Universidade do Minho
Escola de Engenharia

Marta Catarina Andrade Rodrigues

Algoritmos probabilísticos para *Wi-Fi Fingerprinting*

Dissertação de Mestrado
Mestrado Integrado em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação
Professor Doutor António Costa
Professora Doutora Maria João Nicolau

Outubro de 2016

Agradecimentos

Neste momento que completo mais uma etapa importante na minha vida gostaria de expressar o meu agradecimento a todos aqueles que contribuíram com a sua ajuda e apoio.

Em primeiro lugar, começo por agradecer ao Professor Doutor António Costa e à Professora Doutora Maria João Nicolau pela orientação, pela paciência, disponibilidade, companheirismo e competência, pedras angulares na execução deste projeto onde partilharam materiais e sugestões que permitiram desenvolver esta dissertação. Agradeço ainda pela motivação que me foram dando ao longo deste projeto. Foi um grande privilégio ter desenvolvido esta dissertação sob a vossa orientação.

Agradeço a toda a minha família, em especial aos meus pais e irmã pelo apoio durante todo o meu percurso académico e pelos sacrifícios suportados para que eu pudesse concluir esta etapa. Um grande agradecimento aos meus tios, Domingos e Deolinda, pela incrível oportunidade que me proporcionaram.

Quero agradecer aos meus colegas e amigos que me acompanharam durante todo o percurso académico. Um agradecimento à minha amiga de longa data, Stéphanie Oliveira, pela atenção, pelo apoio e motivação que foi demonstrando ao longo destes anos. Agradeço também às minhas colegas e amigas Diana Cunha, Inês Xavier e Filipa Gomes, pelo companheirismo, e acima de tudo, pela amizade incondicional. Destes 5 anos além de um curso, levo também comigo amigos para a vida.

A todos vocês, um muito Obrigada!

Resumo

A técnica *Wi-Fi Fingerprinting* é uma técnica amplamente utilizada no posicionamento em interiores. Através desta técnica é possível determinar a posição do dispositivo, combinando os valores da intensidade do sinal recebidos com os valores da intensidade do sinal pré-adquiridos, presentes numa base de dados. O grande problema desta técnica é que, ao longo do tempo o cenário vai sofrendo várias alterações, condicionando a estimativa do posicionamento. Já foram propostos vários algoritmos de localização baseados em *fingerprintin*g, sendo o mais popular o algoritmo *k Nearest Neighbors (KNN)*.

O propósito desta dissertação centra-se em construir novos algoritmos que permitam estimar o posicionamento, baseados na técnica *Wi-Fi fingerprinting*. São abordados nesta dissertação dois tipos de algoritmos, algoritmos determinísticos e algoritmos probabilísticos, com o intuito de avaliar o desempenho de cada um deles em ambientes *indoor*. Entre os algoritmos determinísticos, foi escolhido e implementado um algoritmo hierárquico já existente. Este algoritmo inclui três etapas distintas, nomeadamente a identificação do edifício, depois do respetivo piso e finalmente a estimativa da localização. Tendo em conta o ambiente em estudo, este algoritmo hierárquico apresenta resultados satisfatórios, sendo utilizado como referência na análise de desempenho dos restantes algoritmos aqui apresentados. Ainda nos algoritmos determinísticos, são efetuadas propostas de alteração ao algoritmo hierárquico de forma a melhorar os resultados. Relativamente aos algoritmos probabilísticos, são descritas e implementadas três variantes. Estas três variantes calculam a probabilidade de uma *fingerprint* pertencer a um determinado local, utilizando diferentes metodologias. A primeira variante, faz uso de uma distribuição baseada em histogramas. É construído um histograma de valores da intensidade do sinal para cada ponto de acesso de uma *fingerprint*. A segunda variante recorre à probabilidade de um ponto de acesso ter sido observado numa determinada posição. A terceira variante utiliza a função gaussiana de *Kernel* para cada ponto de acesso. Todos estes algoritmos, tanto os determinísticos como os probabilísticos foram testados recorrendo a *datasets* de dados reais, que permitiram obter os resultados descritos neste documento.

Abstract

Wi-Fi Fingerprinting is a widely used technique in interior positioning systems. Due to this technique it is possible to determine the position of a device, combining the values of the received signal intensity with the values of the signals intensity pre-acquired from a database. The main problem of this technique is that, over the time the scenario suffer several changes conditioning the estimated position. There have been proposed several localization algorithms based in fingerprinting in which the most popular is the k Nearest Neighbors algorithm.

This dissertation focuses on developing new algorithms that permit the estimation of the positioning, based in the Wi-Fi fingerprint technique. In this dissertation we make two approaches, deterministic algorithms and probabilistic algorithms, with the aim to evaluate the performance of each one in indoor environments. Between the deterministic algorithms, an existent hierarchical algorithm was chosen and then implemented. This algorithm includes three different steps, the building identification, the floor identification and finally the estimated localization. Taking into account the study environment, this hierarchical algorithm shows decent results, so it is used as a reference in the performance analyses of the other algorithms presented here. Still in the deterministic algorithms, it is made several proposals to modify the hierarchical algorithm in order to improve the results. Relatively to the probabilistic algorithms it is described and implemented three variants. These three variants calculate the probability of a fingerprint belong to a particular location, using several methodologies. The first uses distribution histograms. It is built an histogram of the signal intensity values for each access point of a fingerprint. The second resorts on the probability of an access point being observed in a certain position. The third uses the Kernel's gaussian function for each access point. All of these algorithms, both deterministic as probabilistic were tested using datasets of real data, that permitted to obtain the results described in this document.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Acrónimos

A-GPS *Assisted*-GPS

AoA *Angle of Arrival*

AP *Access Point*

CAD *Computer Aided Design*

CDMA *Code Division Multiple Access*

FP *Fingerprint*

GPS *Global Position System*

GSM *Global System for Mobile Communications*

ID *Identify*

IDE *Integrated Development Environment*

IEEE *Institute of Electrical and Electronics Engineers*

IPIN *International Conference on Indoor Positioning and Indoor Navigation*

IR *Infrared*

ISM *Industrial Scientific and Medical*

kNN *k-nearest-neighbor*

LTE *Long Term Evolution*

MAC *Media Access Control*

MLP *Multilayer Perceptron*

PoA *Phase of Arrival*

RF *Radio Frequency*

RFID *Radio – Frequency Identification*

RP *Reference Point*

RSS *Received Signal Strength*

RSSI *Received Signal Strength Indicator*

RTOF *Roundtrip Time of Flight*

SVC S*upport Vector Classification*

SVM S*upport Vector Machine*

SVR S*upport Vector Regression*

TDoA *Time Difference of Arrival*

ToA *Time of Arrival*

UML *Unified Modeling Language*

UWB *Ultra – Wideband*

WCDMA *Wide Band Code Division Multiple Access*

WkNN *Weighted k-nearest-neighbor*

WLAN *Wireless Local Area Network*

Símbolos

Tabela 1 - Lista de símbolos

Símbolo	*Descrição*
$FP_{i_{teste}}$	Vetor de valores RSS de teste (*fingerprint i* de teste)
F	Vetor de *fingerprints*
M	Conjunto da fase *offline*
RSSn	Valor RSS do ponto de acesso *n*
RSS_j	Valor RSS do AP_j da *fingerprint online*
AP_{teste}	AP mais forte de teste
AP_{treino}	AP mais forte de treino
$(long_i, lat_i)$	Coordenadas de um estado de localização
t	Tempo de propagação de um sinal entre o emissor e o recetor
v	Velocidade de propagação do sinal
θ	Ângulo de um determinado ponto de referência
ρ	Valor médio
σ	Desvio padrão
n	Número de APs
$obsAP_n$	Número de vezes que o AP_n foi observado
k, k1, k2	Número de vizinhos a selecionar
r	Valor do intervalo
defRSSI	Valor RSSI por defeito
d	Distância usada como raio de uma circunferência centrada num AP
D_i	Distância Euclidiana
w_i	Peso associado a cada *k* vizinho
Sim()	Função de similaridade
$Hist_n^P$	Histograma do AP *n* para a posição P
$Prob^P$	Probabilidade da posição P
Erro	Erro de cada amostra
Distancia(R_i, E_i)	Distância Euclidiana entre a posição real e a estimada
$Prob_n^P$	Probabilidade do AP_n na posição P
$K(RSSI_{teste};\ \mu)$	Função *kernel*

Capítulo 1

Introdução

Neste primeiro capítulo é apresentado o enquadramento ao tema da dissertação, assim como os objetivos que se pretendem alcançar com a realização da mesma. Na terceira secção é apresentada a metodologia utilizada nesta dissertação e na última, a estrutura adotada na elaboração deste documento.

1.1. Enquadramento

Nos últimos anos, temos assistido a um crescimento acentuado na utilização de dispositivos móveis, ao ponto destes passarem a fazer parte do quotidiano das pessoas. Hoje em dia, quase todas as pessoas carregam um telemóvel, *tablet* ou portátil, onde quer que estejam, não só para estarem contactáveis, mas também para poderem aceder à Internet. Uma vez que as pessoas estão em constante movimento, a questão de posicionamento é muito relevante, isto porque várias aplicações tiram partido da localização do dispositivo para fornecerem um melhor serviço ao utilizador.

O Sistema de Posicionamento Global (GPS) é o sistema de posicionamento exterior predominante, capaz de localizar dispositivos usando os sinais de rádio transmitidos por satélites. No entanto, embora funcione muito bem no exterior, o GPS, não funciona adequadamente em ambientes fechados, como edifícios, casas, ou mesmo espaços urbanos cercados de construções muito altas. Por esse motivo, não pode ser utilizado por determinadas aplicações que necessitam de obter a localização das pessoas no interior de edifícios.

O desenvolvimento de técnicas de localização apropriadas para ambientes interiores tem que lidar com um conjunto de problemas, como a existência de paredes, equipamentos e outros obstáculos, o movimento de seres humanos, portas, etc. Têm sido desenvolvidos muitos projetos de investigação na localização no interior de edifícios, recorrendo a diversas tecnologias, isoladas ou combinadas.

Uma das técnicas mais utilizadas para implementar sistemas de posicionamento em interiores designa-se por *WiFi Fingerprinting*. Esta técnica baseia-se na obtenção de assinaturas *WiFi* (*WiFi Fingerprinting*), que podem ser obtidas pela medição das forças dos sinais *WiFi* recebidos em cada posição. Esta técnica inclui duas fases, a fase *offline* e a fase *online*. Na fase *offline*, cria-se uma base de dados onde são guardados para cada ponto de referência, a sua localização e a respetiva assinatura *Wifi* recolhida à *priori*. Na fase *online*, é realizada uma comparação entre os pontos guardados anteriormente e a assinatura *Wifi* atual do dispositivo. A correspondência mais próxima é selecionada e enviada ao dispositivo como uma estimativa de posição. Para determinar a correspondência mais próxima são utilizados algoritmos de localização.

1.2. Objetivos

Os algoritmos de localização abordados neste projeto são baseados na técnica de localização *fingerprinting*. O trabalho tem por base projetos anteriores [1][2][3][4].

Neste trabalho foi definido como principal objetivo estudar, desenvolver e avaliar novos algoritmos de localização baseados em probabilidades, para o cálculo da posição em múltiplos edifícios contíguos e em edifícios de múltiplos pisos. A solução deve permitir identificar o edifício, o piso e as respetivas coordenadas da posição de um dispositivo, com a maior exatidão possível. Para que estas soluções sejam concebidas, é necessário dar resposta a um conjunto de desafios, entre os quais se incluem:

- Identificar métodos de deteção do edifício;
- Identificar métodos de deteção do piso;
- Definir algoritmos de localização baseados em probabilidades;
- Implementar os algoritmos probabilísticos propostos;
- Comparar os resultados dos algoritmos probabilísticos com os resultados dos algoritmos determinísticos e concluir sobre as melhorias de precisão e exatidão obtidas.

1.3. Metodologia

A fase inicial deste trabalho consiste em adquirir conhecimentos sobre o tema, também conhecida por análise do estado da arte, onde são efetuadas pesquisas de artigos relevantes e o estudo daqueles mais pertinentes para o problema em causa.

Após adquirir os conhecimentos necessários, é essencial analisar os algoritmos já existentes. Depois da análise de toda a estrutura base desses algoritmos é efetuado o desenho de uma proposta de solução, a definição de algoritmos probabilísticos de localização *indoor*.

Os algoritmos de localização utilizados em ambientes interiores, em geral, recorrem a um conjunto de dados recolhidos na fase de treino e a um conjunto de assinaturas, consideradas as assinaturas atuais do dispositivo, como está representado na Figura 1. A fase *offline*, chamada de fase de treino, consiste na recolha de informação associada a cada ponto de referência, informação essa que será guardada numa base de dados de *fingerprints* (mapa de rádio). Esta fase tem como principal função criar um mapa de assinaturas baseadas na força do sinal dos vários APs em cada posição. Este mapa contém uma lista de *fingerprints* e a posição real correspondente. A fase *online* consiste na estimação da localização do dispositivo, através da comparação da assinatura *Wifi* recolhida, *online,* no ponto de localização do dispositivo e a assinatura que foi recolhida na fase *offline*. Considerando um conjunto de dados RSSI recebidos nesta fase é possível encontrar a correspondência mais próxima na base de dados que armazena a localização de cada ponto de referência (base de dados *offline*).

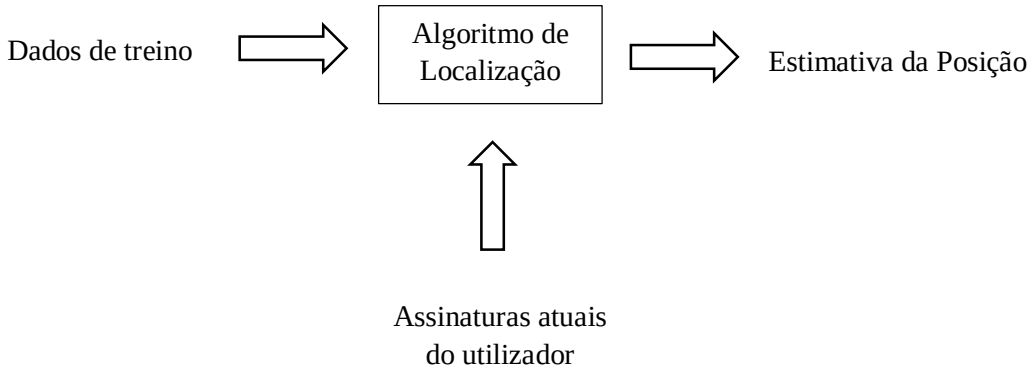

Figura 1 - Esquema geral do Algoritmo de Localização

Neste trabalho, são utilizados dois dos três *datasets* fornecidos pela competição EvALL [5] organizada pela IPIN 2015 [6], nomeadamente o *dataset training* e o *dataset validation.* Os *datasets* são representados por um conjunto de amostras, cada amostra é representada por um vetor, onde as primeiras 520 dimensões correspondem aos valores RSSI dos 520 APs e as restantes 9 às coordenadas, ao piso, ao edifício, ao espaço, à posição relativa, ao ID do utilizador, ao ID do dispositivo e ao *timestamp*, respetivamente. Neste projeto apenas serão utilizadas as coordenadas, o piso e o edifício destas últimas 9 dimensões, visto serem estes os dados necessários para o que é pretendido. Os dados de treino correspondem ao *dataset training* e a as assinaturas atuais do utilizador correspondem ao vetor dos valores RSSI dos 520 APs do *dataset validation.*

No seguimento do desenho da proposta de solução surge a sua implementação em linguagem *Java.* Aquando concluída, permite obter resultados e conclusões de forma a avaliar a legitimidade da proposta apresentada. A validação destes resultados é efetuada através dos dados presentes no *dataset validation*, nomeadamente as coordenadas, o piso e o edifício. Através destes dados é possível comparar a posição estimada com a posição real do dispositivo.

