de dados apresentada, respeitando os formatos descritos.

5.2 Cenário de teste

Inicialmente, de forma a testar-se o funcionamento e os resultados dos algoritmos desenvolvidos, preencheram-se as tabelas da base de dados, descrita na secção anterior, com informações provenientes de recolhas relativas à sala 0.10 do piso 0 do Departamento de Informática (DI) da Universidade do Minho. Este *dataset* foi desenvolvido no âmbito de um projeto anterior [1] e possui as características retratadas na Tabela 5.1.

Medidas	14,30m x 6,10m
Área de Cobertura	84m^2
Número de APs	3
Número de RPs	13

Tabela 5.1: Características do *dataset* do DI da Universidade do Minho

No entanto, decidiu-se que este *dataset* era bastante limitado dado que, a sala onde foram efetuadas as recolhas, em termos geométricos, não é a ideal, pois trata-se de uma sala pequena e que não possui corredores nem paredes entre os APs. Além disso, o número total de APs e de RPs é bastante reduzido. Deste modo, optou-se por utilizar um outro *dataset*, que satisfaça da melhor forma os requisitos necessários para o teste aos algoritmos de localização.

Após várias pesquisas, encontrou-se um *dataset* com origem na Universidade de Mannhein, na Alemanha. Este provém de recolhas efetuadas numa WLAN sob a norma IEEE 802.11b, para implementação de um sistema de localização desenvolvido por Thomas King, Thomas Haenselmann e Wolfgang Effelsberg [33]. Este *dataset* encontra-se disponibilizado pelos autores na plataforma *online* CRAWDAD, referenciado em [34].

As recolhas foram realizadas no segundo piso do Departamento de Ciências da Computação da Universidade de Mannhein e as suas características estão descritas na seguinte tabela.

A área de operação do sistema é de, aproximadamente, 57 metros de largura por 32 metros de comprimento, num total de 221 metros quadrados de área coberta.

Medidas	57m x 32m
Área de Cobertura	221m^2
Número de APs	12
Número de RPs	130

TABELA 5.2: Características do *dataset* da Universidade de Mannhein

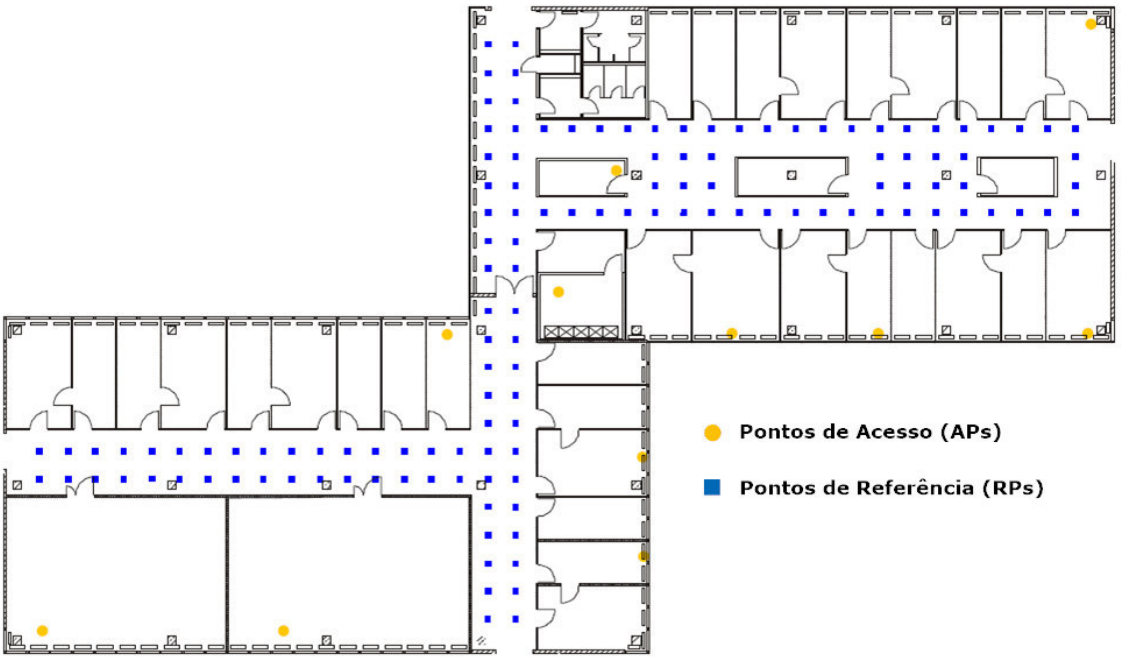


FIGURA 5.6: Mapa *fingerprint* do *dataset* utilizado

Neste cenário, a maioria dos APs apenas têm "cobertura" em certas partes do edifício de teste, e apenas dois deles podem ser alcançados em qualquer lugar do edifício. Um desses APs encontra-se situado no meio do corredor horizontal, na parte do lado direito do edifício. O segundo AP encontra-se num escritório localizado no piso inferior ao cenário de teste. As posições de todos os APs localizados no mesmo piso e dentro da área de cobertura do cenário de teste, bem como os 130 pontos de referência existentes, estão marcados, respetivamente, com círculos cor de laranja e quadrados azuis na Figura 5.6. Os RP encontram-se separados entre si por uma distância de 1,5 metros. As posições de cada AP do cenário encontram-se descritas na Tabela 5.3.

Em termos de *hardware*, todas as recolhas da fase *offline* e, também, da *online*, foram realizadas com um computador portátil IBM Thinkpad R51, com o sistema operativo Linux kernel 2.6.13 e com placa de rede Lucent Orinoco Silver PCMCIA.

Endereço MAC do AP	**Posição (x,y) (em coordenadas locais)**
00:14:BF:B1:7C:54	(-23.626, -18.596)
00:16:B6:B7:5D:8F	(-10.702, -18.596)
00:14:BF:B1:7C:57	(8.596, -14.62)
00:14:BF:B1:97:8D	(8.538, -9.298)
00:16:B6:B7:5D:9B	(-1.93, -2.749)
00:11:88:5A:B9:00	(4.035, -0.468)
00:14:BF:3B:C7:C6	(13.333, -2.69)
00:14:BF:B1:97:8A	(21.17, -2.69)
00:14:BF:B1:97:81	(32.398, -2.69)
00:11:88:5A:B9:60	(32.573, 13.86)
00:11:88:28:5E:E0	(7.135, 6.023)

TABELA 5.3: Posicionamento dos APs no mapa *fingerprint*

Durante a fase *offline*, foram recolhidas 110 amostras de força do sinal recebido em cada ponto de referência. Na fase *online* foram escolhidas ao acaso 46 posições do mapa *fingerprint*, e novamente, foram recolhidas 110 amostras em cada posição. Nenhuma destas posições coincide exatamente com as posições dos pontos de referência.

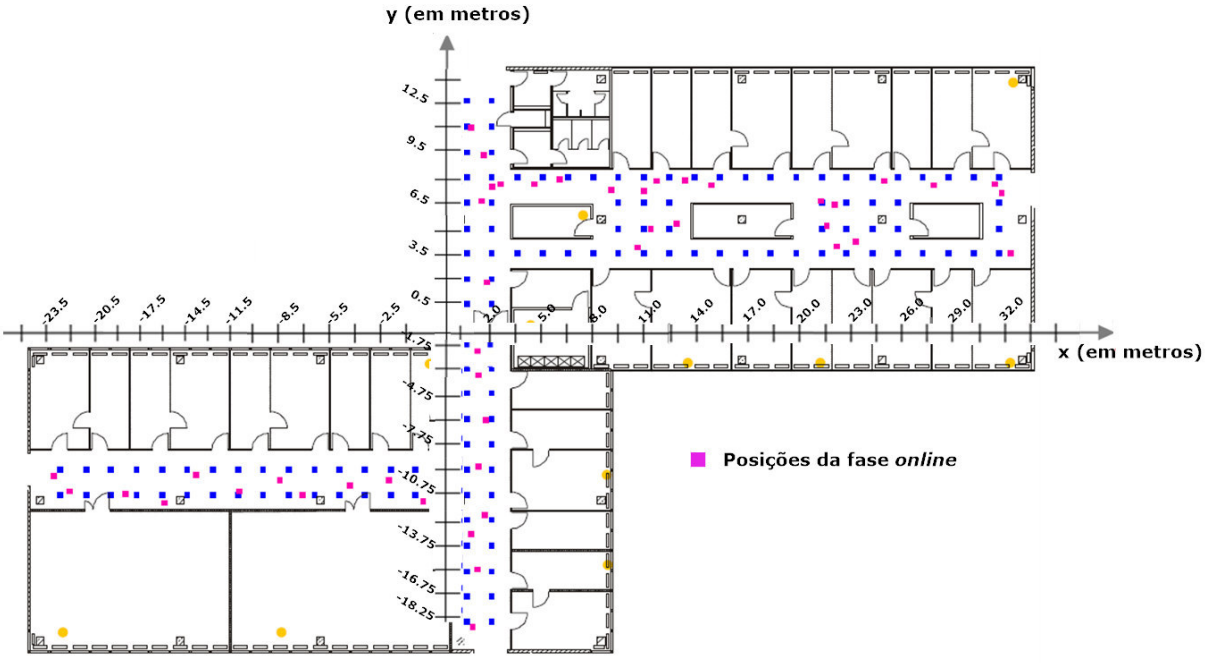


FIGURA 5.7: Mapa *fingerprint* com os eixos das coordenadas locais do *dataset* utilizado

As posições das recolhas efetuadas na fase *online* estão representadas por quadrados cor-de-rosa, na Figura 5.7. De forma a compreender o posicionamento de todos os componentes do cenário, foi desenhada, nesta mesma figura, uma grelha com a respetiva escala em metros.

5.2.1 Movimento do utilizador

Este *dataset* possui uma característica fundamental para a utilização de informação histórica nos algoritmos de localização, que é a existência de movimento de um certo utilizador, a que chamamos movimento *online*. Esta característica consiste na sequência de posições do utilizador durante um determinado trajeto, formando assim um movimento contínuo no espaço e no tempo. Toda esta informação histórica existente é referente a apenas um utilizador em concreto, que transportava um computador portátil IBM Thinkpad R51, com o endereço MAC 00:02:2D:21:0F:33. O movimento *online* deste utilizador está ilustrado na Figura 5.8.

FIGURA 5.8: Movimento de um utilizador no mapa *fingerprint*

Como já foi referido, os quadrados marcados a cor-de-rosa no mapa *fingerprint* representam as posições reais onde foram medidos os valores de RSS na fase *online*. Deste modo, estes pontos serão os locais em que os algoritmos de localização implementados vão tentar localizar o utilizador.

Em suma, após as várias experiências realizadas pelos autores, o melhor resultado por eles obtido foi de 2,86 metros, através de um algoritmo probabilístico.

5.3 Aplicação de teste

De forma a facilitar a avaliação dos algoritmos de localização implementados foi, desenvolvida uma aplicação que executa os algoritmos nos respetivos cenários de teste. Esta aplicação de teste tem como funcionalidade dar resposta aos pedidos de localização a ela efetuados. É responsável, também, por fazer consultas à base de dados existente, de forma a obter os dados relevantes, a fim de poder aplicar os algoritmos de localização, como por exemplo, valores de RSS dos dispositivos, coordenadas dos cenários, valores de RSS dos pontos de referência, entre outros.

A aplicação foi desenvolvida para PC através da linguagem de programação JAVA. A interação da aplicação com a base de dados é feita recorrendo-se ao driver JDBC (*Java DataBase Connectivity*).

O funcionamento da aplicação de teste consiste, basicamente, em introduzir como *input* informações acerca dos dispositivos localizáveis e qual o algoritmo a usar para os localizar. Depois de o algoritmo escolhido efetuar os respetivos cálculos, a aplicação devolve o resultado calculado.

Quando a aplicação é iniciada existem duas diferentes possibilidades, ou introduz o endereço MAC e o instante de tempo do dispositivo que se pretende localizar ou, então, pode-se optar por importar pedidos de localização de um ficheiro de texto. Em ambas as situações, quem se encontra a utilizar a aplicação tem a possibilidade de escolher qual o algoritmo que deseja que esta utilize nos seus cálculos. Esta escolha é efetuada no separador "Algoritmos", como mostra a Figura 5.9. Se o algoritmo escolhido for baseado em *Nearest Neighbors*, é necessário definir o valor pretendido para o K.

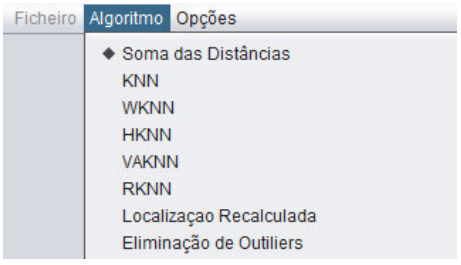


FIGURA 5.9: Escolha do algoritmo a utilizar

Caso se opte pela primeira possibilidade, depois de introduzidos os valores requeridos, apenas é necessário clicar no botão "Localizar", e a aplicação apresentará o resultado da localização em coordenadas cartesianas locais na interface "Resultado:", juntamente com a distância de erro obtida, caso existam dados para a calcular. Os valores de RSS de todos os APs no alcance do dispositivo indicado e no instante de tempo pretendido, incluindo APs que não pertencem ao sistema, são também apresentados na interface "APs e Forças de Sinal Recebido:". Na Figura 5.10 está ilustrado um exemplo deste *output* obtido durante a execução da aplicação de testes.

FIGURA 5.10: Interface da aplicação de testes

Se for escolhida a segunda possibilidade, a importação de um ficheiro, é necessário submeter um ficheiro de texto que respeite um formato pré-definido, de forma a que a aplicação possa recolher informação sobre os endereços MAC e os instantes de tempo dos dispositivos a localizar. A estrutura de cada linha do ficheiro que contém um ou vários pedidos de localização tem, obrigatoriamente, de possuir o seguinte formato:

```
mac="Endereço MAC";insTempo="DD-MM-AAAA HH:mm";
```

Em que o campo `mac` corresponde ao endereço MAC do dispositivo a localizar e o campo `insTempo` corresponde ao instante de tempo no qual se pretende localizar.

Até este ponto, foram detalhados todos os dados de entrada na aplicação de testes. Já os dados de saída, isto é, os resultados devolvidos pela aplicação, podem ser visualizados de três diferentes formas. Por defeito, os resultado obtidos são mostrados na própria aplicação, na interface "Resultados". No entanto, do mesmo modo que se pode importar dados de um ficheiro, também, é possível exportar os resultados para ficheiros. São disponibilizadas duas opções distintas para exportação dos resultados em ficheiros, que podem ser consultadas na Figura 5.11.

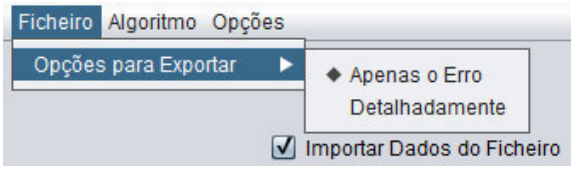


FIGURA 5.11: Diferentes opções de exportação dos resultados obtidos

Caso se opte por exportar os resultados de forma detalhada, o ficheiro de texto possuirá várias informações sobre o que foi feito, como por exemplo, os tipos de algoritmos utilizados, os instantes de tempo, as distâncias do erro obtidas ou as coordenadas locais das posições calculadas. Como na base de dados existe a opção para se armazenar a posição real dos dispositivos na fase *online*, é possível, caso tal informação exista, calcular a distância do erro entre a posição real dos dispositivos e as respetivas posições calculadas pelos algoritmos. Deste modo, também é colocado para o ficheiro de exportação a distância de erro obtida dos cálculos efetuados. Assim sendo, o ficheiro obtido quando se opta por exportar os resultados de forma detalhada, possui o seguinte formato:

```
MAC="Endereço MAC";t="DD-MM-AAAA HH:mm";Algoritmo="Algoritmo Utilizado";
PosCalc[X][Y][Z]="Posição Calculada";PosReal[X][Y][Z]="Posição Real";
Erro="Distância do Erro em metros"
```

Quando se opta por exportar apenas o erro, o resultado do ficheiro de texto terá apenas os valores das distâncias de erro, em metros, dos pedidos de localização efetuados. Este tipo de exportação é bastante útil nos casos em que se pretende estudar minuciosamente as distâncias de erros obtidas, pois é possível importar estes ficheiros em *softwares* estatísticos, de forma a construir-se gráficos ou tabelas com estes dados.

5.4 Algoritmos implementados

Depois de descrita a base de dados, o cenário utilizado e a aplicação de testes, nesta secção são apresentados alguns aspetos relativos à implementação dos algoritmos. No total, foram implementados oito algoritmos: soma das distâncias, KNN, WKNN, HKNN, VAKNN, RKNN, Localização Recalculada e Eliminação de *Outliers*, cujos seus métodos de funcionamento foram descritos nos capítulos anteriores. Dos oito algoritmos implementados, seis deles foram propostos no âmbito deste projeto e dois deles foram estudados anteriormente, embora implementadas algumas alterações. Essas alterações consistem em duas soluções que foram pensadas com o objetivo de melhorar a exatidão dos resultados e foram aplicadas em todos os algoritmos implementados.

A primeira solução foi definir uma escala de "pesos" para os diferentes valores de RSS. Esta solução tem como objetivo atribuir "pesos" aos valores de RSS de acordo com a sua grandeza, isto é, um valor RSS mais elevado possui uma maior confiança de ter sido corretamente lido pelos dispositivos, do que um valor que seja menor. Assim sendo, foi definida uma escala de valores de RSS e respetivos "pesos" atribuídos, com base num artigo técnico [35]. Esta escala encontra-se representada na Tabela 5.4. Como se pode verificar, quanto melhor o valor de RSS maior será o seu "peso", ou seja, será depositada uma maior confiança nesse valor. Esta solução é aplicada aos valores de RSS referentes à fase *online*. É importante referir que os valores de RSS são medidos numa escala negativa, ou seja, quanto mais próximo do 0 mais qualidade eles possuem, sendo que o valor 0 ocorre quando um AP está fora do alcance aquando da medição.

Intervalo valores de RSS	**Qualidade do sinal**	**"Peso" atribuído**
-1 dB a -39 dB	Excelente	5
-40 dB a -54 dB	Muito bom	4
-55 dB a -69 dB	Bom	3
-70 dB a -79 dB	Má	2
-80 dB a $-\infty$	Muito má	1
0	Fora do alcance	1

Tabela 5.4: "Pesos" atribuídos aos valores de RSS na fase *online*

Por último, outra medida proposta e aplicada em todos os algoritmos consiste em desprezar, para cálculo das distâncias euclidianas, os AP's cujos valores de RSS obtidos

numa das fases (*online* ou *offline*) sejam 0, e ao mesmo tempo sejam 0 ou inferiores a -80 dB na fase contrária. Esta medida foi proposta para evitar situações em que, por exemplo, um AP está fora do alcance do dispositivo na fase *online*, ou seja, possui o valor de 0 dB mas, no entanto, durante a fase *offline* na mesma posição, o sinal desse mesmo AP apesar de ser muito fraco era captado. Desta forma, evita-se comparar o valor 0 ocorrido numa das fases com valores abaixo de -80 dB, ocorridos na fase oposta.

Seguidamente, são apresentados alguns procedimentos técnicos que foram realizados na implementação dos algoritmos. Todos eles, tal como a aplicação de testes, foram desenvolvidos através da linguagem de programação *Java*. As consultas e registos efetuados com a base de dados são feitas através ao driver JDBC (*Java DataBase Connectivity*).

- Os valores de RSS, obtidos da base de dados *offline*, são colocados num HashMap, cuja chave identifica o *id* do AP e o valor é a potência de sinal recebida do AP correspondente.
- Os valores de RSS da fase *online* são colocados num *ArrayList*.
- Os endereços MAC de todos os APs são guardados num *array* de *Strings*, em que a posição do AP ao longo do *array* é o seu *id*.
- Existe uma função que, recebendo como *input* um valor RSS, devolve o respetivo "peso" atribuído.
- Todos os valores de RSS, como também as coordenadas das posições e as distâncias em metros, são definidas com o tipo *Float*.

A maioria dos algoritmos implementados necessitam que sejam definidas, a priori, algumas configurações. Estas configurações consistem no número de K vizinhos mais próximos a utilizar-se nos algoritmos que se baseiam em *Nearest Neighbors*, a velocidade média de um humano, a distância de deslocamento máxima permitida, a velocidade máxima de um utilizador, entre outras. O valor de K, em todos os algoritmos que o aplicam, foi definido como 3. Chegou-se a este valor através de alguns testes realizados, que estão descritos no capítulo seguinte, na secção 6.1.

No algoritmo HKNN é necessário definir a velocidade média de um humano e, também, o deslocamento máximo permitido entre atual e a última posição calculada. Estes valores foram definidos como 1,5 m/s e 1,5 metros, respetivamente. Já no algoritmo RKNN é necessário definir um valor para o raio da circunferência da última

posição calculada e, nos testes realizados, foi escolhido 1,5 metros. Por outro lado, no algoritmo VAKNN, foi definido o valor 1,5 m/s como a velocidade necessária para que este algoritmo seja executado. Alguns destes valores têm como base a distância entre os RPs do cenário utilizado, que é 1,5 metros. O valor da velocidade média de um humano foi escolhido com base no artigo referenciado em [6].

Por último, no algoritmo Eliminação de *outliers*, é necessário escolher-se um valor que defina a sensibilidade da deteção de *outliers*. Esse valor é a distância da qual os valores que se situam acima são considerados *outliers*. Neste trabalho, definiu-se tal valor em 3 metros. A Tabela 5.5 apresenta, de forma sucinta, como foram definidas as configurações necessárias.

Algoritmo	**Valor de K**	**Velocidade humano**	**Deslocamento permitido**	**Velocidade máxima**	**Distância para *Outlier***
Soma das Distâncias	-	-	-	-	-
KNN	3	-	-	-	-
WKNN	3	-	-	-	-
HKNN	3	1,5 m/s	1,5m	-	-
RKNN	3	-	1,5m (raio)	-	-
VAKNN	3	-	-	1,5 m/s	-
Localização Recalculada	3	-	-	-	-
Eliminação de *Outliers*	3	-	-	-	3m

Tabela 5.5: Configurações dos algoritmos implementados

Capítulo 6

Testes e resultados

Finalmente, depois de desenvolvidos e implementados os algoritmos de localização e integrados com a base de dados relativa ao cenário escolhido, procedeu-se à execução, através da aplicação de localização, de uma série de testes. Foram levadas a cabo um conjunto de medições, de forma a verificar e certificar o comportamento de todos os algoritmos implementados.

Neste capítulo, são apresentados todos os testes e experiências realizadas, juntamente com os respetivos resultados obtidos na utilização dos diferentes algoritmos no cenário escolhido.

6.1 Escolha do K

Em algoritmos onde é utilizado o conceito de *Nearest Neighbors*, a escolha do valor de K influencia o desempenho dos mesmos. De forma a avaliar a sensibilidade dos algoritmos tendo em conta o número de K vizinhos mais próximos, realizaram-se algumas experiências. Estas consistiram na utilização dos diferentes algoritmos na localização de dispositivos, usando-se diferentes valores de K vizinhos mais próximos.

Foram utilizados os 46 pontos *online*, disponíveis no *dataset* providenciado pelos investigadores da Universidade de Mannhein, e onde, no total, existem 130 pontos de referência (RPs). Assim, o valor de K tem de estar compreendido entre 1 e 130. Os testes foram realizados, apenas, com K até 15, e somente são apresentados para os algoritmos KNN, WKNN e HKNN, de forma a não ser apresentado um excesso de informação, pois para os outros valores de K e para os restantes algoritmos baseados em *Nearest Neighbors*, o efeito obtido foi idêntico.

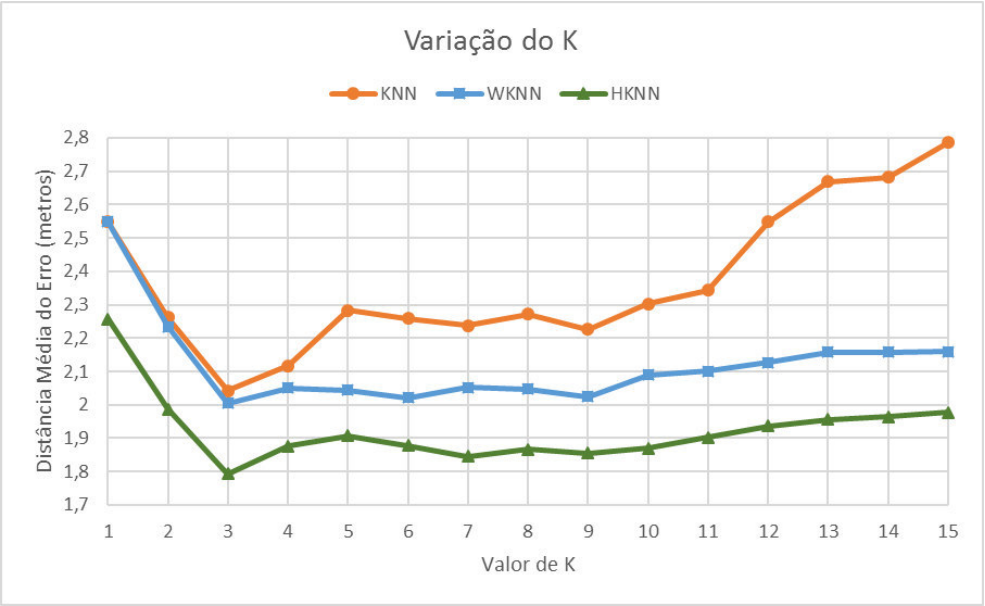


FIGURA 6.1: Distâncias médias do erro consoante o valor de K

Pela análise do gráfico representado na Figura 6.1, conclui-se, de uma forma geral, que a partir do valor 3, quanto maior o valor de K escolhido, maior será a distância média do erro. No entanto, este efeito é menor no caso dos algoritmos WKNN e HKNN, em que as respetivas variações do erro com o aumento do K não são tão acentuadas.

É importante referir que, nos algoritmos KNN e WKNN, a distância do erro obtida quando o K possui o valor 1 é a mesma que a obtida no algoritmo soma das distâncias. Os algoritmos baseados em *Nearest Neighbors* mais básicos, sem uso de dados históricos, como é o caso do KNN e do WKNN, quando utilizam apenas 1 vizinho mais próximo, funcionam exatamente da mesma forma que o algoritmo soma das distâncias, daí os resultados serem os mesmos. A partir de dois vizinhos escolhidos, os algoritmos apresentam resultados diferentes do algoritmo soma das distâncias.

Em suma, o melhor valor de K vizinhos mais próximos é de três, onde se obteve distâncias médias do erro de 2,04 metros, no caso do KNN, de 2,00 metros no caso do WKNN e, por fim, de 1,79 metros no HKNN. Na tabela Tabela 6.1 estão indicados outros detalhes acerca das experiências efetuadas com diferentes valores de K vizinhos escolhidos, especificamente, nos algoritmos mais básicos (KNN e WKNN).

Pode-se verificar que as distâncias de erro máximas e mínimas obtidas, também se alteram conforme a variação do número de K vizinhos mais próximos.

Em todos os testes realizados aos algoritmos de localização implementados, que são detalhadamente apresentados na secção seguinte, foram realizados usando-se três

Algoritmo	**K vizinhos**	**Erro Médio (metros)**	**Erro Máximo (metros)**	**Erro Mínimo (metros)**
KNN	2	2,26	7,81	0,39
KNN	3	2,04	7,82	0,32
KNN	5	2,28	9,01	0,19
KNN	9	2,22	8,10	0,23
KNN	15	2,78	8,52	0,51
WKNN	2	2,23	7,37	0,18
WKNN	3	2,00	7,47	0,32
WKNN	5	2,04	8,47	0,17
WKNN	9	2,02	8,15	0,18
WKNN	15	2,15	8,35	0,42

Tabela 6.1: Efeito da variação do valor de K

vizinhos mais próximos. Optou-se por este valor pois, devido às experiências efetuadas para diferentes valores de K, conclui-se que esse valor é o melhor, devido a apresentar, em média, uma distância de erro mais baixa.

6.2 Testes aos algoritmos de localização

No *dataset* providenciado pelos investigadores da Universidade de Mannhein, os diferentes algoritmos de localização foram testados em 46 pontos do cenário, que podem ser visualizados na Figura 5.6 ou, mais detalhadamente, na Figura 5.7, e ainda o movimento do utilizador na Figura 5.8.

Como já foi explicado, no *dataset* da Universidade de Mannhein, foram recolhidas 110 amostras em cada uma das 46 posições *online*. No entanto, nos testes realizados, foi apenas selecionada, de forma aleatória, uma amostra por posição. As posições reais testadas e os respetivos instantes de tempo de localização encontram-se descritos na Tabela 6.2. O campo *ID* é o identificador de cada posição, que servirá para identificar, de forma escrita, qual a posição que está a ser referida.

Nas secções seguintes, são apresentados, de uma forma geral, os resultados obtidos em cada um dos algoritmos implementados. De uma forma mais detalhada são apresentados, também, os resultados obtidos em cada uma das posições testadas, que podem ser consultados em anexo, no Apêndice A.

ID	Pos. Real	Instante Tempo	ID	Pos. Real	Instante Tempo
1	(-23.9, -9.65)	24-08-2007 01:40:33	24	(1.52, 9.32)	24-08-2007 02:22:22
2	(-23.0, -10.55)	24-08-2007 01:42:13	25	(0.78, 10.94)	24-08-2007 02:23:58
3	(-19.65, -10.7)	24-08-2007 01:43:58	26	(1.39, 6.61)	24-08-2007 02:25:33
4	(-17.35, -11.2)	24-08-2007 01:45:38	27	(4.51, 7.63)	24-08-2007 02:27:08
5	(-15.5, -9.6)	24-08-2007 01:47:13	28	(6.0, 7.88)	24-08-2007 02:28:38
6	(-12.9, -10.5)	24-08-2007 01:48:49	29	(9.08, 7.24)	24-08-2007 02:31:28
7	(-10.55, -9.9)	24-08-2007 01:50:26	30	(10.62, 3.87)	24-08-2007 02:33:07
8	(-9.2, -10.75)	24-08-2007 01:52:06	31	(10.99, 7.19)	24-08-2007 02:34:42
9	(-6.35, -10.2)	24-08-2007 01:54:21	32	(11.76, 7.76)	24-08-2007 02:36:16
10	(-4.1, -9.9)	24-08-2007 01:56:01	33	(11.39, 5.0)	24-08-2007 02:37:54
11	(-2.05, -11.1)	24-08-2007 01:57:39	34	(12.95, 5.25)	24-08-2007 02:39:34
12	(-0.95, -9.95)	24-08-2007 01:59:26	35	(13.46, 7.85)	24-08-2007 02:41:11
13	(0.85, -18.55)	24-08-2007 02:02:08	36	(14.98, 7.55)	24-08-2007 02:43:00
14	(1.15, -15.15)	24-08-2007 02:03:53	37	(21.45, 6.62)	24-08-2007 02:44:37
15	(0.75, -13.05)	24-08-2007 02:05:36	38	(21.84, 5.16)	24-08-2007 02:46:11
16	(1.55, -11.95)	24-08-2007 02:07:12	39	(22.38, 3.94)	24-08-2007 02:47:46
17	(1.2, -9.05)	24-08-2007 02:08:53	40	(22.3, 6.36)	24-08-2007 02:49:22
18	(1.65, -6.3)	24-08-2007 02:10:33	41	(28.12, 7.57)	24-08-2007 02:52:37
19	(1.2, -3.7)	24-08-2007 02:12:05	42	(31.78, 7.62)	24-08-2007 02:54:15
20	(1.85, -2.25)	24-08-2007 02:13:39	43	(32.16, 7.08)	24-08-2007 02:55:51
21	(1.71, 1.81)	24-08-2007 02:17:00	44	(32.68, 3.48)	24-08-2007 02:57:33
22	(2.02, 7.45)	24-08-2007 02:18:45	45	(23.5, 4.22)	24-08-2007 03:00:40
23	(2.49, 7.6)	24-08-2007 02:20:23	46	(25.23, 7.78)	24-08-2007 03:02:22

Tabela 6.2: Posições e instantes de tempo da fase *online*

6.2.1 Soma das Distâncias

A primeira série de medições e respetivos testes foram realizados com o algoritmo soma das distâncias. As posições estimadas por este algoritmo, bem como as respetivas distâncias dos erros obtidos, são apresentadas em anexo na Tabela A.1. O erro obtido é a distância euclidiana entre a posição calculada pelo algoritmo e a sua verdadeira posição. As posições reais podem ser consultadas pelo respetivo ID na Tabela 6.2.

Analisando os resultados obtidos na Tabela A.1 e no histograma ilustrado na Figura 6.2, pode-se verificar que, na maioria das posições de teste, o erro mais frequente situa-se entre 0 e 1 metros e, também, entre os 2 e 3 metros. Como já foi referido, este é

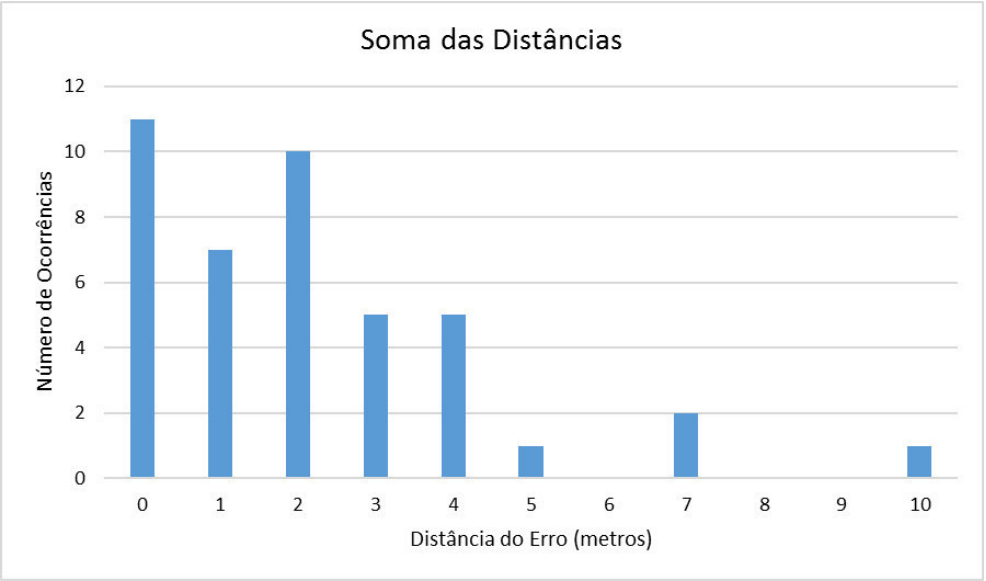


FIGURA 6.2: Distância do erro no algoritmo soma das distâncias

o algoritmo mais simples, e apenas consegue calcular a posição de um dispositivo como sendo a posição do ponto de referência mais próximo.