1.4. Estrutura da Dissertação

Esta dissertação está organizada em seis capítulos. No capítulo 1 é apresentada uma introdução ao tema da dissertação através do enquadramento deste, e os respetivos objetivos a alcançar.

O capítulo 2 apresenta um levantamento do estado da arte dos sistemas de localização. São descritas as técnicas de localização existentes, os sistemas de localização mais conhecidos e por fim, uma comparação entre eles.

No capítulo 3 são apresentados os algoritmos de localização baseados em *Wi-Fi Fingerprinting*. É apresentado um algoritmo de localização determinístico, baseado num modelo já existente, que servirá como referência na análise de desempenho. São ainda apresentadas três propostas de alterações a este algoritmo, com vista a melhorar o seu desempenho. São ainda apresentadas três novas propostas de soluções para implementar a estimativa da posição de um utilizador, baseadas em algoritmos probabilísticos.

O capítulo 4 diz respeito à implementação dos algoritmos de localização baseados em *Wi-Fi Fingerprinting*. Este capítulo é composto por duas secções, onde na primeira é descrito o cenário de teste e na segunda as funções utilizadas na implementação dos algoritmos.

No capítulo 5 são descritos os testes experimentais realizados e os resultados obtidos na avaliação dos algoritmos implementados. São ainda comparados os diferentes resultados e apresentados alguns testes complementares, de forma a compreender melhor os fatores que influenciam os resultados dos algoritmos.

No capítulo 6, são apresentadas as conclusões desta dissertação, resumindo os objetivos cumpridos e possíveis caminhos a serem explorados em trabalhos futuros.

Os apêndices A, B, C, D e E apresentam um conjunto de resultados de experiências e análises complementares realizadas no contexto deste trabalho.

Capítulo 2

Técnicas e sistemas de localização

Hoje em dia, diversos sistemas e aplicações utilizam a localização das pessoas e dos dispositivos móveis. A maioria dessas aplicações tem como requisito principal a localização com precisão em tempo real de objetos no interior de edifícios. Desta forma os serviços de localização em interiores, tornaram-se num pré-requisito fundamental.

Os modelos de localização não garantem exatidão 100%. Existem apenas técnicas ou algoritmos que permitem uma aproximação aos valores reais. Neste capítulo são apresentadas algumas dessas técnicas e sistemas de localização.

2.1. Técnicas de localização

As técnicas ou algoritmos de localização podem ser divididos em três grupos distintos, proximidade, triangulação e análise de cenário (*scene analysis*). Em seguida são apresentadas e descritas cada uma destas técnicas de localização.

2.1.1. Proximidade

Na técnica da proximidade [7] [8] [9], ou também, denominada de célula de origem, a localização do dispositivo é estimada através da sua proximidade com o ponto de acesso (por exemplo, antenas). Sabendo onde se encontram os pontos de acesso e as suas áreas de cobertura é possível, geograficamente, saber a área de cobertura do ponto de acesso a que o dispositivo está ligado. Quando um dispositivo é detetado por um ponto de acesso, assume-se que o dispositivo está nessa área. Quando um dispositivo é detetado por vários pontos de acesso, a localização do dispositivo é dada pelo ponto de acesso que apresentar maior valor de potência de sinal recebido.

Esta técnica é relativamente simples de implementar. É frequentemente implementada em sistemas de comunicação por infravermelhos (IR), RFID (*Radio Frequency Identification*) e também em redes de dados móveis (GSM, 3G, 4G, LTE, etc).

2.1.2. Triangulação

A triangulação consiste numa técnica de localização que utiliza as propriedades geométricas dos triângulos para estimar a localização de um objeto [7]. Esta técnica divide-se em duas categorias: a lateralização (*lateration*), que usa como medida as distâncias, e a angulação (*angulation*), que usa como medida os ângulos.

2.1.2.1. Lateralização

Na lateralização é possível obter a posição de um ponto através da distância entre o mesmo ponto e pontos de referência. Fazem parte desta técnica os métodos *Received Signal Strenght* (RSS), *Time of Arrival* (ToA) e *Time Difference of Arrival* (TDoA).

Received Signal Strenght (RSS)

Este método permite fazer medições ao nível da potência do sinal recebido, a fim de estimar a distância percorrida pelo sinal. Ao identificar o RSS, sabendo a potência de saída, as suas perdas e ganhos, é possível calcular a distância do dispositivo móvel ao ponto de acesso. Pode-se estimar a posição, medindo a distância para mais do que um ponto de acesso.

Time of Arrival (ToA)

Este método [7] [8] [9], mede o tempo de propagação de um sinal entre o emissor e o recetor (t), esse tempo, posteriormente é multiplicado pela velocidade de propagação do sinal, obtendo assim a distância $d = t \times v$.

A distância é usada como um raio de uma circunferência centrada num *Access Point* (AP).

Para se poder estimar a posição de um determinado dispositivo é necessário medir a distância em pelo menos 3 pontos de referência não colineares, como se pode verificar pela Figura 2.

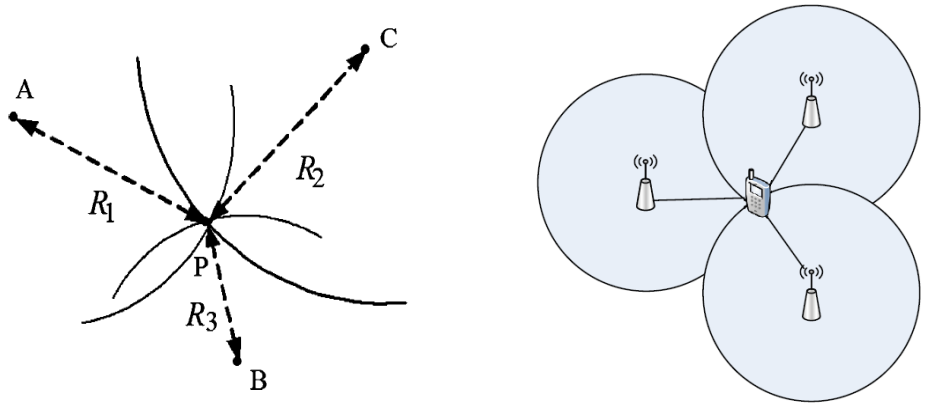

Figura 2 - Localização baseada no método Time of Arrival (ToA)[7][8]

Na Figura 2 à esquerda, o ponto P representa a localização do dispositivo. Com a interseção da circunferência A e B obtemos dois pontos possíveis para a localização do dispositivo, com uma terceira circunferência, ficamos com um único ponto possível para a localização.

Uma das desvantagens deste método é que todos os APs têm de estar sincronizados no tempo, e o ruído do sinal e a propagação *multipath*[1] também podem prejudicar fortemente este método.

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.

Em cada medição TDoA [7] [8] [9] é calculada uma hipérbole, que representa a diferença de distâncias entre os dispositivos de referência. Através da interseção de duas

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

ou mais hipérboles é possível determinar a posição de um determinado ponto, como mostra na Figura 3. A duas hipérboles são formadas através de três dispositivos de referência (A, B, C), para assim obter um ponto de interseção P.

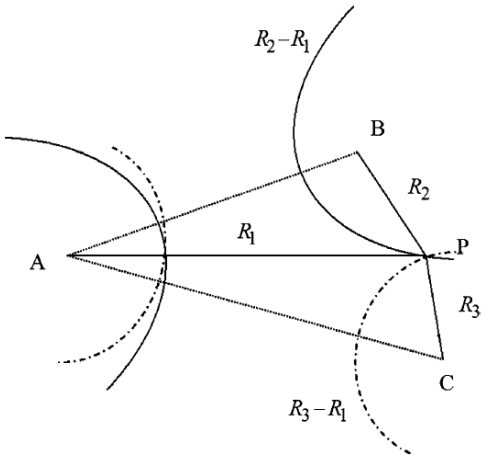

Figura 3 - Localização baseada no método Time Difference of Arrival (TDoA)[7]

Para além destas técnicas de localização existem outras técnicas de lateralização, 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).

2.1.2.2. Angulação

Na angulação a posição do dispositivo móvel é calculada com base no ângulo de incidência em que o sinal é recebido. Nesta técnica é utilizado o método *Angle of Arrival* (AoA).

Angle of Arrival (AoA)

A localização [7] [8] [9] é obtida através da interseção das retas formadas pelos ângulos de referência, como se pode ver na Figura 4. Para a utilização deste método, são necessários pelo menos dois APs.

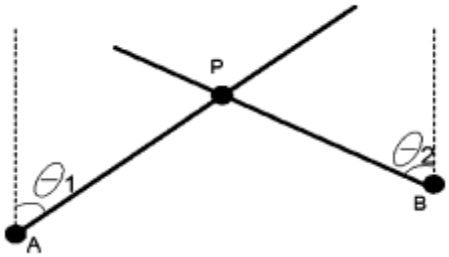

Figura 4- Localização baseada no método Angle of Arrival (AoA) [7]

Na figura anterior são utilizados dois pontos de referência (A e B) para os quais são calculados os respetivos ângulos ($\theta 1$ e $\theta 2$). Através da interseção das duas retas é estimada a localização do dispositivo P.

2.1.3.Scene Analysis

A técnica de análise de cenário consiste em fazer uma análise prévia do ambiente no qual se pretende implementar o sistema de localização.

Nesta técnica [7] [8] [9] são inicialmente recolhidas as características do cenário, e depois estimada a localização do dispositivo, através da correspondência entre os valores atuais e os valores recolhidos à *priori* (*fingerprints*). A técnica de localização baseada em RSS (*Received Signal Strenght*) *fingerprinting* usa como característica a potência do sinal recebido e é a mais utilizada nos sistemas que utilizam o método de análise de cenário.

A localização baseada em *fingerprinting* consiste numa técnica onde é obtida uma "assinatura digital" das características do sinal, em determinadas posições de um cenário. Existem duas fases de localização *fingerprinting*, a fase *offline* e a fase *online*.

Na fase *offline* é realizada uma análise do cenário, são recolhidas informações em determinados pontos de referência (RPs- *Reference Points*) dos vários pontos de acesso (APs-*Access Points*) que fazem parte do cenário, e esses valores da potência do sinal recebido são registados numa base de dados.

Na fase *online*, o sistema de localização realiza uma comparação entre os valores atuais da potência do sinal recebido e os valores registados anteriormente na base de dados, para estimar a localização do dispositivo.

Existe uma variedade de algoritmos que utilizam a técnica de localização baseada em *fingerprinting*, e estes podem ser classificados em dois tipos:

- **Determinísticos:** Este tipo de algoritmos utilizam as distâncias no domínio dos sinais para encontrar o ponto que melhor corresponde à *fingerprint* da fase *online.* Alguns destes algoritmos são o *k-nearest-neighbor* (kNN), o *weighted k-nearest-neighbor* (WkNN) e as redes neuronais.

- **Probabilísticos:** Este tipo de algoritmos tem em conta as informações do cenário, funções de probabilidade e teorema de *Bayes.* Alguns exemplos deste tipo de algoritmos são os métodos probabilísticos (que utilizam o método de *Bayes*) e *support vector machine* (SVM).

2.1.3.1. K-Nearest-Neighbor

O algoritmo KNN [7] [10] [11] tem como objetivo calcular as distâncias entre os valores RSS medidos na fase *online* e todas as *fingerprints* da fase *offline*, de forma a encontrar os *k* pontos de referência mais próximos do objeto a localizar, sendo estes os *k* vizinhos mais próximos.

Para obter os *k* vizinhos mais próximos, é calculada a distância entre o objeto a localizar e cada ponto de referência do cenário, baseado em valores RSS. O cálculo da distância é efetuado recorrendo à distância Euclidiana. O RP mais próximo do objeto a localizar é definido através da distância entre eles, quanto menor a distância entre o RP e o objeto, mais próximo se encontra do objeto.

Considerando um vetor de RSS da fase *online* (fase de teste) $\{RSS_1, RSS_2, \ldots, RSS_n\}$ onde RSS representa a potência do sinal recebido dos *n* respetivos APs e considerando um conjunto de vetores de RSS da fase *offline*, é possível obter a distância Euclidiana através da seguinte equação (1) [10]:

$$D_i = \sqrt{\sum_{j=1}^{n} \left(RSS_{ij} - RSS_j\right)^2} \quad (1)$$

Onde, RSS_{ij} corresponde ao valor de RSS de cada AP_j da *fingerprint* i da fase *offline* e RSS_j corresponde ao valor RSS de cada AP_j da *fingerprint* da fase *online*.

Para cada *fingerprint* da fase *online* é calculada a distância entre ela e cada uma das *fingerprints* da fase *offline*. Essas distâncias são ordenadas, permitindo obter um conjunto de *k* amostras com as menores distâncias obtidas. O valor de *k* é um parâmetro predefinido e este pode assumir valores entre 1 e o número total de RPs.

Após selecionar as *k* amostras, a posição do objeto é dada através da equação (2):

$$\left(long_x, lat_y\right) = \sum_{i=1}^{k} \frac{1}{k} (long_i, lat_i) \quad (2)$$

Onde, $(long_i, lat_i)$ é a respetiva longitude e latitude de cada uma das *k* amostras e $\left(long_x, lat_y\right)$ o resultado final, a posição do objeto a localizar, longitude e latitude da *fingerprint online* em estudo. Ou seja, as coordenadas do objeto são dadas através da média aritmética das coordenadas dos *k* vizinhos mais próximos.

Para além do algoritmo kNN existe ainda uma variante deste, chamada de *Weighted kNN* (WkNN) [6] [12], onde as coordenadas do objeto a localizar não são dadas através do cálculo da média aritmética dos *k* vizinhos mais próximos, mas sim atribuindo pesos mediante a distância Euclidiana de cada vizinho mais próximo. Os pontos de referência não se encontram todos à mesma distância do objeto, alguns encontram-se mais próximos. Aos pontos de referência que se encontram a uma menor distância é-lhes atribuído um peso maior do que àqueles que se encontram mais afastados (maior distância). A Figura 5 ilustra o que acontece com cada um destes algoritmos.