No entanto, em algumas posições o erro obtido é baixo como, por exemplo, nas posições com o ID 11, 21, 22, 28 e 30 que apresentam erros inferiores a 60 centímetros. Por outro lado, em posições como, por exemplo, a 4, 5, 7, 18, 19 e 29, o erro é bastante grande, superior a 5 metros. A distância de erro mínima obtida foi de 0,34 metros (ID 21) e a máxima foi de 10,07 metros (ID 19). Nas 46 posições diferentes testadas, foi obtida uma distância média do erro de 2,82 metros.

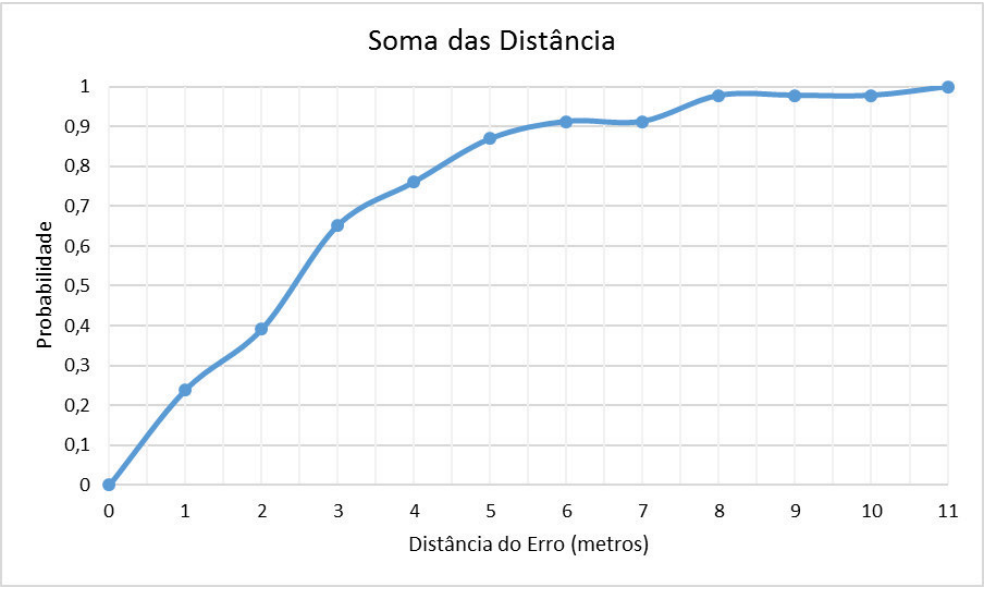


FIGURA 6.3: Distribuição da probabilidade acumulada da distância do erro

De forma a compreender melhor os resultados obtidos neste algoritmo, foi construído um gráfico com a distribuição da probabilidade acumulada, representado na Figura 6.3. Analisando este gráfico, conclui-se que cerca de metade dos casos possui uma distância de erro inferior a 2,50 metros e são poucos os valores em que se obteve um erro superior a 6 metros (cerca de 10%), o que significa que, em 90% dos casos, a distância do erro foi inferior a 6 metros.

6.2.2 *K Nearest Neighbor*

De seguida realizaram-se os testes ao segundo algoritmo de localização implementado, o KNN. Este algoritmo já é mais complexo do que o algoritmo soma das distâncias, e nesta secção irá verificar-se se os resultados obtidos são melhores.

As posições calculadas pelo KNN, bem como as respetivas distâncias dos erros obtidas, encontram-se na Tabela A.2, em anexo. É importante referir que, as posições testadas são as mesmas do algoritmo anterior, e as mesmas dos algoritmos que serão abordados mais à frente, de forma a garantir-se uma homogeneidade dos resultados.

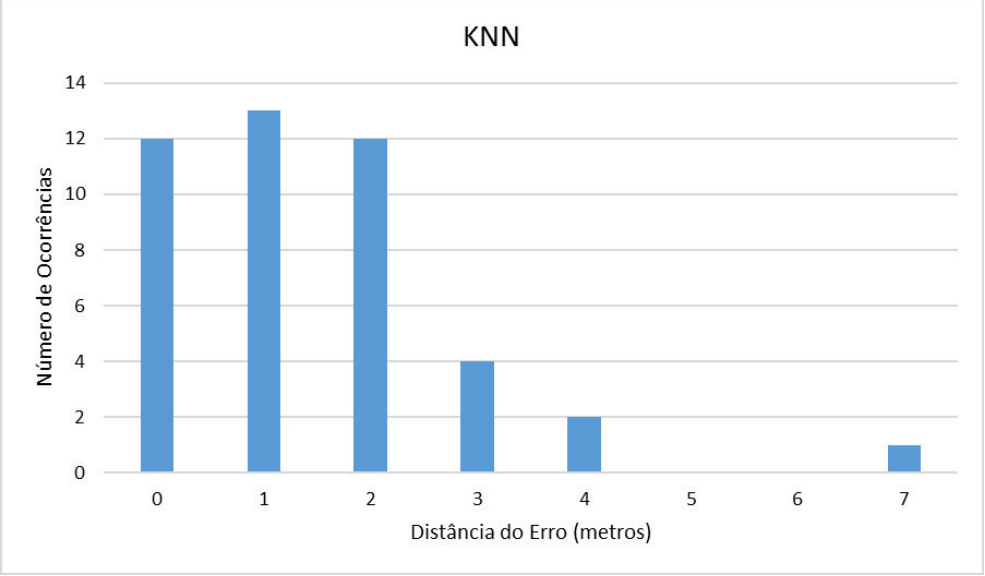


FIGURA 6.4: Distância do erro no algoritmo KNN

Analisando os resultados obtidos na Tabela A.2 e no histograma ilustrado na Figura 6.4, pode-se verificar que, comparativamente com o algoritmo Soma das Distâncias, o número de ocorrências nas distâncias de erro mais elevadas diminuiu, passando a maioria a ocorrer entre os 0 e os 3 metros, e o erro máximo está na casa dos 7 metros. Estes resultados já são uma melhoria relativamente ao algoritmo anterior.

No entanto, é de notar que a distância de erro mínima obtida neste algoritmo é aproximadamente igual (menor 2 milímetros) à obtida no algoritmo soma das distâncias, e ocorreu na posição com o ID 24. A distância de erro máxima, como já foi referido, diminuiu mais de 2 metros, sendo, neste caso, 7,82 metros (ID 13).

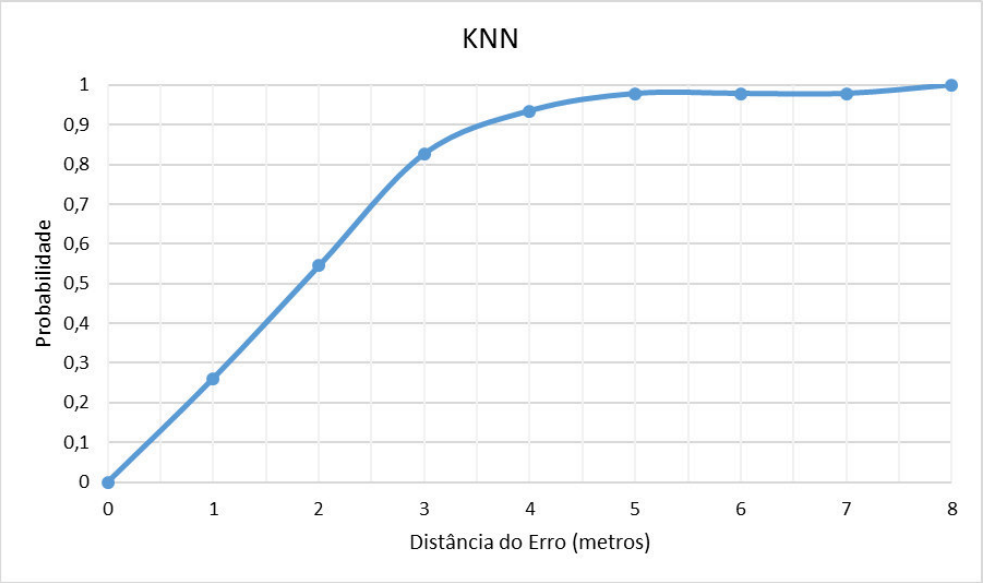


FIGURA 6.5: Distribuição da probabilidade acumulada da distância do erro

Para melhor compreensão dos resultados obtidos com o algoritmo KNN, foi construído um gráfico com a distribuição da probabilidade acumulada, representado na Figura 6.5. Analisando este gráfico, conclui-se que cerca de 54% dos testes efetuados obtiveram um erro inferior a 2 metros e, em 93% dos casos, a distância de erro foi inferior a 4 metros. É importante notar que estes valores já são melhores que os valores obtidos no algoritmo soma das distâncias.

A distância média obtida nos testes efetuados com o KNN foi de 2,04 metros, ou seja, o erro médio foi melhorado em 29,16%, em relação ao algoritmo soma das distâncias. Este era o resultado esperado, uma vez que o algoritmo soma das distâncias é um algoritmo bastante simples, cujas posições calculadas para os dispositivos estão limitadas ao leque de posições dos pontos de referência do cenário.

6.2.3 *Weighted K Nearest Neighbor*

Os testes realizados a este algoritmo seguem o mesmo modelo dos anteriores, ou seja, é realizada a localização nos mesmos 46 pontos escolhidos e a verifica-se se o algoritmo providencia a localização correta. O objetivo deste algoritmo é melhorar os resultados

em relação aos obtidos no KNN, através da atribuição de pesos em conformidade com valor da distância euclidiana, entre o dispositivo a localizar e os K vizinhos escolhidos.

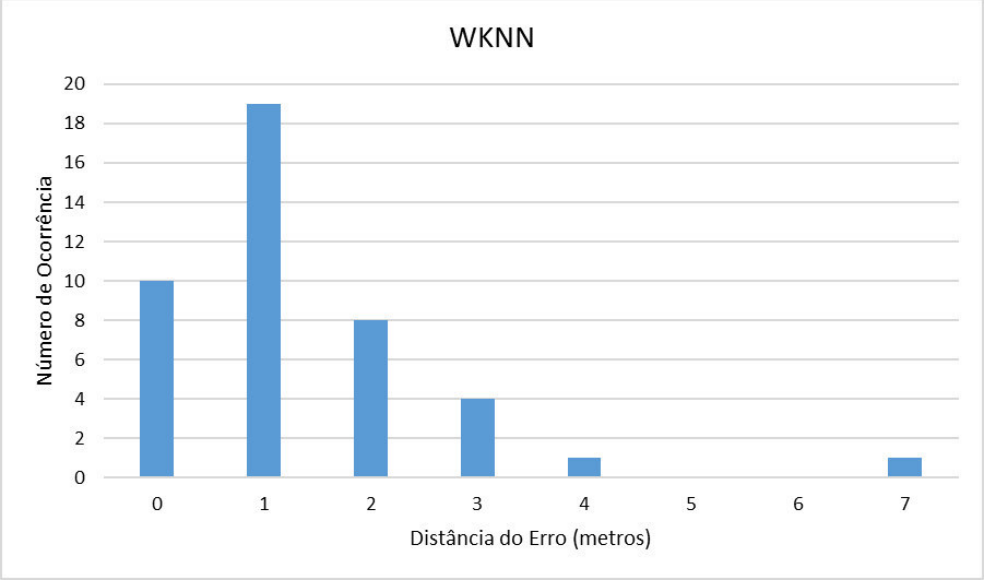


FIGURA 6.6: Distância do erro no algoritmo WKNN

Em anexo, na Tabela A.3, encontram-se representados os resultados obtidos. De forma a sumariar e melhor compreender os resultados, foi construído o histograma ilustrado na Figura 6.6. Neste algoritmo, o número de ocorrências com distância de erro na casa de 1 metro aumenta bastante em relação aos algoritmos anteriores, e diminui em distâncias superiores a 2 metros. Por outro lado, o número de ocorrências para valores abaixo de 1 metro também diminui. A média é ligeiramente inferior ao KNN, sendo 2,00 metros.

O gráfico ilustrado na Figura 6.7 representa a distribuição da probabilidade acumulada do algoritmo WKNN. No entanto, como ilustrado no gráfico, apenas foi obtida uma melhoria marginal em comparação ao algoritmo anterior. Neste caso, verifica-se que em de metade das ocasiões, a distância de erro obtida é menor do que 1,60 metros, e que em cerca de 90 % dos casos esta distância é inferior a 3,50 metros. No total, o valor da distância média do erro foi melhorado em cerca de 1,96%, em relação ao KNN.

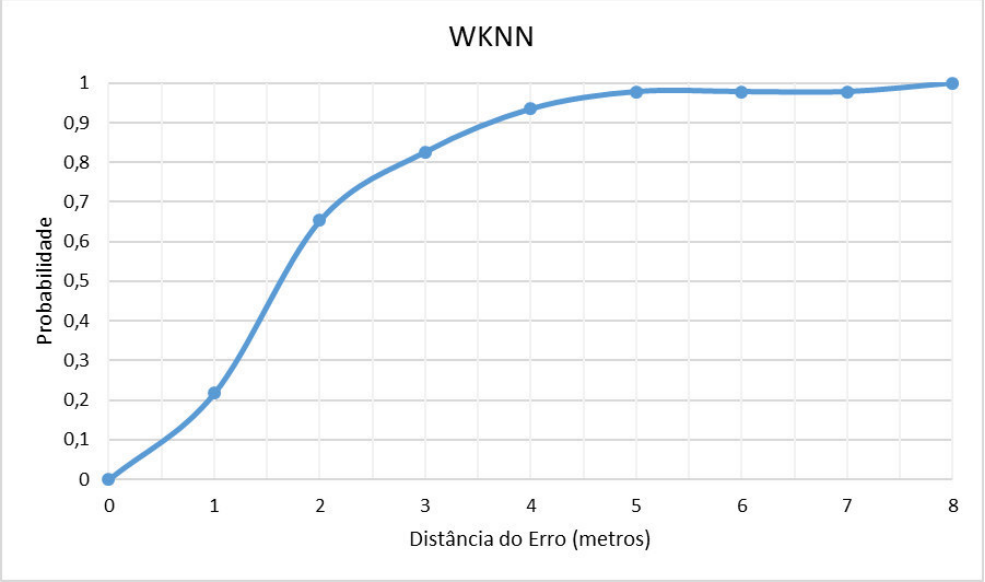


FIGURA 6.7: Distribuição da probabilidade acumulada da distância do erro

6.2.4 *Historical K Nearest Neighbor*

O HKNN é o primeiro algoritmo testado que utiliza informação histórica para cálculo das localizações. Portanto, espera-se que os resultados deste algoritmo tenham uma melhoria considerável em relação aos algoritmos testados anteriormente. Então, nos testes efetuados ao algoritmo HKNN, obtiveram-se os resultados presentes na Tabela A.4, em anexo, e na Figura 6.8.

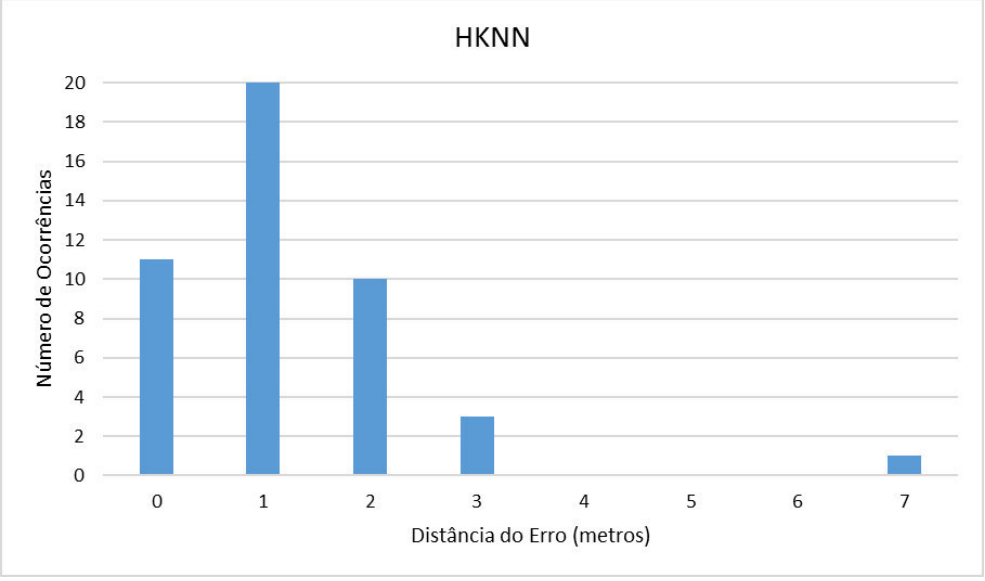


FIGURA 6.8: Distância do erro no algoritmo HNN

Depois de analisado o histograma da Figura 6.8, conclui-se que os resultados são, apenas, um pouco mais satisfatórios do que os obtidos no algoritmo WKNN devido, principalmente, a um ligeiro aumento do número de ocorrências entre os 0 e os 3 metros. A existência de, apenas, um ligeiro aumento, deve-se ao facto de o algoritmo HKNN ser aplicado em apenas alguns casos. Na maioria dos casos, as posições calculadas pelo WKNN não satisfazem as regras para que o HKNN seja executado. Isto acontece devido à distância entre a posição calculada pelo WKNN e a última posição do instante de tempo anterior não ultrapassar 1,5 metros, não serem encontradas posições anteriores ou não existiram K vizinhos mais próximos no espaço de interceção das circunferências.

No entanto, nos casos em que o HKNN é aplicado, este nem sempre melhora a exatidão das localizações, comparativamente aos restantes algoritmos abordados anteriormente. Num total de 46 posições testadas, o algoritmo HKNN foi aplicado 26 vezes, melhorando a exatidão em 19 posições e piorando por 7 ocasiões, relativamente ao WKNN. Ou seja, este algoritmo, quando implementado, conseguiu melhorar os resultados em, aproximadamente, 73% das posições testadas. Nos seguintes tópicos, e na Tabela 6.3, encontram-se diversos dados estatísticos que ajudam a compreender os resultados obtidos no HKNN:

- Número total de posições testadas: 46
- Número de vezes em que o HKNN foi aplicado: 26
- Número de vezes em que o HKNN não foi aplicado: 20
- Número de melhorias : 19
- Número de vezes em que piorou: 7

Na Tabela 6.3 estão representadas as cinco posições onde foram obtidas maiores diferenças entre a distância de erro obtida no algoritmo HKNN e no WKNN. Facilmente se verifica que a maioria das diferenças são resultantes de melhorias, embora na posição com o ID 22 o HKNN piorou bastante o resultado obtido anteriormente.

No total das 46 posições testadas, foi obtida uma distância média do erro de 1,79 metros, isto significa que houve uma melhoria de 10,50% em relação ao algoritmo WKNN. Já as distâncias do erro máximas e mínimas foram, respetivamente, 7,47 metros (ID 13) e 0,08 metros (ID 26).

ID	WKNN (metros)	HKNN (metros)	Diferença (metros)
4	4,31	2,44	-1,86
7	3,57	1,69	-1,88
12	2,43	0,52	-1,90
14	0,65	2,21	+1,56
15	3,90	0,87	-3,03
22	0,71	3,75	+3,04

TABELA 6.3: Comparação dos resultados entre os algoritmos WKNN e HKNN

Na Figura 6.9 está ilustrado o gráfico da distribuição de probabilidades acumuladas do algoritmo HKNN. Nestes resultados já são mais evidentes as melhorias deste algoritmo em relação aos anteriores, mais propriamente entre os 2 e 5 metros. Com a utilização do HKNN, aproximadamente 70% dos resultados possuem uma distância de erro menor do que 2 metros, e 90% inferior a 3 metros. Tal como no KNN e no WKNN, a probabilidade acumulada neste algoritmo atinge os 100%, apenas, aos 8 metros. Isto significa que, nos algoritmos referidos, o erro máximo obtido situa-se entre os 7 e os 8 metros (7,47m).

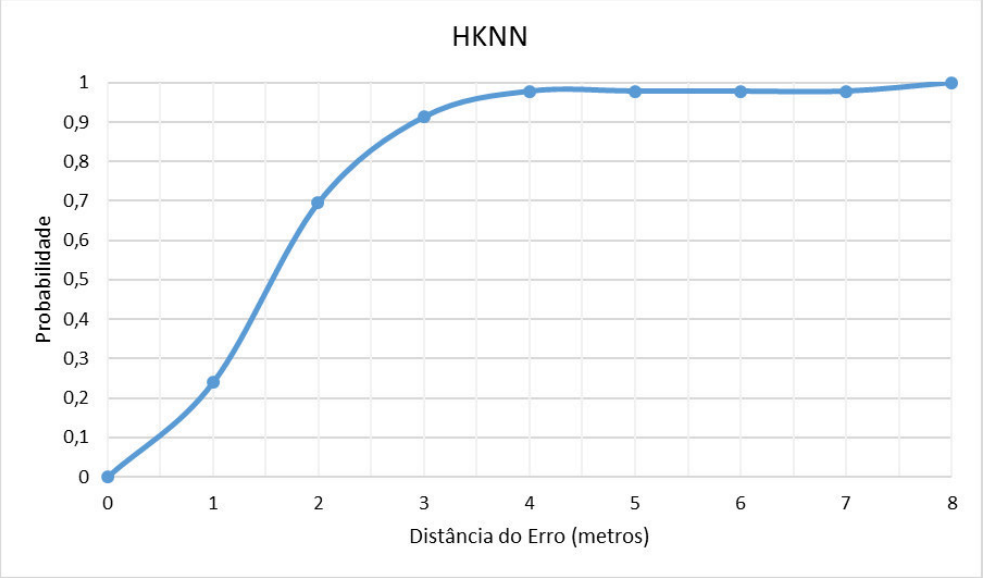


FIGURA 6.9: Distribuição da probabilidade acumulada da distância do erro

6.2.5 *Velocity Aware K Nearest Neighbor*

O que se pretende com a utilização deste algoritmo é obter resultados semelhantes aos do algoritmo HKNN, no entanto, com utilização de menos cálculos nos processos

de localização, como já explicado anteriormente. Para a realização dos testes a este algoritmo, foi definido o valor 1,5 m/s como a velocidade necessária para que o VAKNN seja executado. Isto quer dizer que, quando a velocidade média de deslocamento de um dispositivo entre a posição calculada num certo instante de tempo (pelo algoritmo WKNN) e a sua posição anterior for superior a 1,5 m/s, o algoritmo VAKNN é executado. Este algoritmo utiliza os dados históricos do utilizador, de forma a melhorar os resultados obtidos com a utilização de um algoritmo mais comum. Nos testes efetuados usou-se o algoritmo WKNN.

Assim sendo, para este algoritmo realizaram-se os mesmos testes que já haviam sido efetuados nos algoritmos anteriores. Os resultados aqui obtidos estão apresentados na Tabela A.5, em anexo, e na Figura 6.10.

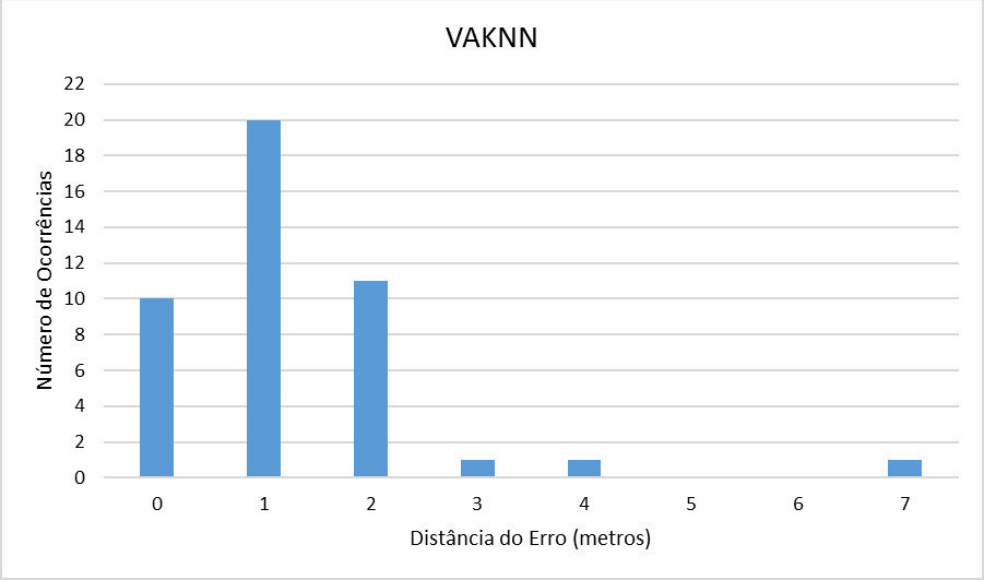


FIGURA 6.10: Distância do erro no algoritmo VAKNN

Com a utilização deste algoritmo, obteve-se a menor distância de erro mínima, em relação aos algoritmos anteriormente testados. Este resultado foi obtido no cálculo de posicionamento na posição com ID 26 com uma distância de erro de 4 milímetros. A distância de erro máxima continua a mesma dos algoritmos WKNN e HKNN, 7,47 metros, o que significa que o VAKNN não conseguiu, também, melhorar o resultado nesta posição (ID 13). A distância média do erro foi de 1,83 metros. É possível verificar, pelo histograma representado na Figura 6.10, que a maioria das distâncias do erro são cerca 1 metro, e que existem poucas situações em que a distância do erro é superior a 2 metros.

Os resultados obtidos em cada uma das 46 posições testadas, com os algoritmos HKNN e VAKNN, não sofreram grandes diferenças. No final dos testes, verifica-se que o VAKNN melhorou os resultados em 16 posições, comparativamente ao HKNN, e piorou em 11. Em 19 posições o resultado foi o mesmo, o que significa que os algoritmos com informação histórica não foram aplicados por não satisfazerem os requisitos necessários, e o resultado do cálculo é o mesmo, dado pelo WKNN. Apesar de, nos testes com a utilização do algoritmo VAKNN, o número de posições em que os resultados foram melhorados ser superior ao número em que foram piorados, a distância média do erro é superior à do algoritmo HKNN. Embora seja um valor ligeiramente superior, isto reflete que os resultados que foram melhorados foram-no superficialmente, enquanto que nos resultados onde o erro foi maior, este foi mais consistente.

A maior diferença de resultados entre estes dois algoritmos ocorreu na posição com o ID 15, onde com a utilização do HKNN obteve-se um resultado cerca de 1,97 metros mais satisfatório. Na Tabela 6.4 estão identificadas as posições onde foram registadas as maiores divergências entre estes dois algoritmos baseados em dados históricos.

ID	**HKNN (metros)**	**VAKNN (metros)**	**Diferença (metros)**
1	1,50	3,44	+1,94
2	1,66	2,84	+1,18
7	1,69	2,43	+0,74
15	0,87	2,85	+1,97
22	3,75	1,81	-1,94
37	1,86	0,50	-1,35

Tabela 6.4: Maiores desconformidades entre os algoritmos HKNN e VAKNN

Tal como no HKNN, neste algoritmo também existe um fator que impede o melhoramento substancial dos resultados obtidos, que consiste na dependência em que ambos os algoritmos possuem em relação ao algoritmo inicial utilizado, neste caso o WKNN. Como é sabido, o VAKNN e o HKNN apenas são aplicados quando o WKNN tem resultados que não satisfaçam determinados requisitos, ou seja, em muitos casos a localização será dada pelo WKNN.

Na Figura 6.11 está ilustrado o gráfico da distribuição de probabilidades acumuladas do algoritmo VAKNN. Na imagem, verifica-se facilmente que a distribuição da probabilidade acumulada é muito idêntica à obtida no algoritmo HKNN, sendo que, neste

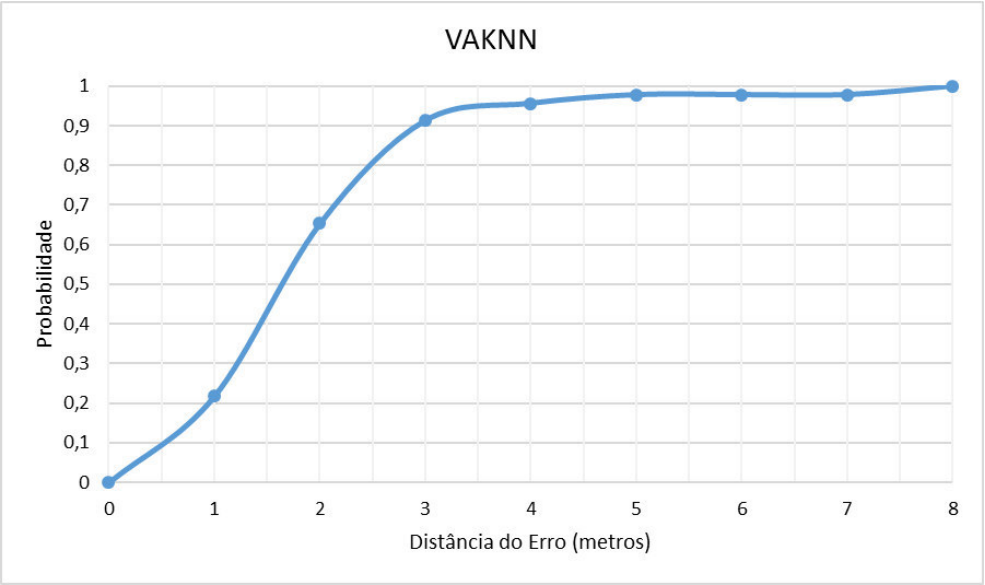


FIGURA 6.11: Distribuição da probabilidade acumulada da distância do erro

último é ligeiramente superior até aos 4 metros de distância do erro.

Desta forma, pelos resultados obtidos e descritos anteriormente, conclui-se que foi obtido o pretendido com a utilização deste algoritmo. O VAKNN apresenta melhorias em relação aos algoritmos sem dados históricos, quer no valor da distância mínima do erro obtida, quer na distância média do erro. Em relação ao algoritmo HKNN baseado, também, em informação histórica, com a utilização do VAKNN obteve-se uma média de distância de erro superior por 4 centímetros, ou seja, os resultados pioraram, apenas, 2,23% em relação ao algoritmo HKNN, embora sendo um algoritmo que consome menos recursos computacionais.

6.2.6 *Restricted K Nearest Neighbor*

O RKNN é um algoritmo de localização que faz uso da informação histórica em todos os cálculos de posicionamento realizados. De forma a avaliar o seu desempenho, em comparação com outros algoritmos, foram realizados os mesmos testes que haviam sido feitos aos restantes algoritmos implementados. Os resultados detalhados obtidos em cada posição encontram-se na Tabela A.6, presente em anexo. Igualmente, foi construído um histograma com os resultados obtidos neste algoritmo, e que se encontra ilustrado na Figura 6.12.

Comparativamente ao algoritmo com melhores resultados sem informação histórica, o WKNN, o RKNN obteve, nas 46 posições testadas, piores resultados em 26 posições, e aumentou a exatidão da localização em 20 dessas posições. Como já referido, este

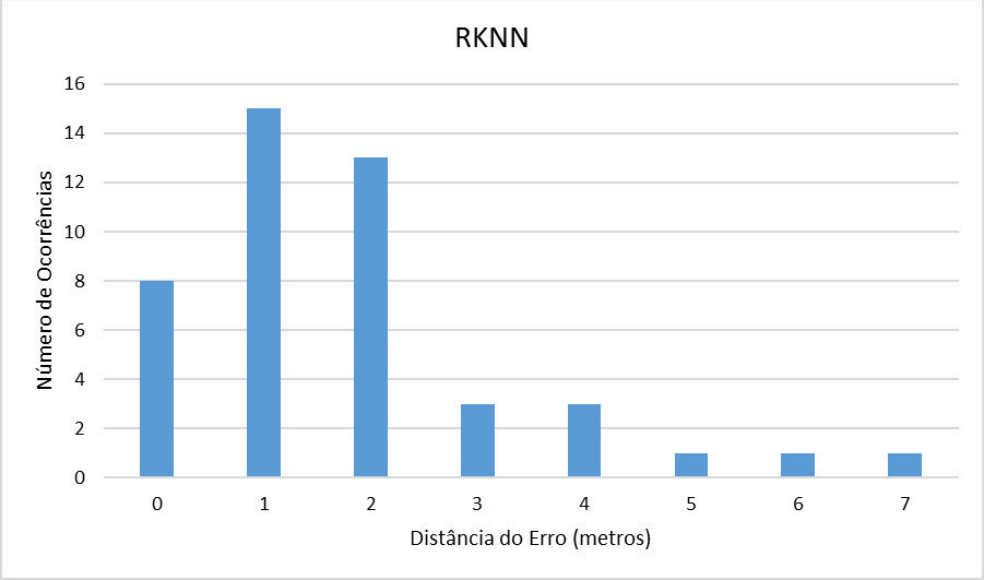


FIGURA 6.12: Distância do erro no algoritmo RKNN

algoritmo, ao contrário do HKNN e do VAKNN, não está dependente da utilização de um algoritmo não histórico. É devido a este factor que, nas 46 posições testadas, não se obteve nenhum resultado igual ao obtido no algoritmo sem dados históricos (WKNN).

Comparando com o HKNN, o algoritmo que até este ponto possui melhores resultados ao nível da exatidão média, o RKNN apresenta resultados menos satisfatórios. Este algoritmo aumentou a exatidão em apenas 17 das 46 posições testadas, piorando os resultados em 29 situações, em relação ao HKNN.