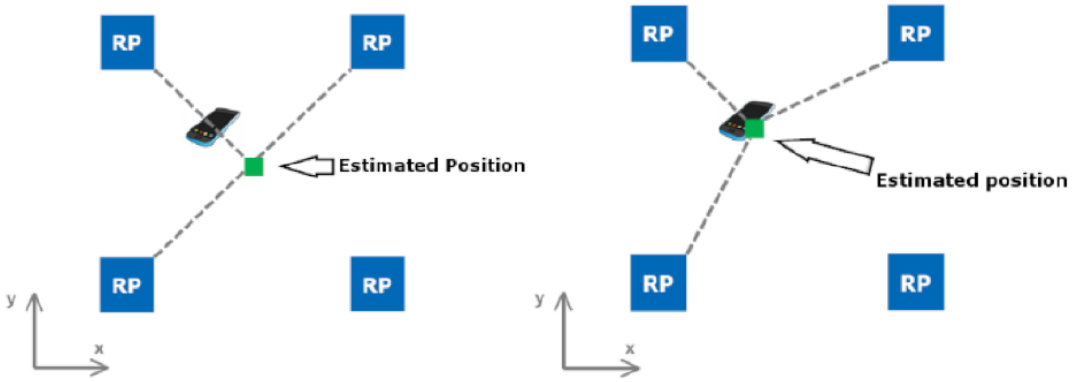

Figura 5 - Exemplo da localização do dispositivo através do algoritmo kNN (à esquerda) e WkNN (à direita)[6]

O peso dos *k* vizinhos é definido através do cálculo do inverso ao quadrado da distância Euclidiana de cada *k* RP, dividindo-o pelo inverso da soma do quadrado das distâncias Euclidianas de todos os *k* RPs. Ou seja, a fórmula apresentada na expressão (3).

$$w_i = \frac{\frac{1}{D^2}}{\sum_{i=1}^{k} \frac{1}{D_i^2}} \qquad (3)$$

2.1.3.2. Redes Neuronais

Uma rede neuronal consiste num conjunto de neurónios dispostos de maneira a formar uma rede, onde a saída de alguns neurónios alimentam a entrada de outros [13].

O neurónio consiste num conjunto de entradas multiplicadas por pesos sinápticos, um somador, que soma cada uma das entradas do neurónio com o respetivo peso sináptico, e uma função de ativação que limita a amplitude da saída do neurónio, tal como se pode visualizar na Figura 6. Uma sinapse é o nome dado à conexão entre neurônios. Nas conexões são atribuídos valores, que são chamados de pesos sinápticos.

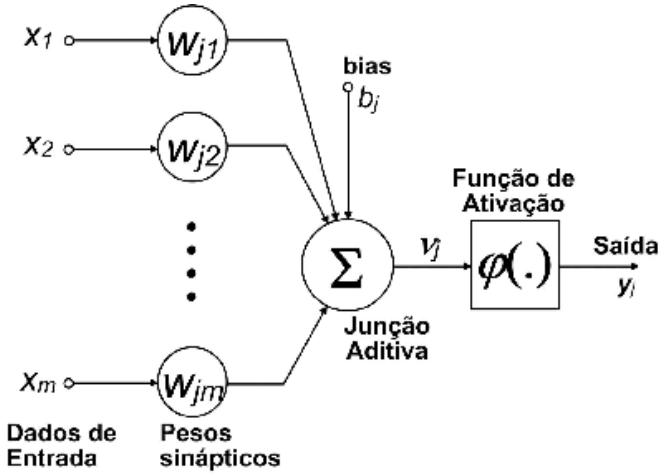

Figura 6 - Diagrama de Blocos de um Neurónio [13]

A função de ativação é uma função não linear e, como se pode ver na Equação (4), geralmente é uma função sigmóide[2][14],

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (4)$$

Normalmente em sistemas de localização baseados em redes neuronais é utilizado uma rede *multilayer perceptron* (MLP) [15][16][17] (Figura 7) com uma camada escondida [7]. Esta camada escondida corresponde à camada que se encontra entre a entrada e a saída.

[2] Função sigmóide - é uma função matemática e o nome "sigmóide" vem da forma do seu gráfico em S.

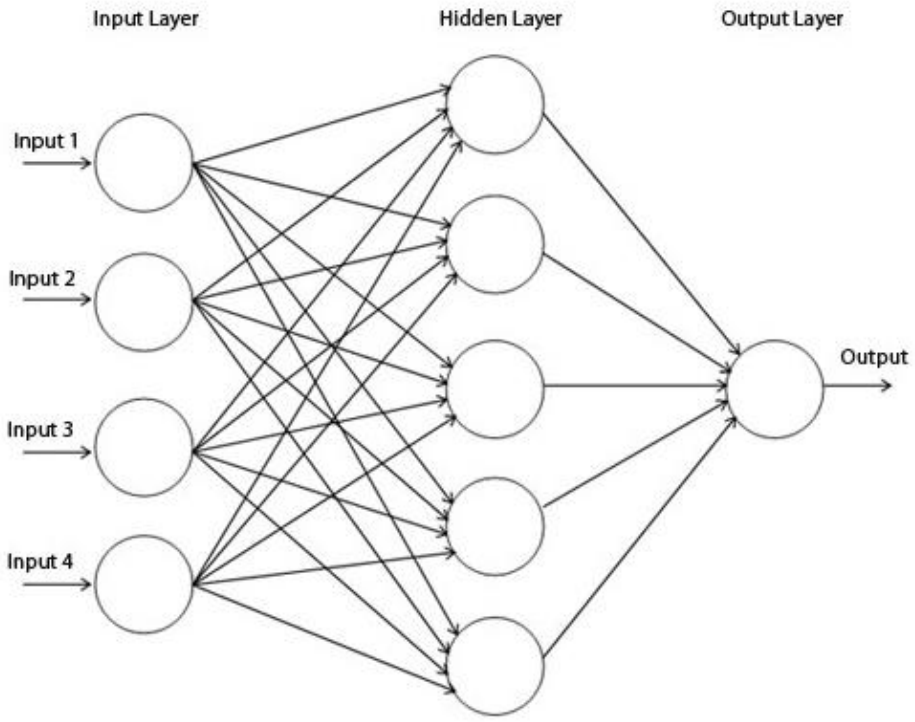

Figura 7- Diagrama de uma Rede Neuronal [18]

Uma rede neuronal MLP pode ser treinada com várias amostras de *fingerprints* recolhidas na fase *offline*, com a finalidade de obter os pesos sinápticos associados a cada uma das entradas.

Back-propagation [13], uma abreviatura para "*backward propagation of errors*", é um método de treino comum em redes neuronais. O objetivo do *back-propagation* é otimizar os pesos de modo a que a rede neuronais possa aprender a mapear corretamente os *inputs* arbitrários para os *outputs*.

2.1.3.3. Métodos Probabilísticos

Os algoritmos probabilísticos têm sido muito usados em trabalhos na área da localização. Este tipo de algoritmos utiliza probabilidades inferidas para determinar qual a *fingerprint* de treino mais parecida com a *fingerprint* de teste.

O modelo probabilístico baseado no teorema de *bayes* estima a localização da *fingerprint* através de probabilidades condicionadas. Este modelo pressupõe um

conhecimento prévio da distribuição de probabilidade da localização do utilizador [19]. Este conhecimento consiste num estudo do modelo de propagação do sinal.

Assumindo que cada localização L tem uma probabilidade *à priori* P(L), que inicialmente pode ser a mesma para cada localização, considerando F o vetor de *fingerprints* da fase *online* e M o conjunto da fase *offline* (mapa de *fingerprints*), através do teorema de *Bayes* temos (Equação (5)):

$$P(L|F) = \frac{P(F|L)\,P(L)}{P(F)} = \frac{P(F|L)P(L)}{\sum_{k\in M}[P(F|L_k)P(L_k)]} \qquad (5)$$

A localização que corresponde à atual *fingerprint* é aquela cuja probabilidade P(L|F) é a maior. Supondo que F corresponde ao vetor de valores RSS observados durante a fase *online*, e as *fingerprints* f_A e f_B às localizações candidatas. Pela regra de decisão de *Bayes*, f_A é selecionada em vez de f_B se (6) [14],

$$P(f_A|F) > P(f_B|F) \qquad (6)$$

Para cada localização L, é possível estimar a função de distribuição de probabilidade P(F|L), a probabilidade condicionada do vetor de *fingerprints* F.

Em [11] são sugeridos dois métodos para estimar esta função: *kernel method* e *histogram method*, que consistem no seguinte:

- *Kernel Method*: Dadas *n* amostras do valor de RSS de um determinado AP, para cada amostra é produzida uma função de probabilidade como uma função gaussiana, denominada de *kernel*.

A função gaussiana, considerada como *kernel*, tem um valor médio ρ, que é igual à média dos valores RSS desse AP numa determinada posição, e tem σ como um parâmetro ajustável que determina a largura de *kernel*. Na Figura 8 é ilustrado o efeito do parâmetro σ.

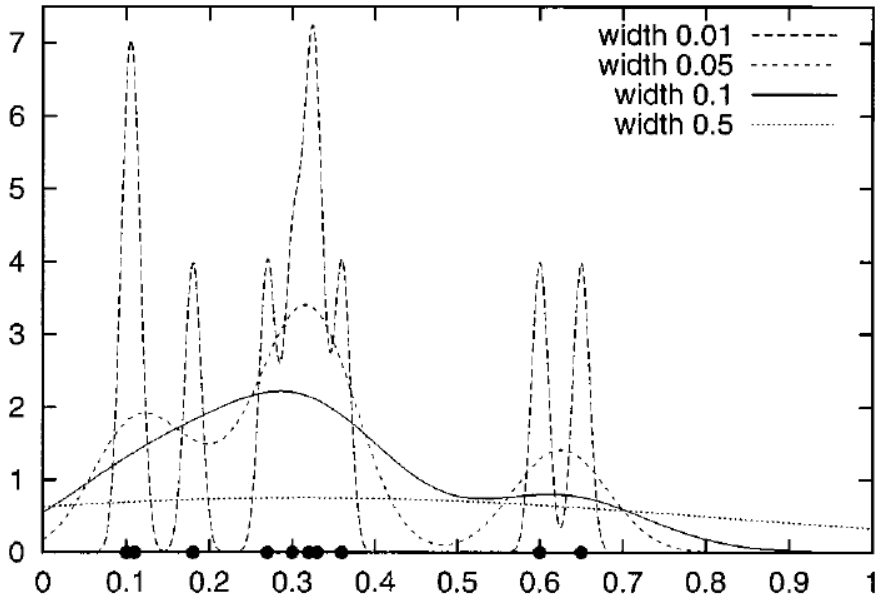

Figura 8 - Exemplos da densidade de kernel estimada com Gaussian kernel para diferentes valores de largura de kernel σ [11]. *Valores observados (0.1,0.11,0.18,0.27, 0.3, 0.32, 0.33, 0.36, 0.6, 0.65)*

Assim, a função de probabilidade de uma amostra S (Equação (7)), de valores RSS de um local L, é a soma de todas as *n* funções *Gaussian Kernel*.

$$P(S|L) = \frac{1}{n}\sum_{i=1}^{x}[\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(S-\rho)^2}{2\sigma^2}}] \quad (7)$$

Para tornar este método ajustável a *p* pontos de acesso, são multiplicadas todas as probabilidades condicionais (Equação (8)).

$$P(S|L) = P(S_1|L)\,P(S_2|L)\,\ldots\,P(S_p|L) \quad (8)$$

- *Histogram Method*: Este método estima as funções de probabilidade contínuas usando funções de probabilidade discretas. Para cada valor RSS é obtida uma probabilidade. Considerando um conjunto de *x* amostras pertencentes à mesma posição, e um determinado AP, a probabilidade para cada um dos diferentes valores RSS observados é dada através do número de ocorrências desse valor sobre o total de valores

medidos (*x* amostras). Neste método é definido um conjunto de *bins*[3], isto é, um conjunto de intervalos não-sobrepostos que cubram todos os valores possíveis, desde o seu mínimo até ao seu máximo de RSS [11].

Em [11] foram efetuados testes para diferentes números de *bins* (3, 7 e 23 *bins*), concluindo que quanto mais *bins* forem considerados, maior é a aproximação do histograma, Figura 9.

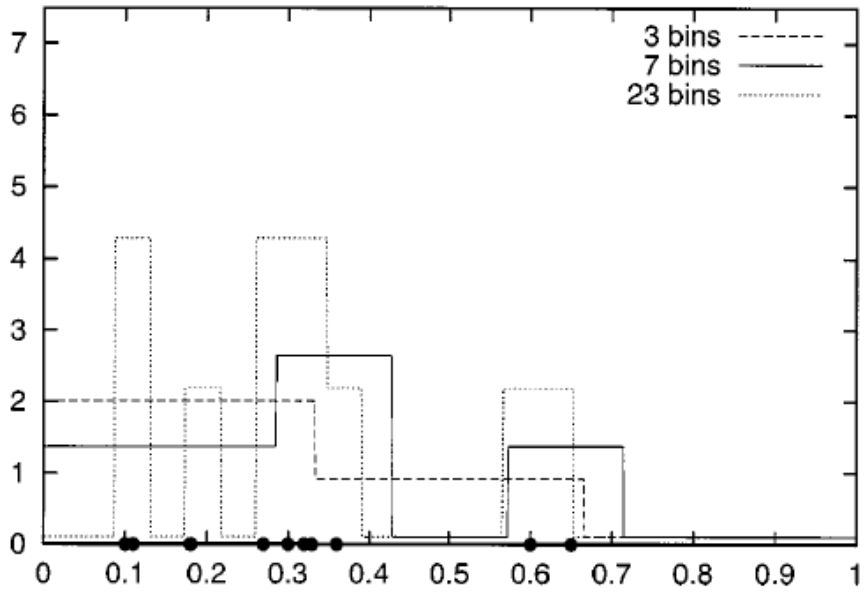

Figura 9 - Exemplos de densidades de histograma estimadas com diferentes números de bins [11]*. Valores observados (0.1, 0.11, 0.18, 0.27, 0.3, 0.32, 0.33, 0.6, 0.65)*

Um método mais sofisticado é apresentado em [20], onde é calculado o valor de P(F|L) a partir de dois histogramas diferentes, onde o primeiro representa os APs detetados numa determinada localização L e o segundo a distribuição de RSS dos sinais observados [20]. Estes histogramas permitem obter para cada AP duas probabilidades condicionais, a probabilidade do AP ser detetado na localização L e a probabilidade do valor RSS *online* do AP ser o valor RSS observado na localização L. Cada uma destas probabilidades é adaptada ao número de APs existentes para cada localização. Através da multiplicação destas duas probabilidades condicionadas é obtida a probabilidade P(F|L) para cada localização L.

[3] *Bins* – Intervalos de valores dos dados.

2.1.3.4. *Support Vector Machine*

Os algoritmos S*upport Vector Machine* (SVM) (Figura 10) são utilizados tanto para tarefas de classificação como para tarefas de regressão. A essência do SVM é a construção de um hiperplano ótimo, de maneira a separar as diferentes classes de dados com a maior margem possível.

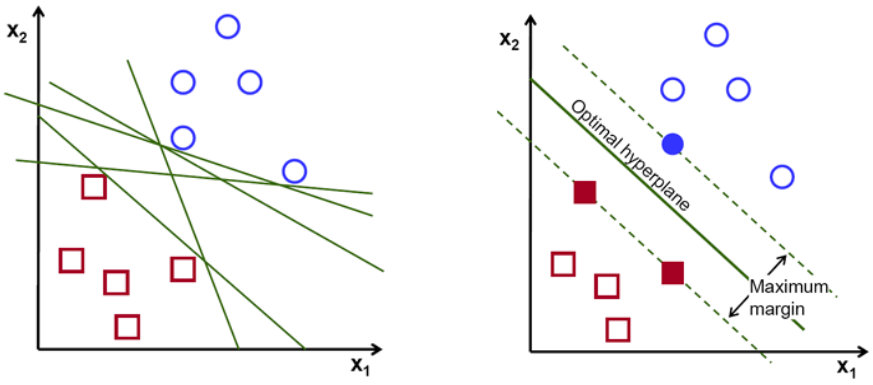

Figura 10- Algoritmo SVM [38]

Para as tarefas de classificação *Support Vector Classification* (SVC), o algoritmo consiste em tentar encontrar a maior margem de separação para diferentes classes de dados através da construção de um hiperplano (generalização do plano em diferentes números de dimensões).