A média da distância de erro obtida foi de 2,43 metros. Este é um valor superior aos obtidos nos algoritmos anteriores. No entanto, com este algoritmo conseguiu-se melhorar alguns resultados em posições onde os algoritmos anteriores falhavam drasticamente. Como exemplo disso, é a posição com o ID 16, onde a distância de erro foi reduzida cerca de 3,46 metros e 2,41 metros, comparando com os algoritmos WKNN e HKNN, respetivamente. Esta redução do erro ocorreu devido à correção da escolha dos K vizinhos mais próximos, em que nos algoritmos WKNN e HKNN estavam a ser escolhidos vizinhos com distâncias muito afastadas entre si. Na Tabela 6.5 encontram-se alguns registos das diferenças obtidas entre o algoritmo sem informação histórica com melhores resultados, e os algoritmos HKNN e RKNN, estes baseados em dados históricos.

No caso das posições com o ID 22 e 29, a distância de erro aumentou bastante. Este aumento da distância do erro deve-se ao facto de, nestas situações, o utilizador movimentar-se muito rapidamente na mesma direção e, na realidade, este saiu mesmo do espaço onde se encontram os possíveis K vizinhos.

ID	WKNN (metros)	HKNN (metros)	RKNN (metros)	Diferença para o WKNN	Diferença para o HKNN
7	3,57	1,69	1,01	-2,56	-0,68
13	7,47	7,47	5,60	-1,86	-1,86
16	4,74	3,69	1,28	-3,46	-2,41
17	2,78	2,78	1,47	-1,30	-1,30
22	0,71	3,75	6,95	+6,23	+3,19
29	2,35	2,35	7,92	+5,57	+5,57

TABELA 6.5: Comparação do RKNN com os algoritmos WKNN e HKNN

Na Figura 6.13 está ilustrado o gráfico da distribuição de probabilidades acumuladas do algoritmo RKNN. No gráfico obtido verifica-se que, como já foi referido, os resultados obtidos estão um pouco à quem dos alcançados nos algoritmos anteriores. Exatamente em metade das experiências com este algoritmo obteve-se um erro inferior a 2 metros. Em apenas 90% dos casos, a distância de erro é inferior a 4,50 metros, e atingindo os 100% nos 8 metros.

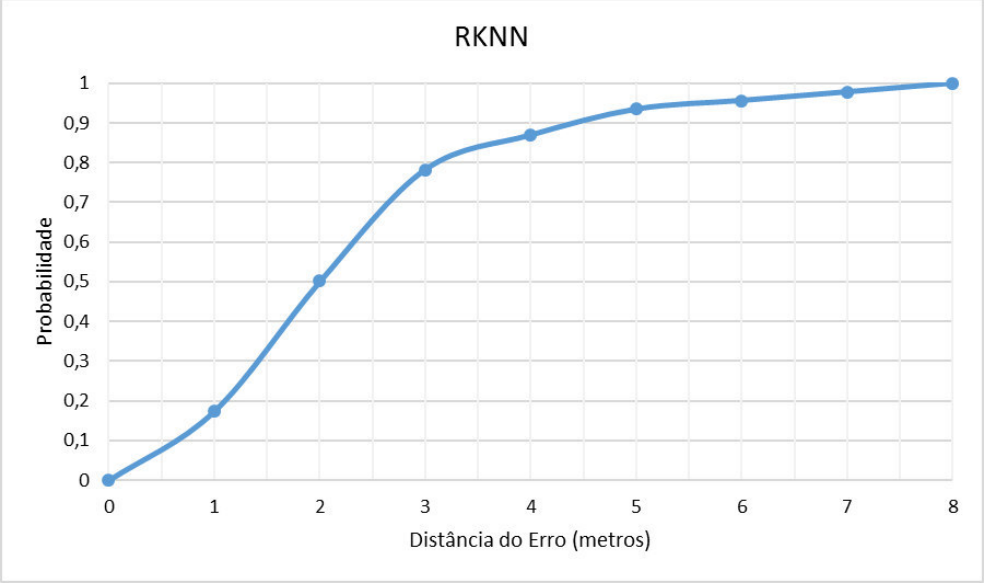


FIGURA 6.13: Distribuição da probabilidade acumulada da distância do erro

Como foi explicado, este algoritmo de localização, que possui um conceito diferente relativamente aos outros algoritmos baseados em dados históricos implementados (HKNN e VAKNN), possui, também, resultados bem diferentes dos anteriores.

6.2.7 *Feedback* de utilizadores

Existem diversos fatores, já descritos anteriormente, capazes de degradar os sinais rádio e, consequentemente, diminuir a exatidão dos algoritmos de localização. Estes fatores acontecem, principalmente, devido a alterações momentâneas nos ambientes de teste, como o abrir ou fechar de portas ou movimentos de pessoas. Como consequência, estes fatores podem levar a que existam certos locais no cenário onde a exatidão de localização seja bem mais inferior que as restantes.

De forma a avalizar a existência de padrões nas distâncias de erro no cenário de teste do *dataset* utilizado, construiu-se um mapa de calor com as distribuições das distâncias de erro ao longo do cenário, ilustrado na Figura 6.14. Para a construção do gráfico utilizaram-se os valores obtidos pelo algoritmo de localização HKNN, com o valor K definido como 3.

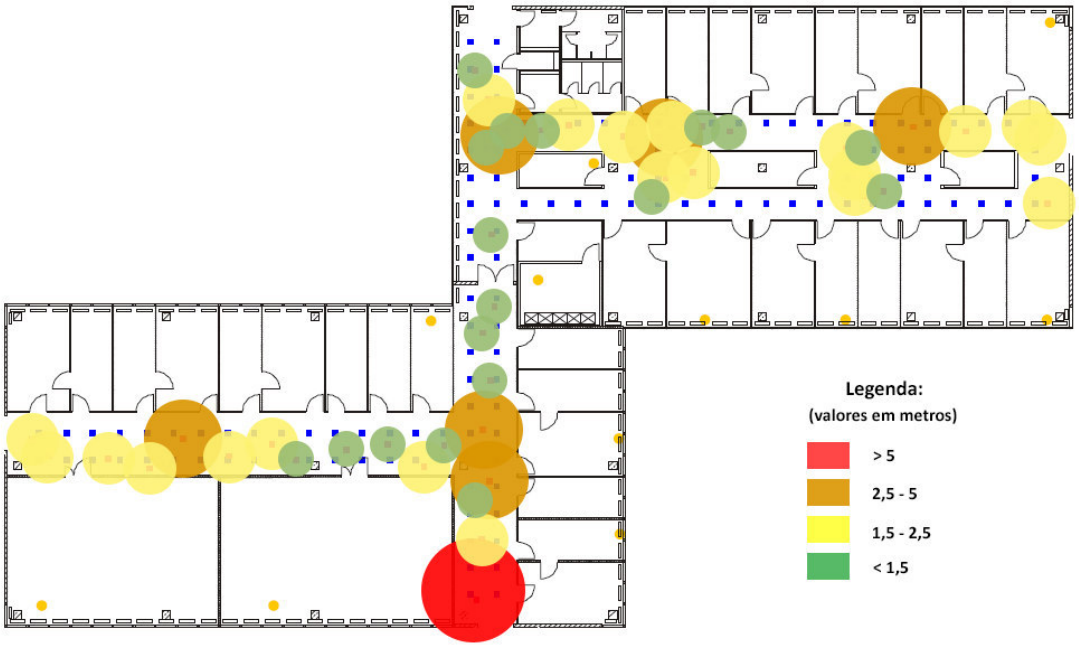


FIGURA 6.14: Mapa de calor com as distâncias de erro obtidas com o HKNN

Como era de esperar, é evidente a existência de um padrão na distribuição dos valores da distância de erro. Os erros mais baixos encontram-se bastante próximos do centro do mapa, visto que é a zona onde a qualidade do sinal recebido dos APs é melhor e existe um maior número de APs com cobertura. Os resultados menos satisfatórios ocorrem nas periferias do cenário, com clara evidência no limite inferior do corredor central. Nesta zona ocorreu um erro superior a 5 metros, posição 13, e dois entre os 2,5 e os 5 metros.

Considerando a Figura 6.14 e, em anexo, a Tabela A.4, assumiu-se essa informação como sendo o *feedback* dado por um utilizador.

6.2.7.1 Localização Recalculada

Como explicado anteriormente, este algoritmo apenas atua quando é indicado um *feedback* negativo por parte de um utilizador do sistema. De forma a simular-se essa informação no *dataset* utilizado, foi definido como o *feedback* do utilizador, a avaliação dos resultados obtidos no algoritmo HKNN. Quando a distância da posição devolvida calculada por este algoritmo é superior a um limite pré-definido pelo administrador, o sistema calcula novamente usando um método diferente.

Nos testes que serão seguidamente descritos, o limite foi definido em 1,50 metros. Este valor é usado apenas para simular possíveis *feedbacks* que podem ser dados por utilizadores em casos reais, onde estes, nos casos em que a posição não está correta, indicam em qual das áreas onde realmente estão posicionados. Ou seja, sempre que o erro obtido no HKNN for superior a 1,50 metros, é assumido que o utilizador atribui um *feedback* negativo, o que corresponde às posições assinaladas a amarelo, laranja e vermelho da Figura 6.14.

Outro aspeto importante para os testes a este algoritmo, é a divisão do cenário em várias áreas, que englobam o conjunto de APs que forneçam os melhores resultados de localização para cada área. Após a realização de alguns testes auxiliares, conclui-se que nem sempre os APs que estão mais perto são os que têm melhor potência de sinal recebida mas, no entanto, os APs que possuem piores valores de RSS não podem ser totalmente desprezados, pois os seu valores interferem com o resultado das localizações.

O melhor conjunto de APs observado (com melhor potência de sinal recebida) nem sempre é o conjunto ótimo. Cada ponto do cenário de teste tem um conjunto de APs que, quando utilizado pelos algoritmos de localização, dão como resultado uma solução ótima. Isto poderia ser realizado através de um algoritmo de aprendizagem, que usasse os dados recebidos em inúmeras posições dos cenários, e apresentasse a melhor solução para essas posições, agregando-as e dividindo, de forma o dinâmica, o cenário em diferentes áreas. No entanto, chegar a essa solução é um problema bastante complexo, que não foi abordado neste trabalho.

Para testar este algoritmo e dividir o cenário em áreas que levem a melhores resultados, identificou-se as zonas de maior erro e os APs que oferecem melhores resultados

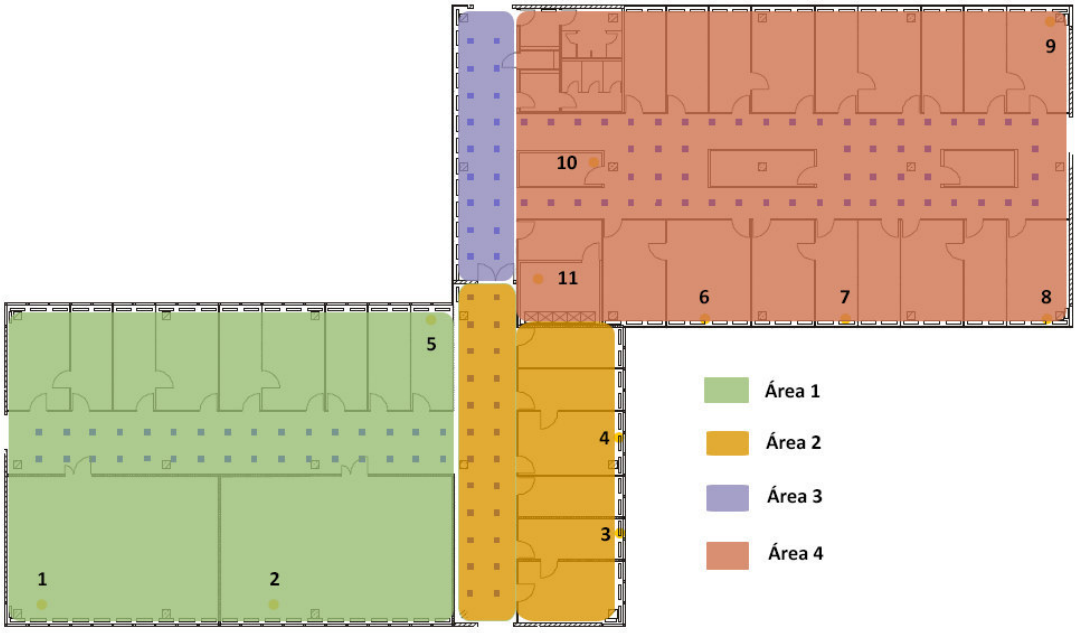


FIGURA 6.15: Divisão do cenário em quatro áreas distintas

nessas zonas. A solução encontrada foi dividir o cenário em quatro áreas distintas, representadas na Figura 6.15. O conjunto de APs pertencentes a cada uma das quatro áreas estão representados na Tabela 6.6. O AP que se encontra localizado no piso inferior, e não está representado no mapa, pertence ao conjunto de APs das áreas 2, 3 e 4.

Área	**APs a considerar**
Área 1	1, 2, 3, 4 e 5
Área 2	1, 2, 3, 4, 5, 10 e 11
Área 3	1, 2, 6, 7, 10 e 11
Área 4	6, 7, 8, 9, 10 e 11

TABELA 6.6: APs considerados em cada área

Definidas as áreas do cenário e os APs que as englobam, procedeu-se à realização dos respetivos testes de posicionamento nas várias posições do *dataset*. No total das 46 posições testadas, o algoritmo Localização Recalculada foi aplicado por 28 vezes. Das 28 posições em que se assumiu *feedback* por parte do utilizador, conseguiu-se diminuir a distância do erro 23 vezes. Na Tabela 6.7, encontram-se os resultados onde foram obtidas maiores diferenças entre os algoritmos HKNN e o Localização Recalculada. Todos os valores presentes estão em metros.

ID	Distância do erro HKNN	Erro com feedback	ID	Distância do erro HKNN	Erro com feedback
5	2,77	1,12	16	3,69	0,72
7	1,69	0,58	17	2,78	0,38
13	7,47	3,86	22	3,75	1,62
14	2,21	0,53	31	3,74	2,28

TABELA 6.7: Comparação dos erros obtidos com os algoritmos HKNN e Localização Recalculada

Em algoritmos anteriormente testados, a posição 13 era onde se obtinha maior erro, e mesmo com a utilização de algoritmos baseados no histórico, não foi possível obter-se um resultado mais satisfatório. Com o uso do *feedback* do utilizador, nesta mesma posição, a distância do erro foi diminuída para os 3,86 metros. Ainda assim, não é um valor aceitável. No entanto, como vai ser criado um novo RP nas posições em que foram atribuídos *feedbacks* negativos e, após as correções efetuadas, atribuídos *feedbacks* positivos, o resultado para esta posição, ao longo do tempo e em posições próximas, vai sendo melhorado.

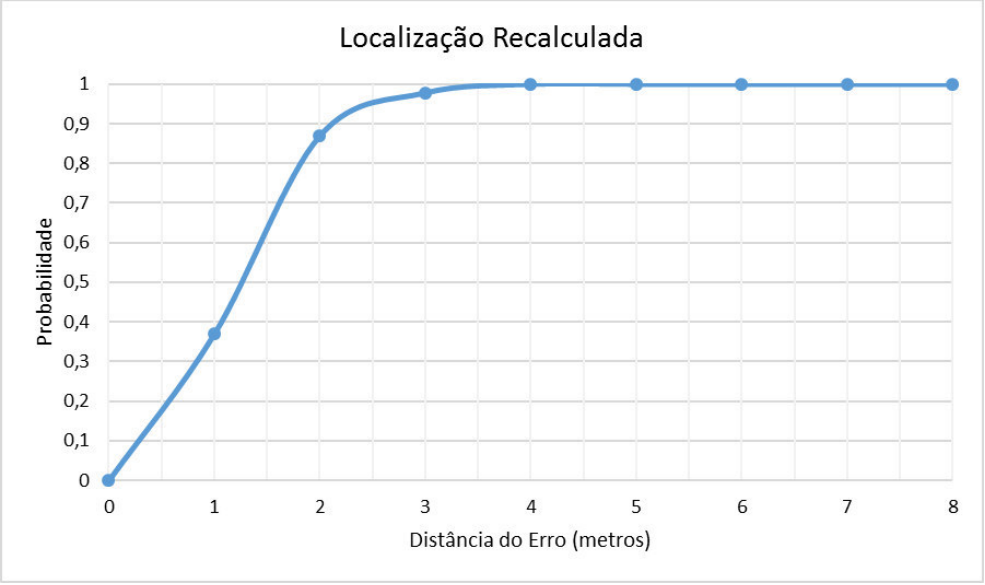


FIGURA 6.16: Distribuição da probabilidade acumulada da distância do erro

A distância média do erro obtida foi de 1,24 metros. Isto significa que houve uma melhoria de 30,72% em relação ao HKNN, o algoritmo com menor distância média do erro até então. A distância de erro mínima foi de 0,08m (posição 26) e a máxima de 3,86m (posição 13).