As tarefas de regressão, *Support Vector Regression* (SVR), ao contrário das de classificação, permitem estimar funções de valores reais.

Em [21] é referenciado o algoritmo de classificação (SVC) com margens flexíveis, onde são efetuadas duas fases, a de treino e a de teste. Na fase de treino são fornecidas amostras de entrada e saída, cujas coordenadas são conhecidas. Esta fase de treino tem como objetivo determinar uma função que relacione as amostras de entrada e saída. Na fase de teste, são utilizadas um conjunto de amostras de teste (amostras de entrada), e através da função anteriormente definida (função determinada na fase de treino) obtém-se as saídas desejadas e os respetivos resultados. O algoritmo SVC permite obter uma

função que classifique os dados de teste, isto é, permite encontrar o hiperplano desses dados.

Também em [21] é referenciado o algoritmo SVR, e este é apresentado como uma extensão do SVC, no entanto para casos de regressão é introduzida uma *loss function*. Este algoritmo utiliza os mesmos princípios de SVC, porém apresenta algumas diferenças. Uma das principais é o facto de SVR determinar um hiperplano ótimo, onde as amostras de treino se encontram o mais próximo possível.

2.2. Tecnologias e Sistemas de Localização

Atualmente existem vários tipos de sistemas de localização baseados em tecnologias sem fios, como se pode verificar na Figura 11. Esta figura representa no eixo horizontal a resolução em metros e no eixo vertical a escala dos diferentes ambientes de localização. De seguida serão abordados alguns desses sistemas de localização, nomeadamente baseados em *Global Position System* (GPS), *Radio – Frequency Identification* (RFID), redes móveis, *Ultra – Wideband* (UWB), *Bluetooth* e *Wireless Local Area Network (*WLAN).

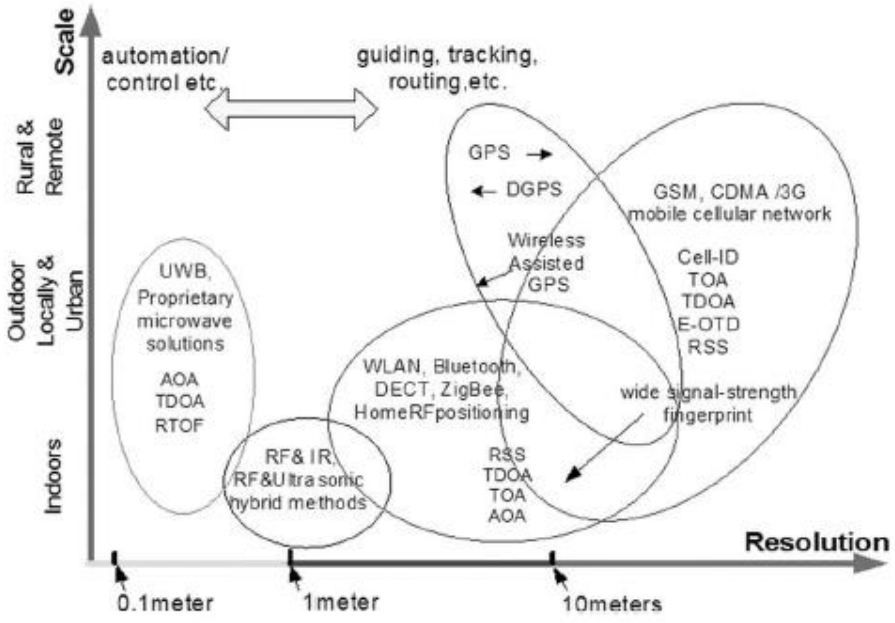

Figura 11 - Sistemas de localização baseados em tecnologias sem fios [7]

2.2.1. Global Position System (GPS)

O sistema de posicionamento global, também conhecido por GPS (*Global Position System*) é o sistema de localização *outdoor* mais conhecido e utilizado em todo o mundo. No entanto, devido à pouca cobertura do sinal de satélite a precisão deste sistema diminui em ambientes *indoor* [7].

Existem alguns sistemas de localização *indoor* baseados em GPS, como por exemplo, o *Locata* [22], [23], que tanto funciona em ambientes *outdoor* como *indoor*. *Locata* é uma tecnologia de posicionamento que foi projetado para superar as limitações de outros sistemas de posicionamento GPS no interior. Foi inventado um *transceiver* (transmissor e recetor combinados) *pseudolite*[4] sincronizado no tempo, chamado de *LocataLite*. Uma rede de *LocataLites* forma uma *LocataNet* [23], que transmite sinais de GPS semelhantes. Estes sinais de GPS, têm capacidade de permitir o posicionamento com precisão (utilizando um transportador de fase) para um dipositivo (um *Locata*).

Na tentativa de encontrar uma solução para este problema, foi desenvolvido um método denominado de GPS assistido (*Assisted*-GPS). O A-GPS utiliza um servidor local que se encontra ligado, simultaneamente, às estações base. O servidor descarrega as informações orbitais do satélite e armazena em base de dados (estações base). Um dispositivo pode conectar-se a essas estações base e descarregar a informação através de redes móveis, tais como GSM, CDMA *(Code Division Multiple Access),* WCDMA (*Wide Band Code Division Multiple Access*), LTE (*Long Term Evolution*). Durante a comunicação são recolhidas informações da propagação dos sinais e posteriormente são enviadas ao servidor A-GPS. O servidor A-GPS, com base na informação recebida e nos dados recebidos pelos satélites GPS, calcula a posição do dispositivo.

O sistema *Snap Track* [7] foi pioneiro na implementação da tecnologia baseada em A-GPS e possui uma exatidão média entre 5 a 50 metros em ambientes *indoor*.

[4] *Pseudolite* – Redução do termo "pseudo-satélite", utilizado para referir-se a transmissores situados na Terra que transmitem uma imitação de um sinal de satélite GPS.

2.2.2. RFID

A tecnologia RFID *(Radio-Frequency Identification)* permite armazenar e recuperar dados através da transmissão eletromagnética a um circuito integrado RF (*Radio Frequency*) compatível. Um sistema RFID é composto por vários componentes básicos, incluindo um número de leitores RFID, etiquetas RFID e a comunicação entre eles.

O leitor RFID é capaz de ler os dados emitidos pelas etiquetas, visto que utilizam a mesma frequência rádio e o mesmo protocolo para transmitir e receber dados. As etiquetas podem ser classificadas como passivas ou ativas.

As etiquetas passivas funcionam sem bateria, isto é, não possuem uma fonte própria de alimentação. A energia necessária para o seu funcionamento é obtida da radiação eletromagnética emitida pelo leitor, por indução eletromagnética. Desta forma, o funcionamento do sistema está limitado ao alcance do leitor.

As etiquetas ativas possuem um emissor/recetor de RF ativo alimentado por baterias. Os sistemas que utilizam este tipo de etiquetas podem obter um alcance de alguns quilómetros de comunicação entre o leitor e a etiqueta. E é esta a principal diferença entre a tecnologia RFID passiva e ativa, o alcance de leitura.

A vantagem da técnica RFID é permitir a identificação automática de pessoas e objetos à distância, através de ondas de rádio.

Um sistema para localizações *indoor* com base em etiquetas RFID ativas é o LANDMARC [7], [24]. Trata-se de um sistema constituído por leitores RFID, etiquetas ativas implementadas nos dispositivos a localizar, e um servidor que realiza a comunicação com os leitores e realiza os cálculos das localizações. Os leitores RFID são capazes de ler a informação emitida pelas etiquetas RFID.

Para este sistema, a fim de aumentar a precisão sem que sejam introduzidos mais leitores, foram implementadas etiquetas de referência. Estas etiquetas de referência funcionam como pontos de referência numa posição fixa, auxiliando na calibração do sistema. Para localizar um dispositivo equipado com uma etiqueta RFID, os leitores medem os valores RSS do dispositivo e os valores RSS das etiquetas de referência mais próximas. Essa informação é enviada ao servidor, e este por sua vez aplica o algoritmo kNN estimando assim a localização do dispositivo.

2.2.3. Redes Móveis

Através das redes móveis (GSM, CDMA, 3G, 4G e LTE) é possível realizar a localização em ambiente *indoor* se o edifício possuir uma cobertura de várias estações base, ou uma estação base em que a potência recebida (RSS) pelos dispositivos seja elevada.

Em [25] os autores apresentam um sistema de localização *indoor* baseado na tecnologia GSM. Este sistema utiliza *fingerprints* com os valores de potência do sinal recebido das várias estações base GSM. As *fingerprints* GSM incluem os valores de seis células com maior potência de sinal, e incluem ainda valores das células adicionais, a maioria dos quais são demasiado fracos para serem usados em comunicações eficientes, mas suficientemente fortes para serem detetados.

Os autores realizaram testes para três edifícios diferentes com vários pisos, aplicando o algoritmo *k-nearest-neighbor* (kNN). Os resultados obtidos mostraram que o sistema de localização conseguia diferenciar as localizações entre os vários pisos e desta forma conseguiram obter uma exatidão média de 2.5 metros em localizações num único piso.

2.2.4. UWB

O UWB (*Ultra-Wide-Band*) [7] baseia-se no envio de pulsos com um *duty cycle* baixo.

A localização baseada em UWB explora as características do tempo de sincronização das comunicações UWB, para obter uma maior precisão na localização interna (20 cm). Desta forma torna-se adequada para sistemas de localização em 2-D e 3-D. Nos sistemas de localização a 3-D são utilizados dois meios de medições [7]: TDoA, que mede a diferença do tempo de chegada de um pulso UWB a vários sensores, e AoA que calcula o respetivo ângulo de chegada dos sinais. A utilização destes dois meios tem como vantagem o aumento da exatidão do sistema. Para além disso a localização pode ser obtida a partir de apenas dois sensores, diminuindo assim a densidade de sensores em relação aos sistemas que usam apenas TDoA ou AoA.

2.2.5. Bluetooth

A tecnologia *Bluetooth* tem por objetivo eliminar os cabos para a realização de conexão entre dispositivos. Quase todos os dispositivos que são habilitados com *Wi-Fi*, também possuem o *Bluetooth* incorporado.

A tecnologia *Bluetooth* [7] opera na banda de frequência ISM (*Industrial Scientific and Medical)* de 2.4 GHz, e possui um alcance de até 100 metros. A vantagem de usar o *Bluetooth* para troca de informação é que esta tecnologia é de alta segurança, baixo custo, baixo consumo de energia e tamanho reduzido. Cada etiqueta *Bluetooth* tem uma identificação única, que pode ser utilizada para a localizar.

Uma das desvantagens da utilização da tecnologia *Bluetooth* na localização é que, para estimar a localização de um dispositivo é realizado um procedimento de descoberta, o que faz com que aumente significativamente a latência de localização (10-30 s), bem como o consumo de energia.

2.2.6. WLAN

O termo WLAN (que significa *Wireless* LAN ou *Wi-Fi*) [7] designa uma rede local sem-fios que opera com uma frequência de 2.4 GHz, e utiliza a norma IEEE 802.11. Hoje em dia, a norma IEEE 802.11 é dominante nas redes locais sem fios, e desta forma torna-se atrativa para implementações de uma infraestrutura para localização *indoor*. Muitos aeroportos, hotéis, centros comerciais, faculdades e até empresas de *fast-food* oferecem acesso público a redes *Wi-Fi*, conhecidos como "*hotspots*".

A precisão de um sistema de localização numa WLAN baseada em medições RSS, utiliza a combinação de várias variáveis.

2.2.6.1. RADAR

O sistema RADAR [26] foi um dos primeiros sistemas de localização implementados em redes *Wi-Fi*, baseado na força de sinal recebida. Este sistema é formado por um conjunto de APs, e a estimativa da posição é efetuada através de uma análise de cenário.

Na análise de cenário, são recolhidas informações que posteriormente são utilizadas na deteção da posição do dispositivo.

Os autores descrevem que quanto mais perto se está do emissor, mais forte é o sinal recebido. Estes descrevem também que o sistema utiliza duas fases, a fase *offline* onde são criados os mapas de assinaturas de cada local e a fase *online* onde são analisados os dados recolhidos no momento. Os dados recolhidos são comparados por métodos determinísticos com os dados existentes numa base de dados. Os dados armazenados nessa base de dados, são os dados dos mapas de assinaturas da fase *offline*.

Apesar de ser um dos primeiros sistemas na área, ainda hoje é o mais referenciado e um dos principais alvos de comparações sempre que surge um modelo novo.

2.2.6.2. HORUS

O sistema HORUS [27] utiliza uma estratégia baseada nos clientes, ou seja, o cálculo da localização é realizada no cliente, permitindo assim obter um maior número de utilizadores. Este sistema aumenta a performance em relação ao RADAR, pois utiliza essencialmente métodos probabilísticos para determinar a posição do dispositivo.

Os autores descrevem uma implementação com dois objetivos: obter alta precisão (através de técnicas de localização probabilísticas) e baixos requisitos computacionais (através da utilização da técnica de *clustering*[5]).

Este sistema, também apresenta duas fases, uma *offline* onde são selecionados os pontos de acesso e recolhida a força do sinal, e a fase *online* onde são utilizadas as amostras recolhidas, para estimar a localização do dispositivo.

Os autores referem que durante as experiências efetuadas, foi obtida uma exatidão de 1,40 metros em 90% dos casos.

[5] *Clustering*- técnica de *Data Mining* para fazer agrupamentos automáticos mediante o grau de semelhança.

2.2.6.3. COMPASS

O sistema COMPASS [28] utiliza algoritmos probabilísticos de posicionamento para determinar a localização de um dispositivo. À semelhança dos outros sistemas, tem duas fases. Na fase *offline* este realiza medições da força do sinal para diferentes orientações em cada ponto de referência, construindo assim o *radiomap*. Estes dados são armazenados numa base de dados como histogramas. Na fase *online*, o utilizador recolhe um conjunto de amostras com as respetivas forças do sinal recebidas, e a sua orientação. Através da combinação dos dados recolhidos na fase *online* com os dados *offline* é calculada a probabilidade para cada ponto de referência. Para cada ponto de referência apenas são selecionados histogramas de força de sinal (*radiomap*) cuja orientação é similar à orientação corrente do utilizador (*online*). Na Figura 12 são apresentados 8 pontos de referência numa área de operação exemplar, cada um dos pontos de referência têm exatamente oito histogramas de força de sinal. Apenas os histogramas de força de sinal com orientação similar ao utilizador (círculo verde) são selecionados (setas vermelhas) e utilizados no algoritmo de posicionamento.

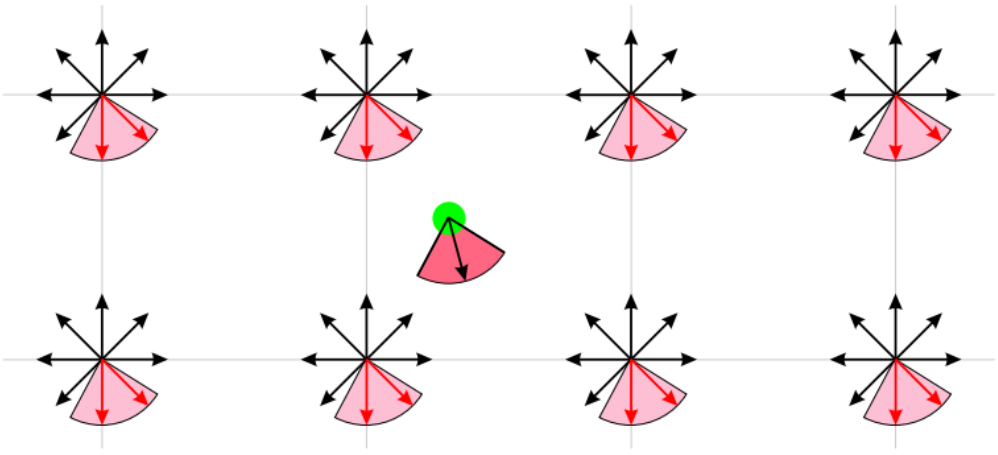

Figura 12 - Ponto de referência com oito histogramas de força de sinal [28]

Nas análises realizadas a este sistema, foi verificado que por vezes há mais do que um candidato adequado para um determinado conjunto de pontos de referência. Os pontos de referência circundantes são considerados como prováveis se o utilizador se encontrar no

meio destes. Mediante esta ocorrência, para determinar o ponto de referência mais provável é realizada a média dos k pontos de referência mais prováveis.

Enquanto nos sistemas anteriores são apresentadas limitações na exatidão devido aos efeitos de bloqueio causados pelo corpo humano, neste sistema este bloqueio é ultrapassado através da utilização de bússolas digitais para detetar a orientação do utilizador.

O sistema apresenta uma média de erro na determinação da posição a rondar os 1.65 metros.

2.2.6.4. HERECAST

O sistema HERECAST [29] é um pouco diferente do sistema RADAR, na medida em que a localização do dispositivo é obtida através da recolha do endereço MAC do AP, identificador único dos APs, com maior RSSI, o sistema retorna a informação referente ao mesmo. Esta informação é previamente introduzida por um utilizador na base de dados, com dados genéricos como o país, cidade, o nome do local onde o dispositivo se encontra, o nome do edifício, até dados mais específicos como a sala e andar. Quando um cliente se liga a um AP, se este estiver na base de dados do sistema, consegue obter a informação sobre o local onde se encontra, caso o AP seja desconhecido o sistema pede a este que adicione informação sobre o mesmo, sendo os clientes, os responsáveis por manter as informações atualizadas.

Um dos problemas encontrados durante o desenvolvimento do *Herecast* foi a dificuldade em manter o sistema atualizado devido às constantes alterações de APs em edifícios, podendo mesmo serem removidos ou adicionados novos. Outro problema prende-se na recolha dos dados. Se por um lado se pode concluir que este sistema é bom, no sentido que é recolhida muita informação com pouco esforço por parte dos utilizadores, pois o esforço é distribuído por vários utilizadores que inserem informação, também pode ser mau, pois podem haver utilizadores mal-intencionados que insiram informação incorreta.

2.2.6.5. ARIADNE

O sistema ARIADNE [30] tem como objetivo, determinar a localização de um dispositivo apresentando a informação na forma de coordenadas físicas, tentando ter a maior precisão possível.

ARIADNE é baseado numa rede local que necessita de pelo menos três APs de forma a funcionar corretamente. Cada AP recolhe os dados necessários de todos os utilizadores.

O sistema ARIADNE consiste em dois módulos, Figura 13 . O primeiro consiste em estimar um mapa de intensidade do sinal, sendo dado como entrada um ficheiro CAD com a descrição do ambiente (F-PLAN) e a medição da força de sinal dada por um tripleto de APs M(SA,SB,SC)(LR,t). Este tripleto de APs diz respeito a um dispositivo M localizado num determinado ponto de referencia (LR) num determinado instante de tempo *t*. O mapa de assinaturas para o local é criado através de *ray tracing*[6] e do algoritmo *simulated annealing*. O segundo módulo consiste na determinação da localização de um dispositivo M, quando é dado como entrada o mapa de intensidades de sinal estimado (SS-MAP) e a intensidade do sinal atual do tripleto de APs M(SA,SB,SC) (L,Now).

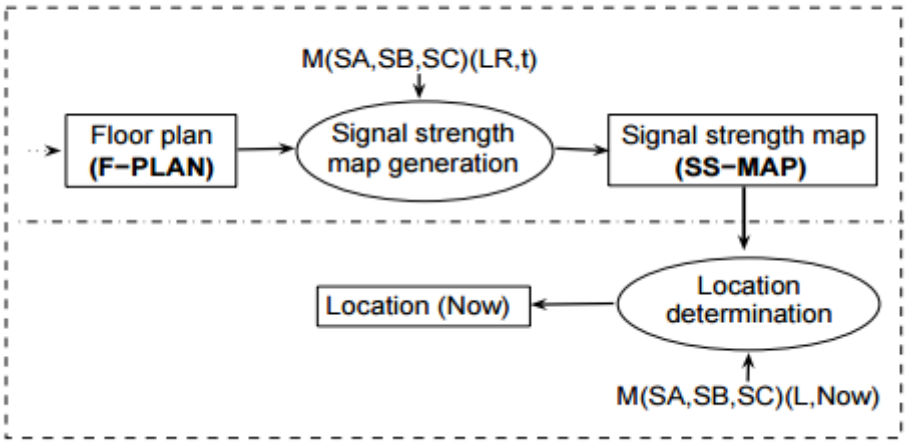

Figura 13- ARIADNE [30]

A simulação das assinaturas de um determinado ambiente pode possuir erros, pois mesmo utilizando um ficheiro CAD, existem sempre objetos e tipos de materiais

[6] *Ray tracing* – algoritmo de computação gráfica utilizado para sintetizar imagens tridimensionais.

diferentes que atenuam o sinal. Estes erros na simulação podem prejudicar na precisão do sistema de localização.

A precisão deste sistema ronda os 2 a 3 metros, sendo que os resultados são ligeiramente melhores que o sistema RADAR.

2.2.7. Comparação entre sistemas de localização

Na Tabela 2 é apresentada uma comparação entre os diferentes sistemas de localização. Para além dos sistemas abordados anteriormente, são também mencionados outros sistemas de localização, nomeadamente o *SpotOn*, o *Ubisense*, o *Sappire Dart* e o *TOPAZ*. Para cada um destes sistemas, é abordada a precisão, os princípios utilizados para a localização e o respetivo custo de implementação, sendo estas características importantes para a escolha de um sistema de localização.

Tabela 2 - Comparação dos sistemas de localização

Sistemas de localização		*Precisão*	*Princípios utilizados para localização*	*Custo*
GPS [7]		6m – 10m	ToA	Alto
	Snap Track	5m – 50m	TDoA	Alto
RFID [7], [8]		1m – 2 m	Proximidade; ToA; Modelo de propagação teórico RSSI;	Baixo
	LANDMARC	<2m	*Tags* ativas; kNN;	Baixo
	SpotON	Depende do tamanho do *cluster*	*Tags* ativas; kNN;	Baixo
Redes Móveis [7], [25]				
	GSM fingerprinting	5m	GSM (RSS); WkNN;	Médio
UWB [7]				
	Ubisense	15 cm	TDoA; AoA;	Médio alto

	Sappire Dart	<0.3m	TDoA	Médio alto
Bluetooth [7], [8]		2m – 5m	RSSI *fingerprinting*; Modelo de propagação teórico RSSI;	Alto
	TOPAZ	2m	RSS	Médio
WLAN [7]				
	RADAR [26]	3m – 5m	RSS; kNN;	Baixo
	HORUS [27]	2m	RSS; Métodos Probabilísticos; Técnica de *clustering*;	Baixo
	COMPASS [28]	<2m	Métodos Probabilísticos; Orientação do dispositivo;	Baixo
	HERECAST [29]	--	Identificador único (MAC)	Baixo
	ARIADNE [30]	2m – 3m	CAD; *Ray tracing; Simulated annealing;*	Baixo

Capítulo 3

Algoritmos de Localização para *Wi-Fi Fingerprinting*

Os algoritmos que utilizam a técnica de localização baseada em *fingerprinting* são classificados em dois tipos, determinísticos e probabilísticos. A estimação da posição destes algoritmos é efetuada na fase *online* da técnica *Wi-Fi fingerprinting*. Os algoritmos determinísticos estimam a localização de um dispositivo através das distâncias no domínio dos sinais, de forma a obter o ponto correspondente à *fingerprint* em estudo. Enquanto que, os algoritmos probabilísticos estimam a localização de um dispositivo através de funções de probabilidade.

3.1. Intensidade do sinal *Wi-Fi* recebido

RSS (*Received Signal Strength*) e RSSI (*Received Signal Strength Indicator*) são unidades de medida diferentes, no entanto representam ambas a força do sinal. A diferença é que RSSI é um índice relativo, enquanto RSS (medido em dBm) é um número absoluto que representa níveis de potência em mW (*milliwatts*).

Neste projeto são utilizados dois *datasets*: o *dataset* de treino, utilizado como mapa de rádio (dados de treino), e o *dataset* de teste que contém os dados recolhidos na fase *online*. Os APs observados nos *datasets* possuem uma vasta gama de valores RSS, estes possuem valores que vão desde os -104dBm até os 0dBm e os APs não observados tem como valor 100, isto porque os valores ideais para os APs são todos valores negativos. Ao referir que os valores negativos são os ideais para os APs observados não quer dizer que todos valores negativos sejam valores aceitáveis, pois os melhores sinais são aqueles que se

aproximam mais de 0dBm. Na Tabela 3 são apresentadas as intensidades de sinal aceitáveis.

Tabela 3 - Intensidades do Sinal [31]

Intensidade do Sinal	*Classificação*
[0 dBm ; -33 dBm]	Excelente
[34 dBm ; -67 dBm]	Muito Bom
[-68 dBm ; -77 dBm]	Regular
[-78 dBm ; -85 dBm]	Mau
< -90 dBm	Péssimo

No *dataset* de teste assim como no *dataset* de treino, os APs que não são observados, ou seja, os que não apresentam qualquer sinal, são considerados com o valor 100. Em alguns dos algoritmos apresentados em seguida, este valor 100 é substituído por um *defRSSI*, um valor RSSI por defeito que seja considerado como bastante fraco. Foram experimentados valores entre -90 e -110.

3.2. Algoritmos Determinísticos Implementados

Foi implementado um algoritmo determinístico, nomeadamente uma interpretação ao algoritmo hierárquico [6]. Também foram implementadas algumas alterações a este algoritmo, dando origem a três novas formas de calcular a similaridade e uma de calcular o *centroid*.

3.2.1. Algoritmo Hierárquico [6]

A pesquisa e o desenvolvimento da localização *indoor*, tem captado cada vez mais atenção em todo mundo [6]. A competição EvAAL [5] organizada pela conferência IPIN (*Internattional Conference on Indoor Positioning and Indoor Navigation*) 2015, permitiu a vários grupos a apresentação das suas propostas para preencher as lacunas ainda existentes nas soluções de *Wi-Fi fingerprinting*. Em [6] são apresentadas duas abordagens

para estimar as posições associadas a cada uma das amostras presentes no *dataset* de teste. A primeira abordagem apresentada trata-se de uma abordagem hierárquica com base na filtragem, regras de maioria e estimativa de *k-Nearest Neighbors,* onde são estimados o edifício, piso e coordenadas, um de cada vez. A outra abordagem é baseada na estimativa *Weighted k-Nearest Neighbors*. Nesta dissertação será apresentada uma interpretação à primeira variante da primeira abordagem mencionada em [6], que servirá como referência para a análise de desempenho das novas propostas apresentadas.

Esta secção é composta por duas subsecções. A primeira apresenta a descrição geral a esta abordagem do algoritmo hierárquico. A segunda apresenta uma variante a este algoritmo com propostas de alterações ao mesmo.

3.2.1.1. Descrição do Algoritmo Hierárquico: Filtragem, regra da maioria, *k-Nearest-Neighbor*

Este algoritmo determinístico é considerado como um algoritmo hierárquico pois, segue três importantes etapas até à conclusão da estimativa da localização.

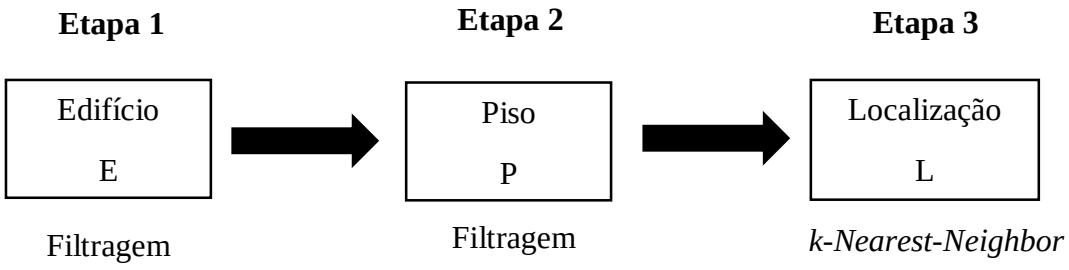

Figura 14 - Etapas do algoritmo hierárquico

Assim como é percetível na Figura 14, este algoritmo começa por estimar o edifício (*E*), seguido do piso (*P*) e por fim a localização (L). Estas etapas são efetuadas para cada *fingerprint* ($FP_{i_{teste}}$) da base de dados *online* (*DataTest)*, Tabela 4.

Tabela 4- Estrutura do conjunto DataTest (Online)

Ap1	…	*Ap520*
RSSI	RSSI	RSSI
…	…	…
RSSI	RSSI	RSSI

Etapa 1: Descoberta do Edifício

Nesta abordagem são seguidos uma série de passos para a estimativa do edifício (*E*). Para cada *fingerprint* de teste selecionada são realizados os passos descritos de seguida:

1. Para a *fingerprint* ($FP_{i_{teste}}$) é selecionado o AP mais forte (AP_{teste});

Considerando uma *fingerprint* de teste $FP_{0_{teste}}$, é determinado o AP mais forte desta, ou seja, de entre os valores de RSSI aceitáveis (valores negativos) é encontrado aquele com a melhor intensidade. O AP mais forte é o AP que possui o melhor valor de RSSI.

2. Em seguida, é percorrido o *dataset* de *treino* (*DataTraining*, Tabela 5), e descoberto o AP mais forte de cada *fingerprint* (AP_{treino});

Tabela 5 - Estrutura do DataTraining (Offline)

Ap1	…	*Ap520*	*Long*	*Lat*	*Piso*	*Edifício*
RSSI	RSSI	RSSI	x	y	P1	E1
…	…	…	…	…	…	…
RSSI	RSSI	RSSI	x	y	Pn	En

3. Através das amostras onde o AP mais forte é o AP identificado no passo 1 ($AP_{\text{teste}} = AP_{\text{treino}}$) é construído um novo subconjunto do anterior;

Em cada amostra de treino são comparados os dois APs, o AP mais forte de teste e o AP mais forte de treino. No caso de estes serem iguais, essa amostra é guardada num subconjunto *Building*. Se porventura existir mais do que um AP mais forte, isto é, mais

do que um AP que possua o mesmo valor RSSI (o melhor valor RSSI), têm de ser comparados todos esses APs. Considerando que existem dois APs mais fortes numa *fingerprint* de treino, inicialmente é verificado se o AP mais forte de teste é igual a um dos APs mais fortes de treino. No caso dos APs serem iguais, a amostra é guardada, caso contrário é descartada. Logo de seguida é verificado o outro AP mais forte, seguindo a mesma estratégia.

A utilização desta metodologia do AP mais forte permite filtrar as amostras de treino, tornando-se uma mais valia.