Na Figura 6.16 verifica-se que, com a utilização deste algoritmo, em metade dos testes realizados o erro obtido foi inferior a cerca de 1,25m e a 2,00m em quase 90% das experiências. Não houveram erros superiores a 4 metros, como também mostra a Figura 6.17. A maioria dos erros ocorreram com distâncias entre os 0 e 1 metro.

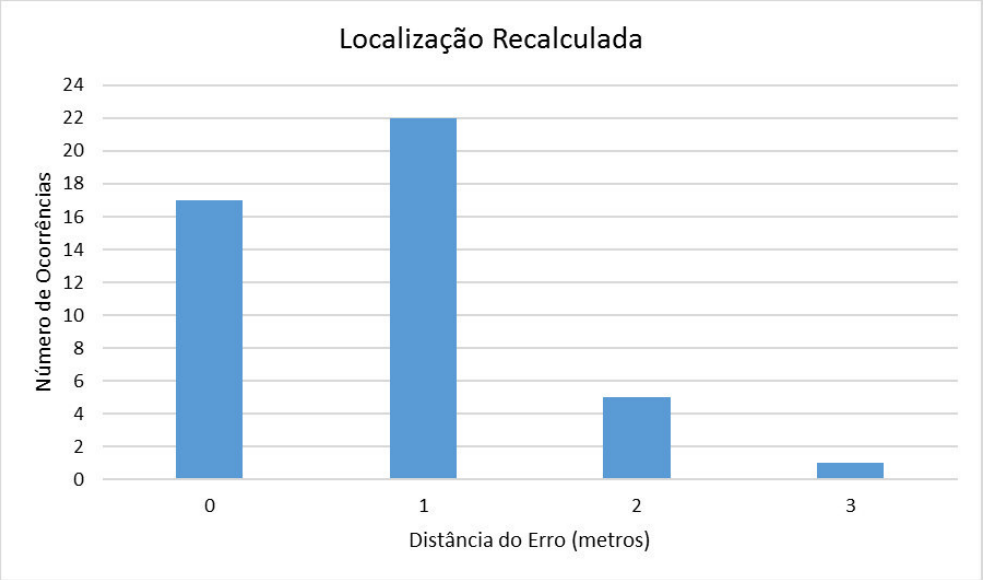


FIGURA 6.17: Distância do erro no algoritmo Localização Recalculada

A distância do erro obtida em cada posição, juntamente com as coordenadas calculadas, podem ser consultadas, em anexo, na Tabela A.7.

6.2.7.2 Eliminação de *Outliers*

Da mesma forma que foi realizado no algoritmo denominado de Localização Recalculada, simulou-se o *feedback* dos utilizadores como sendo a distância do erro obtida no algoritmo KNN com K=3. Foi escolhido este algoritmo para testes, pois é o mais simples e requer menos recursos computacionais, em comparação com os algoritmos baseados na escolha dos K vizinhos mais próximos.

Definiu-se que um *feedback* negativo é dado quando a distância do erro da posição calculada pelo KNN é igual ou superior a 1,50 metros. Nestas situações, o cálculo da localização dos dispositivos será efetuado através do algoritmo de eliminação de *outliers*. No KNN, algoritmo usado como base neste teste, foram obtidos erros iguais ou acima de 1,5 metros por 26 vezes. Os resultados obtidos em cada uma das posições estão representados na Tabela A.8, presente em anexo. Por outro lado, e de uma forma geral, na Figura 6.18 e na Figura 6.19, encontram-se, respetivamente, o gráfico da distribuição da probabilidade acumulada e o histograma dos resultados deste algoritmo.

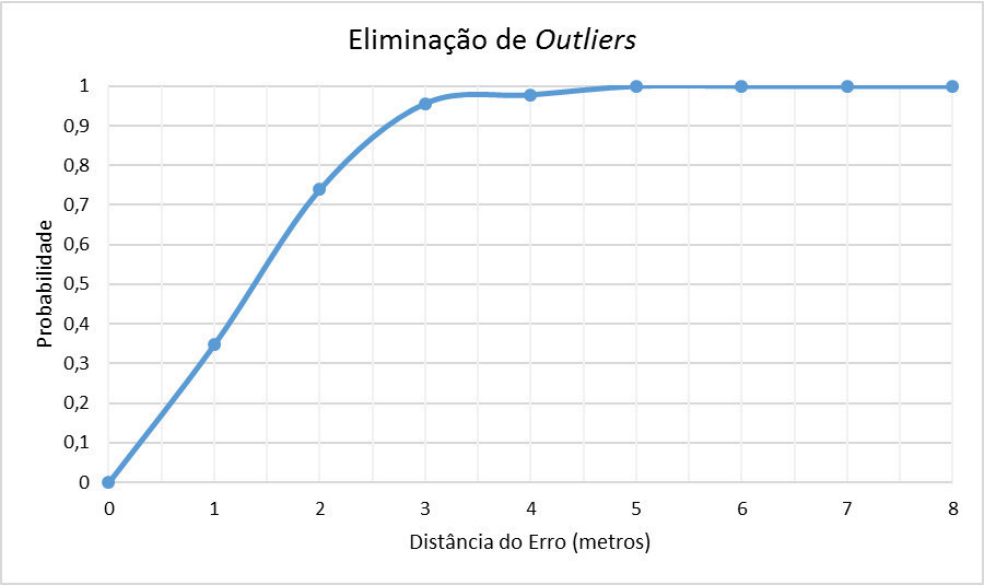


FIGURA 6.18: Distribuição da probabilidade acumulada da distância do erro

Analisados os resultados, neste algoritmo obteve-se uma distância de erro mínima de 0,32 metros, e a máxima de 4,22m na posição 13 do cenário. A distância média do erro foi de 1,52m, um valor bastante satisfatório quando comparado com o valor obtido pelo KNN.

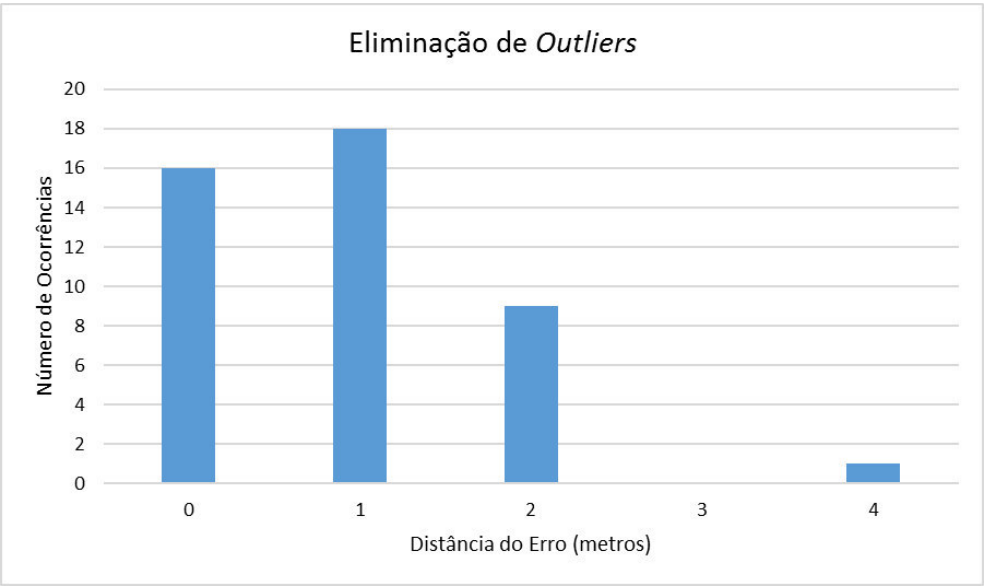


FIGURA 6.19: Distância do erro no algoritmo eliminação de *outliers*

No gráfico da distribuição da probabilidade acumulada deste algoritmo, verifica-se que, em pouco mais de 50% das posições testadas, o erro obtido foi inferior a 1,50m. Observou-se ainda que, em quase 90% dos casos, a distância do erro foi inferior a 2,50m.

Pela observação do histograma elaborado, conclui-se que os resultados são muito homogéneos. A maioria das distâncias de erro ocorridas foram da ordem os 0 e 1 metros, existindo apenas alguns casos de cerca de 2 metros e um de 4 metros (erro máximo).

6.3 Comparação dos algoritmos implementados

Para se concluir a compreensão dos resultados, obtidos durante os testes a todos os algoritmos implementados, nesta secção é apresentada uma comparação, aprofundada, entre eles.

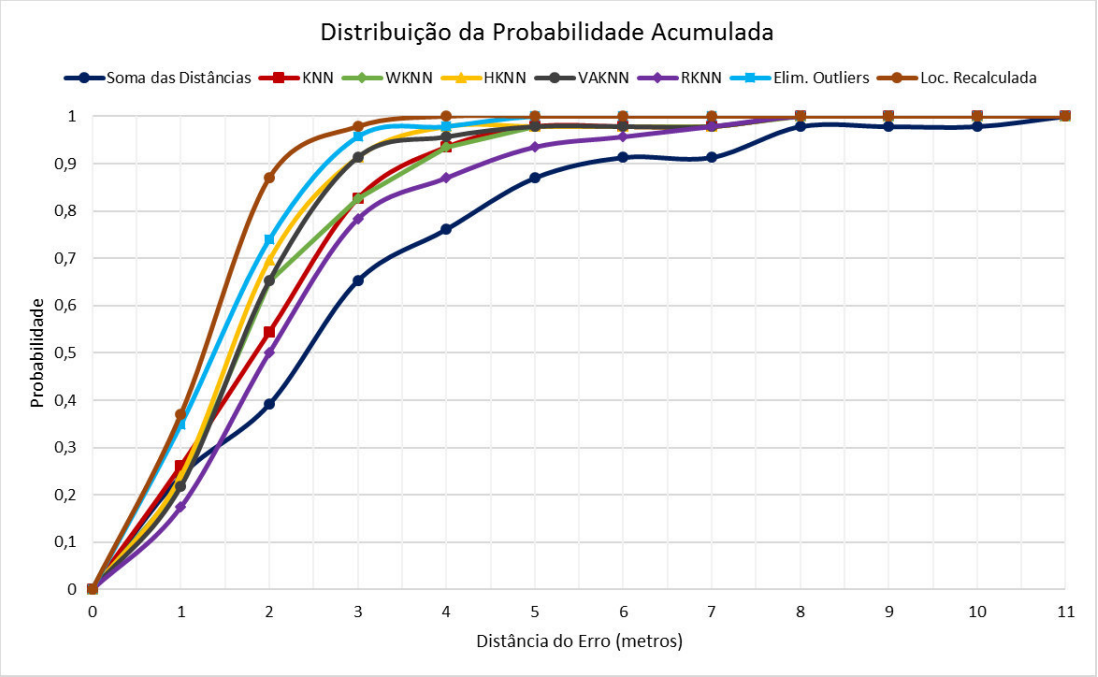


FIGURA 6.20: Comparação da distribuição da probabilidade acumulada entre os diferentes algoritmos

O gráfico ilustrado na Figura 6.20 representa, em conjunto, os valores da distribuição das probabilidades acumuladas dos algoritmos implementados. Em distâncias de erros inferiores a 1 metro, todos os algoritmos, com exceção dos que implementam *feedback* do utilizador, são bastante idênticos. Desde o inicio e ao longo de toda a escala, os algoritmos Eliminação de *Outliers* e Localização Recalculada possuem uma vantagem considerável, relativamente aos restantes algoritmos. O que possui melhores resultados é mesmo o algoritmo Localização Recalculada, o que era de esperar, pois este algoritmo baseia-se na solução ótima de cada área do cenário.

Ao analisar-se erros inferiores a 2 metros já são percetíveis maiores diferenças entre os vários algoritmos. Nota-se que os resultados obtidos no algoritmo Soma das Distâncias

são bastante inferiores dos demais algoritmos, sendo mesmo o único que não atinge pelo menos os 50%. Desde este ponto em diante, os resultados deste algoritmo ficam muito à quem dos restantes, atingindo os 100% apenas nos 11 metros (distância do erro máximo).

Entre o primeiro metro até aos 3 metros de distância do erro, também é visível que os algoritmos KNN, WKNN e RKNN possuem resultados menos satisfatórios do que os restantes.

A maior discrepância ocorre em erros inferiores a 3 metros, onde se observa que a diferença entre o algoritmo com melhores resultados e o que possui resultados menos satisfatórios é cerca de 47% nos 2 metros e 32% nos 3 metros.

Através do gráfico, facilmente se conclui que, no global, quando se obtém melhores resultados é com a utilização dos algoritmos baseados em informação histórica, mais propriamente o HKNN e o VAKNN, e nos algoritmos que fazem uso do *feedback* dos utilizadores.

Algoritmo	**K escolhido**	**Erro Médio (metros)**	**Erro Máximo (metros)**	**Erro Mínimo (metros)**
Soma das Distâncias	-	2,88	10,07	0,34
KNN	3	2,04	7,82	0.32
WKNN	3	2,00	7,47	0.32
HKNN	3	1,79	7,47	0.08
VAKNN	3	1,83	7,47	0.04
RKNN	3	2,43	7,92	0.51
Eliminação de *Outliers*	3	1,52	4,22	0.32
Localização Recalculada	3	1,24	3,86	0.08

Tabela 6.8: Comparação das distâncias de erro obtidas nos vários algoritmos

Outros dados, obtidos aquando da realização dos testes aos algoritmos implementados, encontram-se descritos na Tabela 6.8. Nesta tabela podem ser consultadas as distâncias médias do erro de cada algoritmo implementado no âmbito deste projeto, bem como as respetivas distâncias máximas e mínimas obtidas.

Com o melhor algoritmo baseado em dados históricos, o HKNN, consegui-se melhorar, em cerca de 33,9%, valor de distância média do erro obtido no projeto anterior [1] que antecedeu este (2,71m). Comparando com o melhor algoritmo baseado no *feedback*

dos utilizadores, a melhoria que se obteve foi cerca de 54,2%. Contudo, é importante referir que no projeto anterior foi utilizado um cenário diferente, a sala 0.10 do DI, como já referido anteriormente.

Quanto à precisão, verificada pelo gráfico da distribuição da probabilidade acumulada ilustrado na Figura 6.21, no projeto anterior, em cerca de 50% dos testes realizados obteve-se uma distância de erro inferior a 2,50 metros e em 90% inferior a 5,25 metros. Neste projeto, em metade dos testes realizados, a distância do erro foi inferior a cerca de 1,55m com o HKNN e inferior a 1,25m com o algoritmo Localização Recalculada. Em 90% dos casos o erro foi inferior a 3,00m e a 2,20m com os algoritmos HKNN e Localização Recalculada, respetivamente.

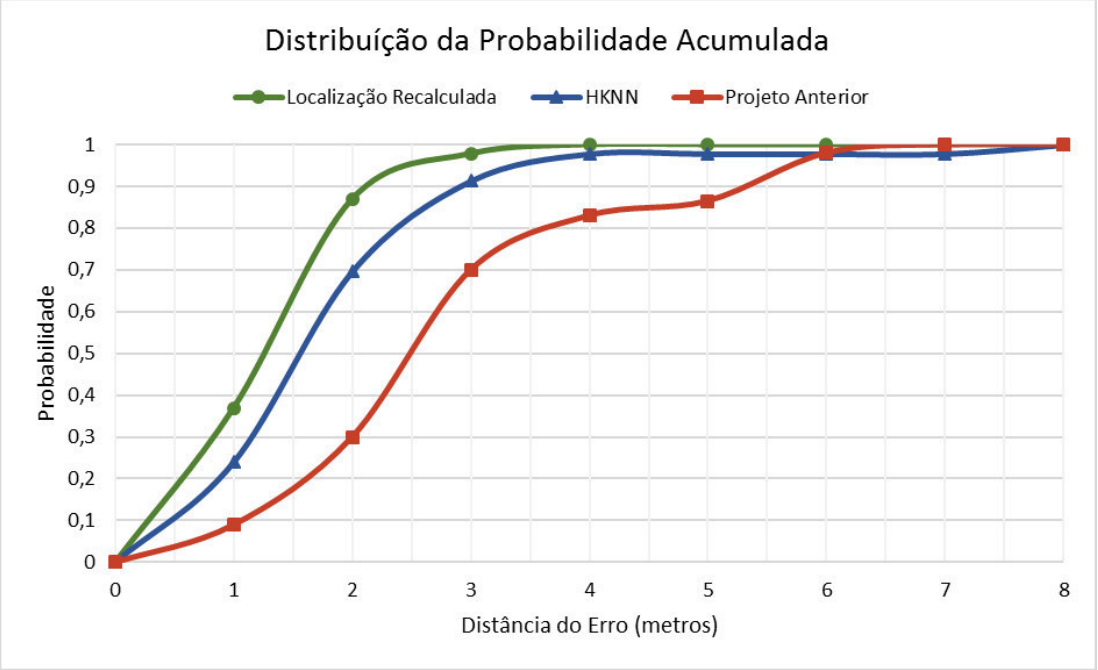


FIGURA 6.21: Comparação da distribuição da probabilidade acumulada entre os melhores algoritmos

Por outro lado, os autores do cenário de teste utilizado obtiveram, com a utilização de um algoritmo probabilístico, uma distância média do erro de 2,86 metros. Com a utilização dos algoritmos propostos no âmbito deste trabalho no mesmo cenário, melhorou-se esse resultado em 32,5% com o algoritmo HKNN, e 53,2% com o algoritmo Localização Recalculada.