4. Se não existir nenhuma amostra em que isso aconteça, são repetidos os passos 1, 2 e 3 para o 2º AP, 3º AP mais forte, e assim sucessivamente, até que o subconjunto *Building* contenha pelo menos uma amostra;

Na utilização da metodologia do AP mais forte, pode existir casos em que não exista nenhuma amostra onde o AP mais forte de teste seja igual ao AP mais forte de treino. Para este caso, são repetidos novamente os passos anteriores. Nesta repetição, é agora considerado o 2º AP mais forte, tanto de teste como de treino, visto que o primeiro já foi analisado. Esta estratégia é utilizada até que existam amostras onde isso se verifique.

5. No subconjunto *Building* (criado anteriormente) é utilizada a regra da maioria, onde são contadas o número de amostras associadas a cada edifício. O edifício que apresentar o maior número de ocorrências será o escolhido. Escolhido o edifício, são guardadas num novo subconjunto (*BuildingB*) as amostras do conjunto *DataTraining* cujo edifício é o edifício anteriormente estimado.

Etapa 2: Descoberta do Piso

Também para a estimativa do piso (*P*) são seguidos um conjunto de passos:

1. É construído um subconjunto *BuildingB* do *dataset* de *training*, com todas as amostras onde o edifício é E (edifício estimado anteriormente), filtrando deste modo as *fingerprints* que pertencem ao edifício *E*;

2. Através do subconjunto *BuildingB* é criado um novo (subconjunto *BuildB),* com todas as amostras onde o AP mais forte (AP_{treino}) é igual a um dos três primeiros APs mais fortes de teste ($AP_{0_{teste}}$, $AP_{1_{teste}}$ ou $AP_{2_{teste}}$);

Aqui é necessário verificar todos os APs mais fortes de treino, no caso de existir mais do que um AP com o melhor sinal RSSI. Considerando uma *fingerprint* de treino com dois APs mais fortes, para esta *fingerprint* terão de ser verificados estes dois APs. Primeiro é verificado para um dos APs mais fortes de treino, se é igual a um dos três APs fortes de teste. No caso se ser igual, a *fingerprint* é guardada, caso contrário é descartada. Em seguida é utilizada a mesma estratégia, mas para o outro AP forte de treino.

3. É verificado se o subconjunto *BuildB* possui alguma amostra, se possuir é utilizado no passo seguinte, senão é utilizado o subconjunto *BuildingB*;

4. Após a seleção de amostras, é calculada a similaridade, *Sim(),* (equação (9)), entre a *fingerprint* de teste ($FP_{i_{teste}}$) e todas as *fingerprints* do conjunto criado anteriormente. É possível que com a estratégia anterior não existam amostras em que tal acontece, então é considerado para o cálculo da similaridade o subconjunto anterior a esse (*BuildingB*). A similaridade é a diferença entre dois objetos (vetores) ou entre dois grupos de objetos ("*clusters*"). Quanto menor o valor da similaridade entre as *fingerprints*, significa que mais semelhantes são as duas amostras.

$$Sim(fp_1, fp_2) = \frac{1}{N} \times distância\ Manhattan - 2 \times Cm \qquad (9)$$

$$DManhattan = \sum_{i=1}^{n} |RSSIfp1_i - RSSIfp2_i| \qquad (10)$$

Onde: fp_1 é a *fingerprint* de teste e fp_2 a *fingerprint* de treino; N é o número total de APs observados na fp_1 e fp_2; Cm o número de APs em comum, isto é, os APs que foram

observados tanto na fp_1 como na fp_2; A distância de *Manhattan* é dada pela equação (10).

A similaridade utilizada, equação (9), utiliza como parâmetros o número de APs comuns e o número total de APs. A utilização destes parâmetros permite que o valor da similaridade seja menor para amostras de teste mais semelhantes à amostra de treino. Quanto maior for o número de APs em comum, menor será o valor da similaridade. Aquando do cálculo da similaridade, a intensidade do sinal dos APs não observados é substituída por um valor por defeito (*defRSSI*), valor esse que deve ser suficientemente fraco. Na tentativa de obter melhorias na fórmula de similaridade foram propostas três outras formas de a calcular, variando entre a distância Euclidiana e a distância de *Manhattan*. A escolha destas duas distâncias, baseou-se nos resultados apresentados para a segunda abordagem descrita em [6]. A distância de *Manhattan* era a distância utilizada no algoritmo original, a distância Euclidiana foi escolhida baseada nos resultados apresentados para a segunda abordagem de [6]. Nesta segunda abordagem, são apresentados testes para diferentes funções de distância, nomeadamente a distância de *Manhattan*, de *Euclidean*, de *Chesbyshev* e de *Canberra*. Entre estas quatro, a distância de *Euclidean* era a que apresentava melhores resultados. Para além disso estas duas distâncias são as mais comuns.

5. As *k1* amostras do subconjunto que apresentarem maior similaridade à $FP_{i_{teste}}$ são selecionadas e guardadas num subconjunto *Floor*;

6. Entre as amostras selecionadas anteriormente, e mais uma vez utilizando a regra da maioria, são contadas o número de amostras associadas a cada piso. O piso que apresentar um maior número de amostras é o piso *P*;

Etapa 3: Descoberta da Localização

Por fim, para a estimativa da localização (*L*), é percorrido o subconjunto *Floor* e criado um novo subconjunto deste (*FloorF)*, com todas as amostras onde o piso é *P* (piso estimado anteriormente), filtrando deste modo as *fingerprints* que pertencem ao piso *P*.

É verificada a similaridade entre a $FP_{i_{teste}}$ e todas as *fingerprints* do subconjunto *FloorF* (subconjunto criado anteriormente).

As *k2* amostras do subconjunto que apresentarem maior similaridade à $FP_{i_{teste}}$ são selecionadas e guardadas num subconjunto *Coordinates*.

A Localização (*L*) é dada através do *centroid* das coordenadas de todas as amostras selecionadas. O *centroid* é associado a uma forma geométrica conhecida como centro geométrico, ou seja, através das coordenadas das *k2* amostras é determinado o centro, sendo esta a localização final. A Figura 15 representa o *centroid* dos pontos de acesso A, B e C.

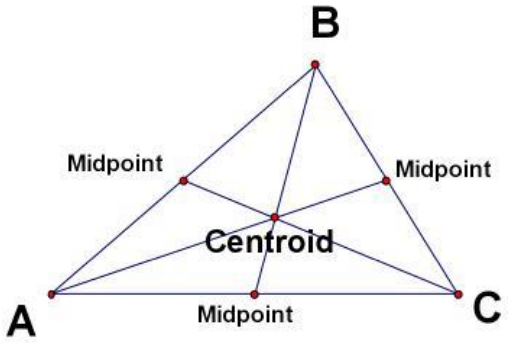

Figura 15 - Representação do centroid de um triângulo [32]

O *centroid* foi calculado recorrendo a duas estratégias, a estratégia utilizada no algoritmo original proposto em [6], ou seja, o S*imple Centroid*, calculado usando a equação (11). Este utiliza o método kNN e consiste na média aritmética das coordenadas das *k2* amostras, amostras que correspondem aos *k* vizinhos mais próximos. A estratégia introduzida como alternativa ao cálculo do *centroid* é o *Weighted Centroid.* Nesta estratégia são atribuídos "pesos", equação (12) , mediante o valor da distância. Quanto menor a distância, significa que esse vizinho se encontra mais próximo. O peso é dado pela razão do inverso da distância Euclidiana ao quadrado de cada amostra, e o inverso da soma do inverso da distância Euclidiana das *k2* amostras. A localização é estimada através da razão entre soma das longitudes e latitudes multiplicadas pelo peso, e a soma de todos os pesos.

$$\left(long_x, lat_y\right) = \sum_{i=1}^{k} \frac{1}{k} (long_i, lat_i) \quad (11)$$

$$w = \frac{\frac{1}{D^2}}{\sum_{i=1}^{k} \frac{1}{D^2}} \quad (12)$$

3.2.1.2. Variante do algoritmo hierárquico

Partindo da abordagem anterior, são propostas algumas alterações. Consideram-se dois itens, suscetíveis de alteração: a função de similaridade e o cálculo do *centroi*d. Sendo assim., são então apresentadas as alterações propostas à abordagem do algoritmo hierárquico, relativamente ao cálculo da similaridade, que faz parte do passo 4 da etapa 2.

A função de similaridade utilizada na variante original, anteriormente descrita, utiliza a distância de *Manhattan*. Esta distância é definida como a soma das diferenças entre x e y em cada dimensão [33]. Entre as funções de distâncias de similaridade mais comuns, além da distância de *Manhattan*, temos também a distância Euclidiana. Esta última é definida como a soma da raiz quadrada da diferença entre x e y nas suas respetivas dimensões [33].

Em [34] é apresentada uma analogia entre estas duas distâncias, onde geometricamente a distância de *Euclidean* é associada à hipotenusa de um triângulo e a distância de *Manhattan* à soma dos catetos. Seguindo a analogia descrita em [34], podemos imaginar uma rota de GPS para dois veículos diferentes, um carro e um helicóptero. A distância Euclidiana seria o segmento de reta que indicaria uma possível rota para o helicóptero, onde não haveriam preocupações com as ruas (tratando-se de um veículo aéreo) e a distância de *Manhattan* seria um segmento de retas na vertical e na horizontal, semelhante à rota de um carro, pois este é obrigado a respeitar o sentido das ruas.

Nas propostas de alterações às funções de similaridade apresentadas, foram utilizadas exatamente estas duas distâncias, na sua forma original e também com a utilização de alguns parâmetros adicionais, que deram origem às três seguintes fórmulas de similaridade.

Similaridade Euclidiana com parâmetros adicionais

Esta proposta de cálculo da similaridade é idêntica à abordagem original, diferindo apenas na distância utilizada. Nesta $Sim()$, equação (13), a distância utilizada é a distância Euclidiana, equação (14).

$$Sim(fp_1, fp_2) = \frac{1}{N} \times distância\ Euclidiana - 2 \times Cm \quad (13)$$

$$DEuclidiana = \sqrt[2]{\sum_{i=1}^{n} (RSSIfp1_i - RSSIfp2_i)^2} \quad (14)$$

Similaridade Euclidiana

A proposta de cálculo de similaridade $Sim()$, equação (15), é apenas dada pela distância Euclidiana.

$$Sim(fp_1, fp_2) = \sqrt[2]{\sum_{i=1}^{n} (RSSIfp1_i - RSSIfp2_i)^2} \quad (15)$$

Similaridade de *Manhattan*

A proposta de cálculo de similaridade $Sim()$, equação (16), é apenas dada pela distância de *Manhattan*.

$$Sim(fp_1, fp_2) = \sum_{i=1}^{n} |RSSIfp1_i - RSSIfp2_i| \quad (16)$$

O segundo item de alteração diz respeito ao cálculo do *centroid*. No algoritmo original o cálculo do *centroid* é efetuado recorrendo aos *k-Nearest-Neighbors*. A proposta de alteração é a utilização do algoritmo *Weighted-k-Nearest-Neighbor*, como alternativa ao kNN. O algoritmo WkNN é um refinamento do algoritmo kNN, onde tem por objetivo pesar os *k* vizinhos de acordo com a distância até ao ponto de referência. Aos vizinhos que se encontrarem mais próximos do ponto referência, é-lhes atribuído um maior peso w_i. O peso w_i é definido como a razão entre o inverso ao quadrado da distância Euclidiana de cada um dos *k* vizinhos e o inverso da soma ao quadrado das distâncias Euclidianas de todos os *k* vizinhos.

3.3. Algoritmos Probabilísticos Implementados

O algoritmo *kPM* (*k Probabilistic Method*) tem como principal intuito determinar a probabilidade das diferentes posições e selecionar as amostras mais prováveis baseado nessas probabilidades. Existem várias metodologias utilizadas para o cálculo desta probabilidade. Mediante essas metodologias, são apresentadas três variantes para os algoritmos probabilísticos. A primeira é fundamentada numa distribuição baseada em histogramas, um por cada AP. A segunda na visibilidade dos APs num determinado ponto de referência. E a terceira baseada numa distribuição de *Kernel* para cada AP.

3.3.1. Algoritmo *kPM* : Distribuição baseada em Histograma

Este algoritmo probabilístico segue quatro importantes etapas até à conclusão da estimativa da localização.

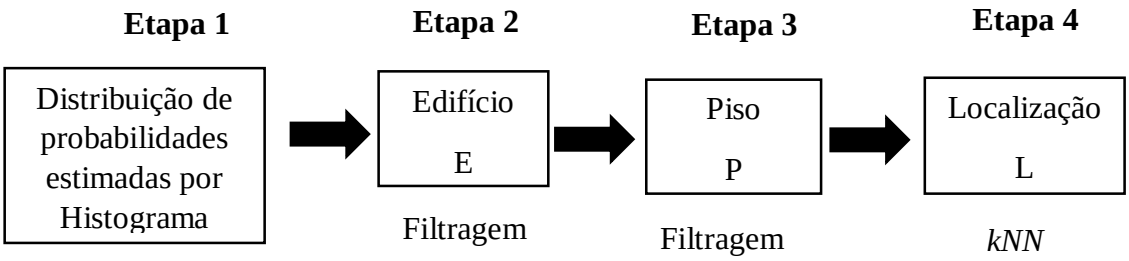

Figura 16 - Etapas do algoritmo probabilístico (Distribuição baseada em Histograma)

Assim como é percetível na Figura 16, este algoritmo começa por criar uma distribuição de probabilidades estimadas por histograma, seguido da estimativa da posição para cada *fingerprint* de dados *online* ($FP_{i_{teste}}$). Para a estimativa da posição são realizadas as etapas 2, 3 e 4, que correspondem à estimativa do edifício (*E*), do piso (*P*) e da localização (L).

Etapa 1: Construção da Distribuição baseada em Histograma

A distribuição baseada em histograma é construída com base nos dados presentes no *dataset* de treino, dados recolhidos na fase *offline*. O intuito desta distribuição de histograma é obter uma lista de diferentes posições, identificadas pela longitude, latitude, edifício e piso. Para além deste identificador, cada uma contém ainda todos os 520 APs da base de dados. Para cada um desses APs é criado um histograma que terá uma determinada probabilidade associada a cada valor RSSI (probabilidade desse valor RSSI ter sido observado nesse mesmo AP). São em seguida apresentados os passos necessários à construção desta distribuição de histograma.

Para cada Posição (*longitude, latitude, edifício, piso*), Tabela 6, do *dataset* de treino (*offline*) é calculado para cada AP_{n} um histograma ($Hist_n^P$), de acordo com equação (*17*),

$$Hist_n^P = \frac{T_n(RSSI_j)}{S_P} \qquad (17)$$

Onde: $T_n(RSSI_j)$ representa o número de vezes em que o AP_n tem o valor $RSSI_j$ e S_P representa o número total de amostras daquela Posição.

Tabela 6 - Estrutura de uma posição do subconjunto DataTraining

Posição				AP_1			...	AP_i		
Longitude	Latitude	Edifício	Piso	$RSSI_0$	…	$RSSI_j$		$RSSI_0$	…	$RSSI_j$

Desta forma, a posição P (P_P) é dada por um conjunto de histogramas para *n* APs, Tabela 7.

$$P_P = \{Hist_1^P, \dots, Hist_n^P\}$$

Tabela 7 - Distribuição baseada em Histograma da Posição P

Posição				AP_1	...	AP_n
Longitude	Latitude	Edifício	Piso	$Hist_1$		$Hist_n$

Etapa 2: Descoberta do Edifício

São seguidos uma série de passos para a descoberta do edifício (*E*), estes passos são realizados para todas as *fingerprints* presentes no *dataset* de teste (*online*). As *fingerprints* são selecionadas individualmente, e para cada uma delas ($FP_{i_{teste}}$), são realizados os passos descritos em seguida.

1. É calculada a probabilidade, equação (18) , para cada Posição;

$$Prob^P = \prod_{i=1}^{n} ProbHist(RSSI_i^{teste}) \quad (18)$$

$$ProbHist(RSSI_i^{teste}) = \sum_{j=(RSSI_i^{teste}-r)}^{(RSSI_i^{teste}+r)} RSSI_j \quad (19)$$

Esta probabilidade, $Prob^P$, é dada pelo produto das probabilidades de cada AP_i. A probabilidade de cada AP_i é dada pela equação (19), onde é efetuado o somatório das probabilidades presentes na distribuição de histograma para um determinado intervalo de valores RSSI. Esse intervalo vai desde o valor $RSSI_i^{teste}$-r até ao valor $RSSI_i^{teste}$+r, (r é um valor ajustável e $RSSI_i^{teste}$ corresponde ao valor RSSI da *fingerprint* de teste).

A proposta inicial desta variante era simplesmente, para cada AP, ir buscar ao histograma a probabilidade do valor RSSI (valor observado na *fingerprint* de teste). A probabilidade desta posição seria dada pela multiplicação dessas probabilidades de cada AP. No entanto após a sua implementação, os resultados obtidos eram insatisfatórios. Imaginando um valor RSSI de -60dBm para a *fingerprint* de teste, e um histograma com os valores -56dBm, -57dBm, -59dBm, -62dBm e -63dBm (estes valores são aqueles em que a probabilidade é diferente de zero). Desta forma ao ir buscar a probabilidade ao histograma do valor -60dBm (valor observado na *fingerprint* de teste) esta seria nula. No entanto o histograma possui probabilidades para valores de RSSI bastante próximos do RSSI de teste. Após esta verificação foi necessária uma alteração ao algoritmo, alteração essa que se baseou na utilização de intervalos de valores. Agora a probabilidade do AP não seria dada apenas pela probabilidade do histograma desse valor, mas sim pela soma das probabilidades de um intervalo de valores, e só depois essa probabilidade seria multiplicada. Considerando que o AP_0 da *fingerprint* de teste detém um valor RSSI, então para o mesmo AP da posição *P*, a probabilidade recolhida do histograma será a soma das probabilidades dos valores RSSI-r até RSSI+r. A probabilidade desse AP é dada por um intervalo de valores RSSI, tendo por base o valor observado na *fingerprint* de teste. O valor que se obtém é a probabilidade de uma determinada posição ter o mesmo valor RSSI que a *fingerprint* de teste.

A decisão de utilizar um valor por defeito para os APs não observados surgiu pois na utilização dos intervalos se o valor RSSI fosse o valor 100 (não observado) esta estratégia não sofreria alterações nos resultados. Como não existem valores positivos e o valor 100 representa os APs não observados, a estratégia dos intervalos não iria funcionar.

Um exemplo prático desta implementação é apresentado em seguida, de forma a esclarecer melhor o que é pretendido com este método. Neste caso foi utilizado um *defRSSI*=-100dBm.

Tabela 8 - Fingerprint de teste

	AP_1	...	AP_n
fp_0	-100	…	-100

Tabela 9 - Distribuição baseada em histograma para a Posição 1

	AP_1				...	AP_n		
$Posição\ 1$	-100	…	-90	-80	…	-85	…	-100
	p_0	⋯	p_{j-1}	p_j	⋯	p_{n_0}	⋯	p_{n_j}

O objetivo é calcular a probabilidade da Posição 1 da Tabela 9. Observando a *fingerprint* de teste, Tabela 8, o AP_1 possui o valor -100. A probabilidade desse AP_1 da posição 1 será a soma das probabilidades de -100-r até -100+r. Considerando r=15 (dá um intervalo de 30 valores) a probabilidade é dada pela soma das probabilidades do valor RSSI de -115 até ao -85. Sendo assim a probabilidade para o AP_1 será a probabilidade de -100 e mais a probabilidade de -90, $ProbAP_1 = p_0 + p_{j-1} + p_j$. A probabilidade calculada é apenas a probabilidade para o AP_1 , ou seja, $Prob^P = ProbAP_1$. Mas é necessário calcular o acumulado para todos os APs, multiplicando os valores. De igual modo para o AP_n , o valor na *fingerprint* de teste é de -100, então a probabilidade desse AP para a posição 1 é, $ProbAP_n = p_{n_0} + p_{n_j}$. Desta forma a probabilidade da posição P fica com, $Prob^P = ProbAP_1 \times ProbAP_n$. Este é o método utilizado para todos os APs da *fingerprint*, obtendo no final uma probabilidade $Prob^P$ para a posição *P*.

2. Após calcular a probabilidade para cada posição *P*, o conjunto de todas as posições é ordenado decrescentemente por probabilidade e guardado num subconjunto *HistOrdered*, ficando assim em primeiro as posições que apresentam maior probabilidade. Desse subconjunto são selecionadas *k1* posições cuja probabilidade é maior, ou seja, as *k1* primeiras posições.

3. Através das *k1* posições selecionadas anteriormente e da regra da maioria é identificado o edifício E. Na regra da maioria é contado o número total de cada um dos edifícios existentes, aquele que possuir o maior número é considerado como o edifício *E*.

Etapa 3: Descoberta do Piso

Na estimativa do piso (*P*), são também seguidos um conjunto de passos, que são apresentados seguidamente.

1. É criado um subconjunto *BuildingB* do subconjunto *HistOrdered*, com as Posições cujo edifício é E (edifício estimado anteriormente), filtrando assim as posições que pertencem ao edifício *E;*
2. São selecionadas as primeiras *k2* posições e guardadas num subconjunto *Fsamples*;
3. No subconjunto *Fsamples* (criado anteriormente) é utilizada a regra da maioria para identificar o piso *P*.

Etapa 3: Descoberta da Localização

Para a estimativa da localização (*L*), primeiramente é percorrido o subconjunto *BuildingB* e criado um novo subconjunto deste (*FloorF)*. Este novo subconjunto terá todas as posições onde o piso é *P* (piso estimado anteriormente), filtrando assim as posições que pertencem ao piso *P*. Em seguida são selecionadas as primeiras *k3* posições do subconjunto *FloorF* e guardadas num subconjunto *Coordinates.*

A localização *(L)* é estimada através do cálculo do *centroid* dessas *k3* posições selecionadas. O *centroid* dessas *k3* posições é calculado através da média das longitudes e latitudes (*Simple Centroid*). Os valores de *k1, k2* e *k3* são valores ajustáveis.

3.3.2. Algoritmo *kPM*: Distribuição de Visibilidade dos APs

Este algoritmo probabilístico à semelhança do anterior, segue quatro importantes etapas até à conclusão da estimativa da localização.

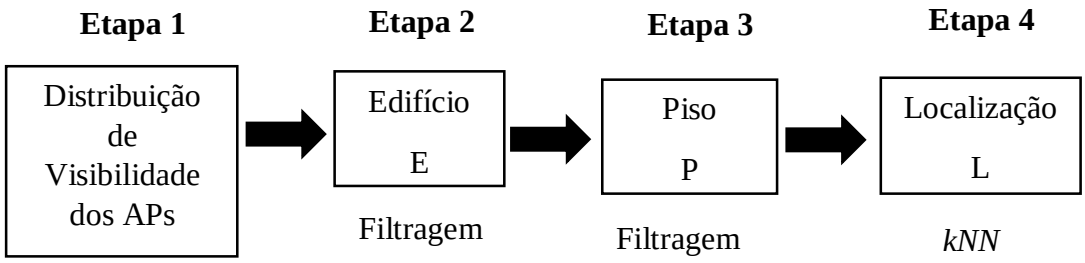

Figura 17 - Etapas do algoritmo probabilístico (Distribuição de Visibilidade dos APs)

Assim como é percetível na Figura 17, este algoritmo começa por criar uma distribuição da visibilidade dos APs, seguido da estimativa da posição para cada *fingerprint* de dados *online* ($FP_{i_{teste}}$). Para a estimativa da posição são realizadas as etapas 2, 3 e 4, que correspondem à estimativa do edifício (*E*), do piso (*P*) e da localização (L).

Etapa 1: Distribuição de Visibilidade dos APs

Esta variante do algoritmo probabilístico, começa por criar uma distribuição de visibilidade dos APs. Esta distribuição é construída com base nos dados do *dataset* de treino (dados recolhidos na fase *offline*). O objetivo desta distribuição é obter uma lista com diferentes posições (identificadas pela longitude, latitude, edifício e piso). Cada uma destas posições terá associada uma lista de probabilidades (probabilidades para os 520 APs), que consiste exatamente na probabilidade de um AP ser observado numa determinada posição. Por conseguinte, são apresentados os passos necessário à construção desta distribuição da visibilidade dos APs.

Para cada Posição (*longitude, latitude, edifício, piso*) do *dataset* de treino (*offline*), calcula-se a probabilidade para cada AP desta posição, através da equação (20);

$$Prob_n^P = \frac{obsAP_n}{S_P} \quad (20)$$

Onde: $obsAP_n$ representa o número de vezes que o AP_{n} foi observado (RSSI ≠ 100) e S_P representa o número total de amostras daquela Posição.

Para a estimativa da posição são selecionadas as *fingerprints* de teste uma a uma ($FP_{i_{teste}}$), do *dataset* de teste (*online*) e realizadas as etapas 2, 3 e 4. Cada uma desta etapas segue um conjunto de passos.

Etapa 2: Descoberta do Edifício

É calculada a probabilidade para cada posição, equação (21);

$$Prob^P = \prod Probabilidade_{APteste} \quad (21)$$

Onde: $Probabilidade_{APteste}$ é a probabilidade associada ao AP de teste observado (RSSI ≠ 100), contida na distribuição de probabilidades para a posição P.

Para calcular a probabilidade $Prob^P$, são analisados os APs observados na *fingerprint* de teste e os APs observados na posição P. Considerando que o AP_1 foi observado na *fingerprint* de teste (detém um valor RSSI diferente de 100), será recolhida a probabilidade do AP_1 na distribuição de visibilidade dos APs, associada à posição P. Esta recolha é efetuada para todos os restantes 519 APs, caso o AP seja observado na *fingerprint* de teste. A probabilidade final da posição P é dada pela multiplicação dessas probabilidades recolhidas.

Aquando da implementação desta variante, foi verificado que para algumas *fingerprints* de teste as probabilidades das posições eram nulas, influenciando na estimativa da posição. Estas probabilidades nulas resultavam do facto de um AP ser observado na *fingerprint* de teste e não ser na *fingerprint* de treino, o que tornava de imediato a probabilidade dessa posição nula. Para os casos em que a probabilidade era zero foi utilizado um valor pequeno (o valor utilizado foi 0.01). No Apêndice A pode-se encontrar uma análise detalhada que levou à tomada de decisão do uso do valor 0.01.

Em seguida é selecionada a posição mais provável e o edifício *E* estimado é considerado como o edifício dessa posição.

Etapa 3: Descoberta do Piso

1. São guardadas num subconjunto *BuildingB* as posições cujo edifício é o edifício *E* (estimado anteriormente);

2. É calculada a probabilidade para cada posição do subconjunto anterior baseada na equação (22);

$$Prob^{P} = \frac{\sum Probabilidade_{APteste}}{NonAP} \qquad (22)$$

Onde: $\sum Probabilidade_{APteste}$ é o somatório das probabilidades dos APs que foram observados na *fingerprint* de teste; *NonAP* é o número total de APs da Posição P que não foram observados na *fingerprint* de teste e o número de APs observados em teste, mas não observados em treino, ou seja, aos 520 APs existentes são subtraídos os APs observados na posição P que sejam comuns aos APs observados na *fingerprint* de teste.

Em análises realizadas foi verificado que utilizando a probabilidade calculada no edifício, algumas das posições que apresentavam maior probabilidade não correspondiam às posições mais próximas (a olho nu). Desta forma, para o cálculo da probabilidade foram utilizados mais parâmetros com o intuito de atenuar essa questão. O cálculo da probabilidade de cada AP observado na *fingerprint* de teste, tem associado um 'peso'. A probabilidade do AP é multiplicada por um valor (peso) mediante o valor RSSI do AP observado na *fingerprint* de teste, Tabela 10. O objetivo da utilização de um 'peso' é exatamente pesar aqueles APs observados com melhor valor RSSI.

Tabela 10 - 'Pesos' utilizados para os valores RSSI

Intervalo de valores RSSI	*'Peso'*
0dBm a -25dBm	200
-26dBm a -75dBm	100
-76dBm a -85dBm	15
<= -86dBm	1

A escolha destes 'pesos' foi efetuada mediante a qualidade do sinal, pois o melhor sinal é aquele cujo valor é mais próximo de 0. Visto a qualidade do sinal para o intervalo de 0dBm até aos -33dBm ser considerado excelente, para o intervalo de -34dBm até aos -67dBm muito bom, para o intervalo de -68dBm até aos -77dBm regular, para o intervalo de -78dBm até aos -85dBm mau e para sinais menores do que -90dBm péssimo, foi escolhido um ´peso´ muito maior para os valores considerados como excelentes, e um peso muito menor para os valores considerados maus e péssimos.

Após somar as probabilidades de cada AP, esse valor é dividido pelo número de APs que não foram observados na posição P, mas que foram observados na *fingerprint* de teste. Desta forma é dada maior probabilidade de ser escolhida à posição que apresentar mais APs observados em comum e menos APs observados não comuns à *fingerprint* de teste. No caso de o AP ser observado na *fingerprint* de teste, mas não na posição, a probabilidade assume o valor 0.01, mas sem nenhum peso associado.

3. São selecionadas as k1 amostras mais prováveis, e através da regra da maioria é estimado o piso *P*.

Etapa 4: Descoberta da Localização

Para a estimativa da localização (*L*), primeiramente é percorrido o subconjunto *BuildingB* e criado um novo subconjunto deste (*FloorF).* Este novo subconjunto terá todas as posições onde o piso é *P* (piso estimado anteriormente), filtrando assim as posições que pertencem ao piso *P.* Em seguida são selecionadas as primeiras *k2* posições do subconjunto *FloorF* e guardadas num subconjunto *Coordinates.*

A localização *(L)* é estimada através do cálculo do *centroid* dessas *k3* posições selecionadas. O *centroid* dessas *k2* posições é calculado através da média das longitudes e latitudes (*Simple Centroid*). Os valores de *k1*, *k2* são valores ajustáveis.

3.3.3. Algoritmo *kPM*: Distribuição de *Ker*nel

Também este algoritmo probabilístico, segue quatro importantes etapas até à conclusão da estimativa da localização.