Desta forma, conclui-se que, os resultados melhoraram consideravelmente através da utilização da informação histórica e do *feedback* dos utilizadores nos algoritmos de localização.

6.4 Variação dos valores de RSS ao longo do tempo

Testar a variação dos valores de RSS observados ao longo do tempo, consiste num teste complementar, relativamente aos efetuados, de forma a avaliar o funcionamento dos algoritmos. Este teste tem como principal objetivo analisar um fator que pode alterar a exatidão dos algoritmos de localização. O fator aqui analisado e testado é a variação dos valores de RSS na mesma posição ao longo do tempo, que pode ser provocado por movimentos de pessoas, abertura e fecho de portas e janelas, etc. Este fator influenciou a maneira como a base de dados e os algoritmos foram implementados. Em vez de, na fase *offline*, serem lidos apenas uma amostra de valores de RSS, é feita uma média de todas as amostras realizadas durante um instante de tempo. No caso do *dataset* utilizado, é feita a média das 110 amostras.

De forma a verificar esta situação, foram recolhidos durante um determinado instante de tempo e sempre na mesma posição, valores de RSS de diferentes APs. A posição escolhida para este teste tem as coordenadas locais (-23.90, -9.65). Com o mesmo dispositivo e durante 1:15 minutos (75 segundos), foram registados os valores de RSS recolhidos dos três APs mais próximos. O resultado obtido encontra-se ilustrado na Figura 6.22.

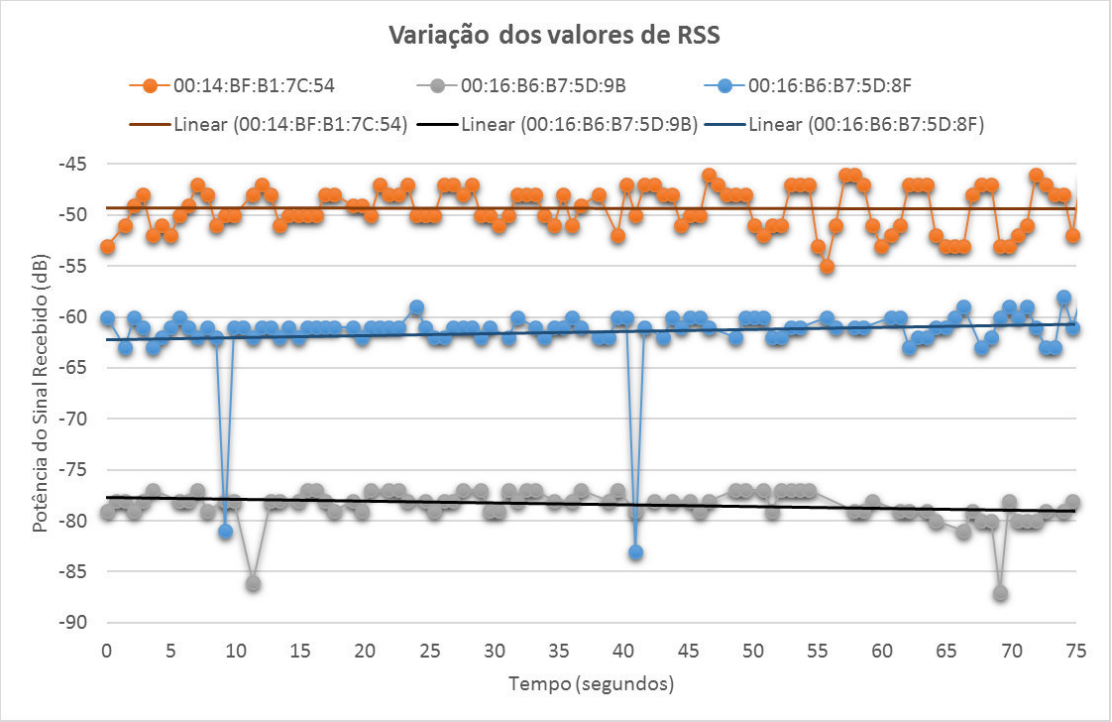


FIGURA 6.22: Variação dos valores de RSS na mesma posição ao longo do tempo

Depois de analisados, cuidadosamente, os resultados obtidos, pode-se verificar que, ao longo do tempo, os valores de RSS possuem uma ligeira discrepância e, em certos

instantes de tempo, pode mesmo existir uma grande variação, o que acontece, principalmente, nos dois APs com valores de RSS mais baixos.

No entanto, pode-se verificar pelas linhas de tendências lineares traçadas no mesmo gráfico, que os valores de RSS médios se mantêm ao longo do tempo. Foi com base neste efeito que resultou a solução implementada nos algoritmos de localização desenvolvidos, de forma a tentar ultrapassar-se o fator das variações existentes dos valores de RSS ao longo do tempo e na mesma posição durante a fase *offline*. Esta solução consiste em fazer-se várias observações dos valores de RSS dos APs e, posteriormente, utilizar a média desses mesmos valores.

Capítulo 7

Conclusão e trabalho futuro

Este trabalho foi desenvolvido no âmbito do Mestrado Integrado em Engenharia de Comunicações, na Universidade do Minho. O principal objetivo era encontrar formas de sumariar a informação histórica e o *feedback* dos utilizadores, de maneira a utilizar essa informação para melhorar os resultados dos algoritmos de localização, na sequência de um projeto anterior.

A localização dentro de edifícios, hoje em dia, trata-se de uma área com muito potencial e objeto de grande apreciação. O estudo do estado da arte dos sistemas e algoritmos de localização foi bastante valioso, pelo que os capítulos que o descrevem são muito importantes. Entre os vários sistemas estudados, destaca-se o RSS *Fingerprinting* baseado em dados históricos, proposto por Khodayari, Maleki e Hamedi. Eles apresentaram um algoritmo baseado em dados históricos bastante robusto, que serviu como ponto de partida para alguns algoritmos propostos neste projeto.

Dado que este projeto surgiu no seguimento de um anterior, foi necessário estudar e assimilar o funcionamento do mesmo. Um dos objetivos era melhorar os resultados dos algoritmos propostos nesse projeto anterior e integrar os novos algoritmos no sistema de localização já desenvolvido. No entanto, este sistema estava limitado ao cenário da sala 0.10 do Departamento de Informática da Universidade do Minho. Inicialmente, ainda se realizaram alguns testes nesse cenário, mas chegou-se à conclusão que a sala era demasiado pequena e não era, de todo, o cenário ideal para se testar os algoritmos de modo rigoroso.

Como consequência de se utilizar um cenário de testes diferente do projeto anterior, optou-se por criar uma aplicação de teste, onde é possível simular estimativas de posicionamento, e por desenvolver de raiz todos os algoritmos implementados, de acordo

com a nova base de dados, também implementada. Desta forma, não foi necessário assimilar nem utilizar o código desenvolvido no projeto anterior nem implementar os novos algoritmos no protótipo existente. Todavia, o projeto anterior não deixou de ser importante neste trabalho, pois os algoritmos implementados utilizam a mesma técnica de localização, isto é, são baseados em RSS *Fingerprinting*.

Encontrar disponível um *dataset* com os requisitos necessários, ou seja, com recolhas de valores RSS numa WLAN efetuadas por dispositivos móveis, numa fase de calibração (*offline*) e, também, na fase *online* foi uma tarefa complicada. Eram encontrados bastantes *datasets* mas, no entanto, a maioria era realizada com outras tecnologias, como por exemplo, RFID ou dispositivos *Bluetooth*.

O *dataset* utilizado foi o único encontrado com os devidos requisitos, e foi bastante útil, pois, ao contrário da sala utilizada no projeto anterior, estas recolhas foram efetuadas num edifício mais amplo, com maior número de APs e RPs e com paredes, portas e janelas entre os APs e os dispositivos localizáveis. Ainda assim, este *dataset* poderia ser mais vantajoso, dado que as medições efetuadas nas duas fases, apenas, foram realizadas nos corredores do edifício.

Os algoritmos e a aplicação de teste foram construídos de forma a permitir uma rápida mudança de cenário de teste. Quando se pretende realizar tal mudança é, apenas, necessário carregar os dados da fase *offline* na respetiva base de dados e introduzir os endereços MAC dos APs que pertencem à rede local. Além disso, a base de dados foi generalizada de forma a suportar qualquer cenário com número ilimitado de APs e RPs e, inclusive, um cenário a 3 dimensões.

Nos algoritmos que utilizam informação histórica ou *feedback* dos utilizadores, propostos neste trabalho, obteve-se uma distância média do erro bem inferior a 2 metros, com exceção do algoritmo RKNN. Comparativamente com os valores encontrados em trabalhos relacionados, é um valor bastante satisfatório.

7.1 Trabalho futuro

Como já referido, o tema da localização *indoor* é alvo de grande estudo e surgem, cada vez mais, novas soluções que visam melhorar os resultados das anteriores ou mesmo inovar a forma como se estimam as posições dos dispositivos móveis. Deste modo, é necessário estar a par destas novas abordagens, de forma a enriquecer os sistemas de localização.

Em termos de trabalho futuro pode ser implementado, nos vários algoritmos de localização, um mecanismo que considere o número de paredes existentes entre os APs e os dispositivos, de forma a compensar a atenuação nos sinais eletromagnéticos por elas causadas.

A curto prazo, poderia-se testar os algoritmos de localização num cenário em 3 dimensões, ou seja, num cenário com vários pisos. Esta medida pode ser rapidamente implementada, dado que a base de dados e os algoritmos desenvolvidos ficaram preparados para trabalharem em ambientes 3D.

Uma funcionalidade interessante que poderá ser implementada é a apresentação dos resultados em coordenadas reais, em vez das utilizadas coordenadas locais. Isto poderá ser implementado graças a uma biblioteca externa do Google Maps, que já contém mapas do interior de edifícios. Conhecendo-se as coordenadas reais dos cenários onde os sistemas de localização estão implementados, é possível converter as coordenadas locais em reais.

De igual modo, num futuro próximo, era importante desenvolver um sistema que, no algoritmo Localização Recalculada, determinasse dinamicamente as várias áreas constituídas pelo conjunto de APs que fornecem o melhor resultado.

Relativamente ao *feedback* dos utilizadores, poderia-se arranjar um mecanismo que disponibilize o *feedback* enviado por um utilizador para os restantes, em vez de, apenas, para si próprio. Este mecanismo tinha de aprovar as avaliações enviadas pelos utilizadores, por forma a evitar que *feedback* errado prejudique a localização de todos.

Numa fase posterior, era interessante implementar os algoritmos num sistema de localização real, e ser desenvolvida uma aplicação para dispositivos móveis (Android, iOS, Windows Phone) que apresentasse, através de uma interface gráfica, as suas localizações atuais, calculadas num servidor de localização pelos algoritmos propostos neste projeto.

Apêndice A

Resultados detalhados

Neste apêndice, encontram-se as tabelas com todas as posições do cenário testadas e os respetivos instantes de tempo, bem como o detalhe dos resultados aí obtidos. Nelas estão representadas a posição calculada pelos respetivos algoritmos, em coordenadas locais, e a distância de erro obtida em relação à real posição do dispositivo.

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-22, -10.75)	2.19	24	(2, 6.5)	2.86
2	(-20.5, -9.25)	2.81	25	(0.5, 12.5)	1.58
3	(-17.5, -9.25)	2.59	26	(0.5, 6.5)	0.89
4	(-10, -10.75)	7.36	27	(6.5, 8)	2.02
5	(-10, -10.75)	5.61	28	(6.5, 8)	0.51
6	(-8.5, -10.75)	4.40	29	(15.5, 3.5)	7.42
7	(-5.5, -10.75)	5.12	30	(11, 3.5)	0.53
8	(-8.5, -9.25)	1.65	31	(9.5, 3.5)	3.97
9	(-7, -10.75)	0.85	32	(8, 8)	3.76
10	(-2.5, -9.25)	1.72	33	(15.5, 3.5)	4.37
11	(-2.5, -10.75)	0.57	34	(12.5, 8)	2.78
12	(0.5, -13.75)	4.06	35	(12.5, 8)	0.97
13	(2, -15.25)	3.49	36	(14, 3.5)	4.16
14	(2, -15.25)	0.85	37	(23, 8)	2.07
15	(0.5, -13.75)	0.74	38	(24.5, 6.5)	2.97
16	(0.5, -13.75)	2.08	39	(21.5, 8)	4.15
17	(0.5, -7.75)	1.47	40	(23, 6.5)	0.71
18	(0.5, -13.75)	7.53	41	(26, 8)	2.16
19	(0.5, -13.75)	10.07	42	(32, 6.5)	1.14
20	(0.5, -1.75)	1.43	43	(29, 8)	3.29
21	(2, 2)	0.34	44	(32, 6.5)	3.09
22	(2, 8)	0.55	45	(24.5, 3.5)	1.23
23	(2, 5)	2.64	46	(26, 5)	2.88

Tabela A.1: Posições calculadas e erros obtidos no algoritmo soma das distâncias

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-21.0, -10.25)	2.96	24	(1.5, 9.0)	0.32
2	(-20.0, -9.75)	3.10	25	(1.0, 10.0)	0.96
3	(-20.5, -9.75)	1.27	26	(1.5, 6.0)	0.61
4	(-14.5, -10.25)	3.00	27	(6.5, 6.5)	2.28
5	(-13.0, -10.25)	2.58	28	(8.0, 7.0)	2.18
6	(-11.5, -10.25)	1.42	29	(10.5, 5.0)	2.65
7	(-7.0, -10.25)	3.56	30	(11.0, 3.5)	0.53
8	(-7.0, -10.25)	2.25	31	(11.0, 3.5)	3.69
9	(-5.0, -10.75)	1.45	32	(9.0, 7.5)	2.77
10	(-4.0, -10.25)	0.36	33	(10.5, 5.0)	0.89
11	(-4.0, -10.25)	2.12	34	(12.5, 6.5)	1.32
12	(1.5, -9.75)	2.45	35	(13.0, 7.5)	0.57
13	(1.5, -10.75)	7.82	36	(15.5, 6.5)	1.17
14	(0.5, -14.75)	0.76	37	(24.5, 7.5)	3.17
15	(1.5, -8.25)	4.85	38	(24.0, 4.0)	2.45
16	(1.5, -7.25)	4.70	39	(24.0, 5.0)	1.93
17	(1.0, -6.25)	2.80	40	(21.0, 4.5)	2.26
18	(2.0, -6.75)	0.57	41	(25.5, 7.5)	2.62
19	(1.0, -2.25)	1.46	42	(31.0, 6.0)	1.79
20	(1.5, -3.75)	1.54	43	(31.0, 6.0)	1.58
21	(1.5, 1.0)	0.83	44	(31.5, 5.0)	1.92
22	(1.5, 6.5)	1.08	45	(24.0, 5.0)	0.92
23	(1.5, 7.5)	0.99	46	(24.0, 7.5)	1.26

Tabela A.2: Posições calculadas e erros obtidos no algoritmo KNN

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-22.79, -10.66)	1.50	24	(1.6, 8.65)	0.67
2	(-20.08, -9.35)	3.15	25	(1.05, 10.12)	0.85
3	(-18.31, -9.34)	1.90	26	(1.35, 5.96)	0.65
4	(-13.09, -10.49)	4.31	27	(6.56, 7.26)	2.08
5	(-12.8, -10.24)	2.77	28	(7.5, 7.38)	1.57
6	(-11.13, -10.31)	1.78	29	(10.65, 5.49)	2.35
7	(-6.99, -10.23)	3.57	30	(11.11, 3.5)	0.61
8	(-7.36, -10.04)	1.96	31	(10.32, 3.5)	3.74
9	(-5.45, -10.75)	1.05	32	(8.72, 7.66)	3.03
10	(-3.82, -10.07)	0.32	33	(10.83, 5.91)	1.06
11	(-3.68, -10.16)	1.88	34	(12.44, 6.67)	1.50
12	(1.27, -10.96)	2.43	35	(12.81, 7.5)	0.73
13	(1.24, -11.09)	7.47	36	(15.2, 6)	1.56
14	(0.5, -15.15)	0.65	37	(23.31, 6.7)	1.86
15	(1.77, -9.28)	3.90	38	(24.29, 4.55)	2.52
16	(1.44, -7.2)	4.74	39	(23.72, 5.25)	1.87
17	(1, -6.27)	2.78	40	(21.16, 5.68)	1.32
18	(2, -6.71)	0.54	41	(26.19, 7.94)	1.96
19	(1.1, -1.81)	1.89	42	(30.54, 5.41)	2.53
20	(1.32, -3.3)	1.17	43	(30.65, 6.27)	1.70
21	(1.67, 0.85)	0.96	44	(31.66, 5.41)	2.18
22	(1.56, 6.9)	0.71	45	(23.59, 5.25)	1.03
23	(1.53, 6.83)	1.23	46	(23.43, 7.04)	1.95

Tabela A.3: Posições calculadas e erros obtidos no algoritmo WKNN

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-22.79, -10.66)	1.50	24	(1.8, 7.58)	1.76
2	(-21.41, -10.05)	1.66	25	(1.22, 9.66)	1.35
3	(-19.03, -9.3)	1.53	26	(1.3, 6.6)	0.08
4	(-15.3, -9.87)	2.44	27	(3.85, 7.09)	0.84
5	(-12.8, -10.24)	2.77	28	(7.5, 7.38)	1.57
6	(-11.31, -10.16)	1.62	29	(10.65, 5.49)	2.35
7	(-8.87, -10.12)	1.69	30	(10.43, 4.25)	0.42
8	(-7.93, -10.02)	1.46	31	(10.32, 3.5)	3.74
9	(-6.22, -10.75)	0.56	32	(10.11, 6.08)	2.35
10	(-4.29, -10.41)	0.54	33	(10.04, 5.83)	1.58
11	(-3.68, -10.16)	1.88	34	(12.44, 6.67)	1.50
12	(-0.99, -10.48)	0.52	35	(12.81, 7.5)	0.73
13	(1.24, -11.09)	7.47	36	(13.85, 6.63)	1.45
14	(0.88, -12.95)	2.21	37	(23.31, 6.7)	1.86
15	(1.13, -12.26)	0.87	38	(24.13, 5.49)	2.31
16	(0.64, -8.37)	3.69	39	(23.72, 5.25)	1.87
17	(1, -6.27)	2.78	40	(22.58, 5.09)	1.30
18	(2, -6.71)	0.54	41	(26.19, 7.94)	1.96
19	(1.55, -4.4)	0.78	42	(29.77, 6.71)	2.20
20	(1.32, -3.3)	1.17	43	(30.65, 6.27)	1.70
21	(1.67, 0.85)	0.96	44	(31.66, 5.41)	2.18
22	(1.78, 3.7)	3.75	45	(23.59, 5.25)	1.03
23	(1.53, 6.83)	1.23	46	(23.71, 5.77)	2.51

Tabela A.4: Posições calculadas e erros obtidos no algoritmo HKNN

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-20.52, -10.33)	3.44	24	(1.48, 8.11)	1.20
2	(-20.29, -9.67)	2.84	25	(1.15, 9.81)	1.18
3	(-18.65, -9.67)	1.43	26	(1.43, 6.64)	0.04
4	(-14.55, -10.24)	2.96	27	(5.6, 6.72)	1.41
5	(-12.8, -10.24)	2.77	28	(7.5, 7.38)	1.57
6	(-11.53, -10.21)	1.40	29	(10.65, 5.49)	2.35
7	(-8.12, -10.12)	2.43	30	(10.96, 4.16)	0.44
8	(-7.93, -10.02)	1.46	31	(10.32, 3.5)	3.74
9	(-6.01, -10.43)	0.41	32	(9.36, 6.71)	2.61
10	(-3.91, -10.04)	0.23	33	(10.57, 6.09)	1.36
11	(-3.68, -10.16)	1.88	34	(12.44, 6.67)	1.50
12	(-0.12, -10.68)	1.10	35	(12.81, 7.5)	0.73
13	(1.24, -11.09)	7.47	36	(14.38, 6.25)	1.43
14	(0.58, -13.78)	1.48	37	(21.13, 6.23)	0.50
15	(1.08, -10.22)	2.85	38	(23.91, 5.21)	2.06
16	(1.34, -7.85)	4.10	39	(23.72, 5.25)	1.87
17	(1, -6.27)	2.78	40	(22.15, 5.48)	0.89
18	(2, -6.71)	0.54	41	(26.19, 7.94)	1.96
19	(1.11, -3.32)	0.39	42	(29.57, 5.76)	2.89
20	(1.32, -3.3)	1.17	43	(30.65, 6.27)	1.70
21	(1.67, 0.85)	0.96	44	(31.66, 5.41)	2.18
22	(1.78, 5.65)	1.81	45	(23.59, 5.25)	1.03
23	(1.53, 6.83)	1.23	46	(23.4, 6.49)	2.24

Tabela A.5: Posições calculadas e erros obtidos no algoritmo VAKNN

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-20.05, -10.01)	3.86	24	(0.5, 6.5)	2.99
2	(-23.5, -10.75)	0.53	25	(2, 9.5)	1.88
3	(-20.5, -9.25)	1.68	26	(2, 8)	1.51
4	(-17.5, -9.25)	1.95	27	(0.5, 6.5)	4.16
5	(-13, -10.75)	2.75	28	(6.5, 8)	0.51
6	(-13, -9.25)	1.25	29	(16.72, 5.15)	7.92
7	(-10, -10.75)	1.01	30	(11, 6.5)	2.65
8	(-10, -10.75)	0.80	31	(11, 3.5)	3.69
9	(-8.5, -9.25)	2.35	32	(9.5, 3.5)	4.82
10	(-5.5, -10.75)	1.63	33	(8, 8)	4.52
11	(-2.5, -9.25)	1.90	34	(12.5, 8)	2.78
12	(-2.5, -10.75)	1.74	35	(12.5, 8)	0.97
13	(-1.28, -13.37)	5.60	36	(12.5, 8)	2.52
14	(0.5, -12.25)	2.97	37	(22.68, 5.57)	1.62
15	(0.5, -15.25)	2.21	38	(23, 8)	3.06
16	(2, -10.75)	1.28	39	(24.5, 5)	2.37
17	(0.5, -7.75)	1.47	40	(23, 6.5)	0.71
18	(0.5, -7.75)	1.85	41	(25.54, 6.14)	2.94
19	(2, -6.25)	2.67	42	(29, 8)	2.80
20	(0.5, -1.75)	1.43	43	(30.76, 6.74)	1.43
21	(1, 1.72)	0.71	44	(32, 6.5)	3.09
22	(2, 0.5)	6.95	45	(22.98, 5.02)	0.94
23	(2, 8)	0.63	46	(23, 6.5)	2.57

Tabela A.6: Posições calculadas e erros obtidos no algoritmo RKNN

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-22.79, -10.66)	1.50	24	(1.58, 8.62)	0.70
2	(-21.41, -10.05)	1.66	25	(1.22, 9.66)	1.35
3	(-19.15, -9.55)	1.25	26	(1.3, 6.6)	0.08
4	(-16.13, -10.11)	1.63	27	(3.85, 7.09)	0.84
5	(-14.43, -9.25)	1.12	28	(7.18, 7.55)	1.22
6	(-12, -9.84)	1.11	29	(10.05, 6.04)	1.53
7	(-9.99, -9.72)	0.58	30	(10.43, 4.25)	0.42
8	(-7.93, -10.02)	1.46	31	(10.91, 4.91)	2.28
9	(-6.22, -10.75)	0.56	32	(10.11, 6.08)	2.35
10	(-4.29, -10.41)	0.54	33	(10.67, 4.9)	0.72
11	(-3.08, -10.29)	1.31	34	(12.44, 6.67)	1.50
12	(-0.99, -10.48)	0.52	35	(12.81, 7.5)	0.73
13	(1.52, -14.75)	3.86	36	(13.85, 6.63)	1.45
14	(1.05, -15.68)	0.53	37	(22.5, 6.09)	1.17
15	(1.13, -12.26)	0.87	38	(24.13, 5.49)	2.31
16	(0.94, -11.57)	0.72	39	(23.72, 5.25)	1.87
17	(1.03, -9.4)	0.38	40	(22.58, 5.09)	1.30
18	(2, -6.71)	0.54	41	(26.19, 7.94)	1.96
19	(1.55, -4.4)	0.78	42	(29.77, 6.71)	2.20
20	(1.32, -3.3)	1.17	43	(32, 5.42)	1.67
21	(1.67, 0.85)	0.96	44	(32, 5.1)	1.76
22	(1.78, 9.05)	1.62	45	(23.59, 5.25)	1.03
23	(1.53, 6.83)	1.23	46	(23.34, 6.33)	2.37

Tabela A.7: Posições calculadas e erros obtidos no algoritmo Localização Recalculada

ID	**Posição Calculada**	**Erro (em metros)**	**ID**	**Posição Calculada**	**Erro (em metros)**
1	(-20.92, -10.4)	3.07	24	(1.5, 9)	0.32
2	(-20.64, -9.74)	2.49	25	(1, 10)	0.96
3	(-20.5, -9.75)	1.27	26	(1.5, 6)	0.61
4	(-15.09, -10.04)	2.53	27	(4.93, 6.63)	1.08
5	(-15.36, -9.9)	0.33	28	(8.47, 6.48)	2.83
6	(-11.5, -10.25)	1.42	29	(10.53, 6.53)	1.61
7	(-9.58, -9.94)	0.97	30	(11, 3.5)	0.53
8	(-9.07, -9.54)	1.21	31	(10.78, 4.86)	2.34
9	(-5, -10.75)	1.45	32	(10.1, 6.73)	1.94
10	(-4, -10.25)	0.36	33	(10.5, 5)	0.89
11	(-4.03, -10.07)	2.23	34	(12.5, 6.5)	1.32
12	(1.69, -10.32)	2.66	35	(13, 7.5)	0.57
13	(0.81, -14.33)	4.22	36	(15.5, 6.5)	1.17
14	(0.5, -14.75)	0.76	37	(22.65, 5.95)	1.37
15	(0.62, -11.09)	1.95	38	(23.35, 3.95)	1.93
16	(1.5, -9.31)	2.63	39	(23.54, 5.48)	1.92
17	(0.98, -6.64)	2.41	40	(21.91, 5.55)	0.89
18	(2, -6.75)	0.57	41	(27.06, 7.26)	1.10
19	(1, -2.25)	1.46	42	(31.27, 4.73)	2.93
20	(0.94, -2.05)	0.93	43	(31.62, 5.58)	1.59
21	(1.5, 1)	0.83	44	(31.57, 5.16)	2.01
22	(1.5, 6.5)	1.08	45	(24, 5)	0.92
23	(1.5, 7.5)	0.99	46	(24, 7.5)	1.26

Tabela A.8: Posições calculadas e erros obtidos no algoritmo Eliminação de *Outliers*

Bibliografia

[1] João André Silva, Maria João Nicolau, and António Costa. Wifi localization as a network service. *In Proceedings of the 2011 International Conference On Indoor Positioning and Indoor Navigation (IPIN 2011)*, September 2011.

[2] T.K. Sarkar, Zhong Ji, Kyungjung Kim, A. Medouri, and M. Salazar-Palma. A survey of various propagation models for mobile communication. *Antennas and Propagation Magazine, IEEE*, 45(3):51–82, 2003. ISSN 1045-9243. doi: 10.1109/MAP.2003.1232163.

[3] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.905750.

[4] André Günther and Christian Hoene. Measuring round trip times to determine the distance between wlan nodes. In Raouf Boutaba, Kevin Almeroth, Ramon Puigjaner, Sherman Shen, and JamesP. Black, editors, *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, volume 3462 of *Lecture Notes in Computer Science*, pages 768–779. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25809-4. doi: 10.1007/11422778_62. URL `http://dx.doi.org/10.1007/11422778_62`.

[5] K. Pahlavan, Xinrong Li, and J.-P. Makela. Indoor geolocation science and technology. *Communications Magazine, IEEE*, 40(2):112–118, 2002. ISSN 0163-6804. doi: 10.1109/35.983917.

[6] S. Khodayari, M. Maleki, and E. Hamedi. A rss-based fingerprinting method for positioning based on historical data. In *Performance Evaluation of Computer and*

Telecommunication Systems (SPECTS), 2010 International Symposium on, pages 306–310, 2010.

[7] M. Bouet and A.L. dos Santos. Rfid tags: Positioning principles and localization techniques. In *Wireless Days, 2008. WD '08. 1st IFIP*, pages 1–5, Nov 2008. doi: 10.1109/WD.2008.4812905.

[8] Brett Dawes and Kwan-Wu Chin. A comparison of deterministic and probabilistic methods for indoor localization. *J. Syst. Softw.*, 84(3):442–451, March 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2010.11.888. URL `http://dx.doi.org/10.1016/j.jss.2010.11.888`.

[9] Binghao Li, James Salter, Andrew G. Dempster, and Chris Rizos. Indoor positioning techniques based on wireless lan. In *LAN, FIRST IEEE INTERNATIONAL CONFERENCE ON WIRELESS BROADBAND AND ULTRA WIDEBAND COMMUNICATIONS*, pages 13–16.

[10] Americas Headquarters. *Wi-Fi Location-Based Services 4.1 Design Guide*. Cisco Systems, Inc., May 2008.

[11] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5):28–34, 2000.

[12] PerK. Enge. The global positioning system: Signals, measurements, and performance. *International Journal of Wireless Information Networks*, 1(2):83–105, 1994. ISSN 1068-9605. doi: 10.1007/BF02106512. URL `http://dx.doi.org/10.1007/BF02106512`.

[13] G.M. Djuknic and R.E. Richton. Geolocation and assisted gps. *Computer*, 34(2): 123–125, 2001. ISSN 0018-9162. doi: 10.1109/2.901174.

[14] Vasileios Zeimpekis, George M. Giaglis, and George Lekakos. A taxonomy of indoor and outdoor positioning techniques for mobile location services. *SIGecom Exch.*, 3(4):19–27, December 2002. ISSN 1551-9031. doi: 10.1145/844351.844355. URL `http://doi.acm.org/10.1145/844351.844355`.

[15] Mark Moeglein and Norman Krasner. An introduction to snaptrack server-aided gps technology. In *Proceedings of the 11th International Technical Meeting of the*

Satellite Division of The Institute of Navigation (ION GPS 1998), pages 333–342, 1998.

[16] Joel Barnes, Chris Rizos, Jinling Wang, David Small, Gavin Voigt, and Nunzio Gambale. Locata: A new positioning technology for high precision indoor and outdoor positioning. In *Proceedings 2003 International Symposium on GPS\ GNSS*, pages 9–18, 2003.

[17] Joel Barnes, Chris Rizos, Jinling Wang, David Small, Gavin Voigt, and Nunzio Gambale. Locata: the positioning technology of the future. In *Proceedings of the 6thInternational Symposium on Satellite Navigation Technology Including Mobile Positioning & Location Services, Melbourne, Australia July*, 2003.

[18] George Dedes and Andrew G Dempster. Indoor gps positioning. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference*, 2005.

[19] Roy Want. An introduction to rfid technology. *Pervasive Computing, IEEE*, 5(1): 25–33, 2006.

[20] Jeffrey Hightower, Roy Want, and Gaetano Borriello. Spoton: An indoor 3d location sensing technology based on rf signal strength. *UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA*, 1, 2000.

[21] Lionel M Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P Patil. Landmarc: indoor location sensing using active rfid. *Wireless networks*, 10(6):701–710, 2004.

[22] E. Trevisani and A. Vitaletti. Cell-id location technique, limits and benefits: an experimental study. In *Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on*, pages 51–60, 2004. doi: 10.1109/MCSA.2004.9.

[23] Yilin Zhao. Standardization of mobile phone positioning for 3g systems. *Comm. Mag.*, 40(7):108–116, July 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1018015. URL `http://dx.doi.org/10.1109/MCOM.2002.1018015`.

[24] Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal Lara. Accurate gsm indoor localization. In Michael Beigl, Stephen Intille, Jun Rekimoto, and Hideyuki Tokuda, editors, *UbiComp 2005: Ubiquitous Computing*, volume 3660 of *Lecture Notes in Computer Science*, pages 141–158. Springer Berlin Heidelberg, 2005. ISBN

978-3-540-28760-5. doi: 10.1007/11551201_9. URL http://dx.doi.org/10.1007/11551201_9.

[25] S. Gezici, Zhi Tian, G.B. Giannakis, Hisashi Kobayashi, A.F. Molisch, H.V. Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *Signal Processing Magazine, IEEE*, 22(4):70–84, 2005. ISSN 1053-5888. doi: 10.1109/MSP.2005.1458289.

[26] Pete Steggles and Stephan Gschwind. The ubisense smart space platform. In *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, volume 191, pages 73–76, 2005.

[27] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2, 2000. doi: 10.1109/INFCOM.2000.832252.

[28] Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.

[29] Zi-Ning Zhen, Qing-Shan Jia, Chen Song, and Xiaohong Guan. An indoor localization algorithm for lighting control using rfid. In *Energy 2030 Conference, 2008. ENERGY 2008. IEEE*, pages 1–6, 2008. doi: 10.1109/ENERGY.2008.4781041.

[30] E.S. Bhasker, S.W. Brown, and William G. Griswold. Employing user feedback for fast, accurate, low-maintenance geolocationing. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 111–120, March 2004. doi: 10.1109/PERCOM.2004.1276850.

[31] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2013. ISBN 1461463955, 9781461463955.

[32] Irad Ben-Gal. Outlier detection. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer US, 2005. ISBN 978-0-387-24435-8. doi: 10.1007/0-387-25465-X_7. URL http://dx.doi.org/10.1007/0-387-25465-X_7.

[33] T. King, T. Haenselmann, and W. Effelsberg. On-demand fingerprint selection for 802.11-based positioning systems. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–8, June 2008. doi: 10.1109/WOWMOM.2008.4594839.

[34] Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. CRAWDAD data set mannheim/compass (v. 2008-04-11). Downloaded from `http://crawdad.org/mannheim/compass/`, April 2008.

[35] *Veris Aerospond Wireless Sensors: Received Signal Strength Indicator (RSSI).* VERIS INDUSTRIES, 2013.