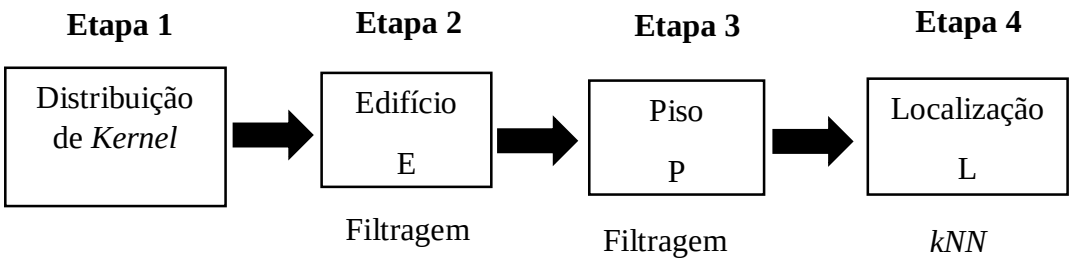

Figura 18 - Etapas do algoritmo probabilístico (Distribuição de Kernel)

Este algoritmo, Figura 18 , começa por criar uma distribuição de *Kernel*, seguido da estimativa da posição para cada *fingerprint* de dados *online* ($FP_{i_{teste}}$). Para a estimativa da posição são realizadas as etapas 2, 3 e 4, que correspondem à estimativa do edifício (*E*), do piso (*P*) e da localização (L).

Etapa 1: Distribuição de Kernel

Como foi mencionado no capítulo 2, para os métodos probabilísticos existem dois métodos para estimar a função de probabilidade, o método do histograma, já explorado no algoritmo probabilístico, que utiliza uma distribuição baseada em histograma, e o método de *Kernel*. Este algoritmo é baseado neste último método.

Sendo assim, inicialmente é criada uma distribuição geral para cada posição do *dataset* de treino, com a média μ dos valores RSSI para cada AP.

Para a estimativa da posição são selecionadas as *fingerprints* de teste uma a uma ($FP_{i_{teste}}$), do *dataset* de teste (*online*) e efetuados um conjunto de passos.

Etapa 2: Descoberta do Edifício

1. É calculada a probabilidade para cada posição, equação (21);

$$Prob^P = \frac{1}{n}\sum_{i=1}^{n} K(RSSI_{teste};\ \mu) \qquad (23)$$

Onde: *n* é número de APs da *fingerprint* e $K(RSSI_{teste};\ \mu)$ a função de *kernel* [35].

A probabilidade $Prob^P$ é dada pela média das probabilidades dos 520 APs. A probabilidade de cada AP é determinada através da função de *kernel*, mais propriamente a função *Gaussiana,* equação (24).

$$K(RSSI_{teste};\ \mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{(RSSI_{teste}-\mu)^2}{2\sigma^2}\right)} \qquad (24)$$

Esta função tem como intuito dar maior probabilidade os APs cujo valor RSSI observado na *fingerprint* de teste seja igual ao valor RSSI observado na posição *offline*. Caso os valores sejam iguais, o valor do exponencial do número de *Euler* será zero, resultando no valor 1, valor máximo que se poderá obter. Caso os valores sejam ligeiramente diferentes, ou até bastante diferentes, o exponencial será menor que 1, logo esse AP terá menor probabilidade, tal como é suposto.

Para esta abordagem foi utilizado um valor ajustável para o desvio padrão σ e não o desvio padrão definido pela raiz quadrada da variância. No caso das *x* amostras de uma determinada posição possuírem o mesmo valor, no cálculo do desvio padrão a subtração do valor RSSI irá coincidir com a média, obtendo um desvio padrão nulo.

Considerando uma determinada posição P que tem exatamente 3 amostras, e o AP1 tem precisamente os valores RSSI presentes na Tabela 11.

Tabela 11 - Amostras de uma posição P para o AP1

AP1
-105
-105
-105

A média μ para este AP seria exatamente -105. E o desvio padrão σ seria,

$$\sigma = \sqrt{\frac{(-105-\mu)^{2} + (-105-\mu)^{2} + (-105-\mu)^{2}}{3}}$$

Como a média é igual ao valor que está a ser subtraído, o σ seria nulo. Assim, foi utilizado um valor ajustável, como é sugerido em [11].

Em [11] é dito que os valores para o parâmetro σ, são valores baixos. Nesta variante implementadas foram utilizados valores altos, ao contrário do que é descrito em [11]. Ao longo do desenvolvimento desta variante foram surgindo vários problemas, sendo um deles exatamente o valor do parâmetro σ.

A função gaussiana utiliza a função exponencial, cuja base é o número de *Euler*. Considerando o caso de e^{-x}, à medida que o expoente do número de *Euler* vai aumentando, menor é o valor de y (tende para zero), e à medida que o expoente vai diminuindo, maior é o valor de y (tende para infinito), Figura 19.

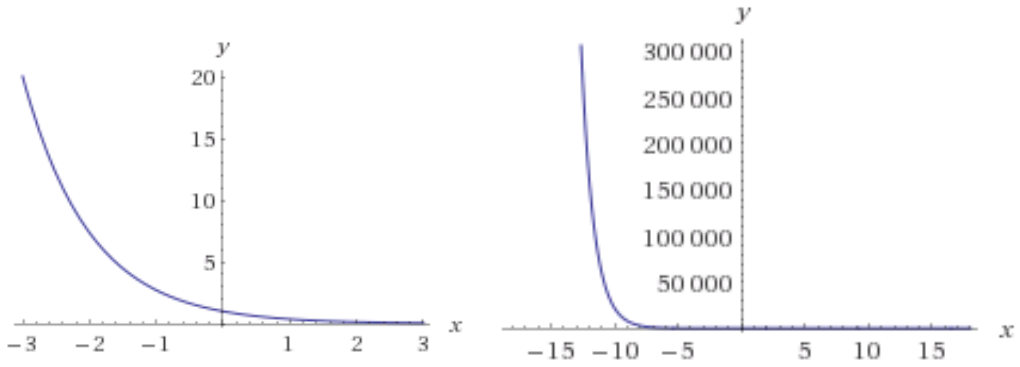

Figura 19 - Com x de -3 a 3 (lado esquerdo) e com x de -15 a 15 (lado direito)

Assim, tendo em conta o exponencial utilizado na função *gaussiana* (24), se o expoente tender para zero, toda a probabilidade dessa posição será zero. E o expoente tende para zero quando o expoente aumenta, logo para que tal não aconteça o valor do expoente deve ser pequeno. Visto que o expoente aqui se trata de uma fração negativa, para que esse valor seja pequeno o valor do denominador deve também ele ser pequeno, o que reforça aqui as conclusões tiradas em [11]. No entanto a nível computacional, para o método *java.lang.Math.exp()*, se o argumento for infinito negativo o resultado é zero positivo [36]. Se o resultado é zero, mais uma vez, toda a probabilidade dessa posição será zero. De forma a contornar esta questão, foram utilizados valores maiores para o parâmetro σ.

Considerando um caso em que um AP tem $RSSI_{teste}$ de -105dBm e μ de -104,65dBm, com estes valores e com um $\sigma = 0.01$, para a exponencial é obtido um resultado de 0, com um $\sigma = 10$, é obtido um resultado de $1{,}36 \times 10^{-6}$. Apesar de em teoria, para valores maiores de σ, o resultado tender para zero, é aceitável na prática, pois tendem para zero, mas não são completamente zero. Aqui não poderia ser utilizado um valor pequeno para o parâmetro σ, pois acabava por descartar esta posição, que como se pode verificar possui valores RSSI muito próximos à média da *fingerprint* de teste neste AP.

2. Após calcular as probabilidades de cada posição, estas são ordenadas. Depois é selecionada a posição mais provável, e é através desta que é identificado o edifício (*E*);

Etapa 3: Descoberta do Piso

São guardadas, num novo subconjunto, todas as posições pertencentes ao edifício *E*, e das *k1* amostras selecionadas é identificado o piso (*P*) através da regra da maioria;

Etapa 4: Descoberta da Localização

São guardadas, num novo subconjunto, todas as posições pertencentes ao piso *P*, e destas selecionar *k2* posições para o cálculo do *centroid*, obtendo assim a localização *L*. O *centroid* dessas *k2* posições é calculado através da média das longitudes e latitudes (*Simple Centroid*). Os valores de *k1*, *k2* são valores ajustáveis.

Capítulo 4

Implementação

Este capítulo descreve a implementação dos algoritmos de localização. Os algoritmos de localização descritos anteriormente foram implementados em linguagem Java e testados com os *datasets* da competição EvALL.

A escolha da linguagem *java* para a implementação foi baseada nos projetos de localização desenvolvidos anteriormente. Estes projetos [2] [1] [3] [4], que foram utilizados como referência neste trabalho, foram também desenvolvidos em linguagem *java*.

4.1. Cenário de Teste

A localização baseada em *Wi-fi fingerprinting* consiste em duas fases distintas, que quando complementadas permitem descobrir a localização de um dispositivo.

Como já foi referido, são utilizados dois *datasets* de dados reais, nomeadamente o *dataset training* e o *dataset validation.* O *dataset training* diz respeito ao mapa de rádio (fase *offline*), composto por dados recolhidos à *priori* (dados de treino). O *dataset validation*, diz respeito aos dados atuais do utilizador (dados de teste).

Neste projeto, foi decidido guardar estes dois conjuntos de *fingerprints* em memória. Antes de descrever como foram carregados para memória é apresentada uma breve descrição sobre a forma como cada um deles é construído.

Na fase *offline* é realizada uma análise de cenário e construída uma base de dados de *fingerprints.* Num determinado ambiente *indoor* são recolhidas, em diferentes pontos de referência, um conjunto de amostras de valores RSS para cada ponto de acesso (AP) disponível naquele local [37]. A posição geográfica de cada ponto de referência $RP(x, y)$ é também recolhida.

Através das *n* amostras recolhidas para cada AP num determinado ponto de referência RP, é produzida a *fingerprint*. Esta *fingerprint* é construída com base na média dos valores RSS das *n* amostras recolhidas para cada AP. Desta forma, a *fingerprint* fica com o seguinte aspeto,

$$fp = \{RSSI_{1RP}, \dots, RSSI_{jRP}\}$$

Onde, $RSSI_{jRP}$ corresponde à média dos valores RSS recolhidos no AP_j e no ponto de referência RP.

Todas as *fingerprints* contruídas são depois armazenadas numa base de dados, dando origem ao mapa de rádio *offline*.

Na fase *online* são recolhidas um conjunto de *fingerprints* pelo dispositivo móvel. O objetivo consiste em obter uma correspondência para o local onde o dispositivo se encontra, através dos dados recolhidos *offline* e dos dados *online*.

Os *datasets* utilizados, são representados por um conjunto de amostras. Cada amostra é representada por um vetor de N dimensões, que neste caso concreto são 529, onde as primeiras 520 dimensões correspondem aos valores RSSI medidos nos APs e as restantes 9 às coordenadas, ao piso, ao edifício, ao espaço, à posição relativa, ao ID do utilizador, ao ID do dispositivo e ao *timestamp,* respetivamente. Neste projeto apenas serão utilizadas as coordenadas, o piso e o edifício destas últimas 9 dimensões, visto ser os dados necessários para o que é pretendido.

Através destes dois *datasets,* foram criadas 3 estruturas de dados diferentes, estruturas de dados utilizadas no desenvolvimento deste projeto:

- *DataTest,* que é um *ArrayList* com a lista dos valores RSSI dos 520 APs do *dataset validation*.
- *DataValidation,* que é um *ArrayList* com os resultados, isto é com a longitude, latitude, edifício e piso reais do *dataset validation.*
- *DataTraining* que é um *ArrayList* de *coordinates,* onde *coordinates* possui a lista dos 520 APs, a longitude, a latitude, o edifício e o piso do *dataset training*.

Todas as classes necessárias à importação dos ficheiros *.csv* para memória estão apresentadas na Figura 20. A classe *ReadCSV* lê do ficheiro *validation.csv* os valores RSS dos 520 APs e guarda na estrutura de dados *DataTest*. A classe *ReadValidationSet* lê o ficheiro *training.csv* e guarda na estrutura de dados *DataTraining*. A classe *RealValues* lê do ficheiro *validation.csv* a longitude, latitude, edifício e piso e guarda na estrutura *DataValidation.*

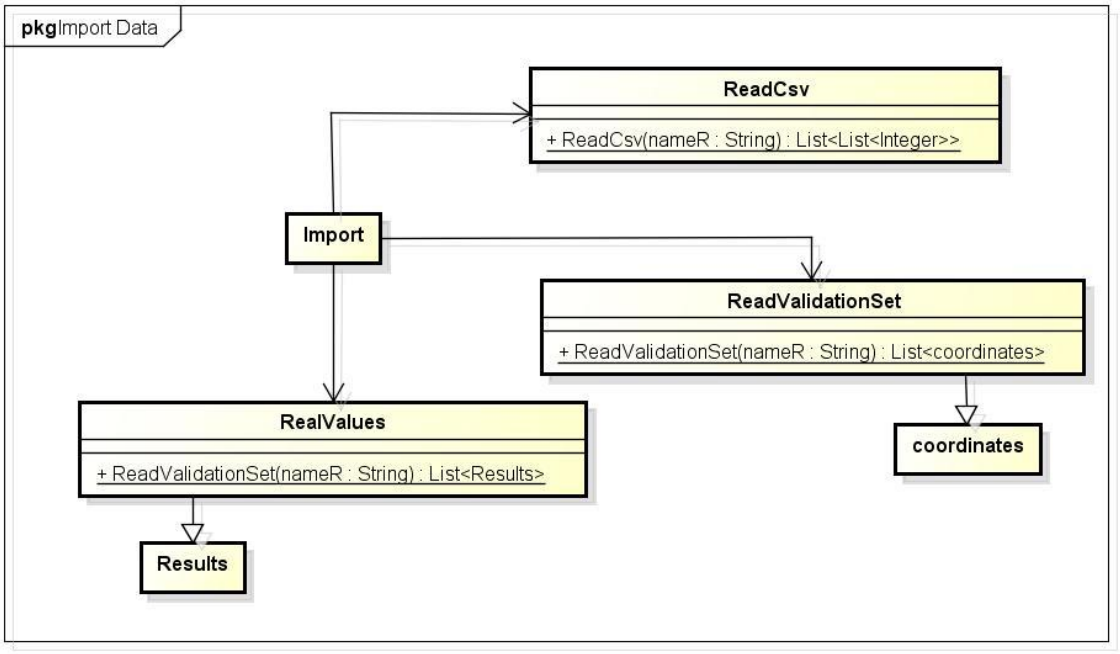

Figura 20- Importação dos dados para memória

Este *package* da importação dos dados é comum a todos os algoritmos implementados.

4.2. Algoritmos implementados

Foram implementados, no total, 4 algoritmos. Dos 4 algoritmos implementados um deles, designado por algoritmo hierárquico, é um trabalho relacionado que foi estudado e usado como implementação de referência. Os restantes três foram propostos no âmbito deste projeto, todos eles baseados em métodos probabilísticos.

4.2.1. Implementação do Algoritmo Determinístico

O algoritmo determinístico hierárquico foi desenvolvido em linguagem Java e a Figura 21 representa o diagrama de classes, com todas as classes desenvolvidas e utilizadas. Tal como foi referido anteriormente, este algoritmo utiliza o *package* da importação de dados (apresentado na secção anteriormente). De forma a simplificar o código, todas as classes utilizadas foram agrupadas em dois *packages*.

- *ImportData:* contém as classes necessárias à importação dos dados de teste, de treino e de validação;

- *HierarchicalAlgorithm:* contém as classes necessárias para o desenvolvimento do algoritmo hierárquico. A classe *IndoorLocalizationWiFi* é responsável por estimar a posição de cada *fingerprint*. A classe *AnalysisResult* é responsável por comparar os resultados estimados com os resultados reais.

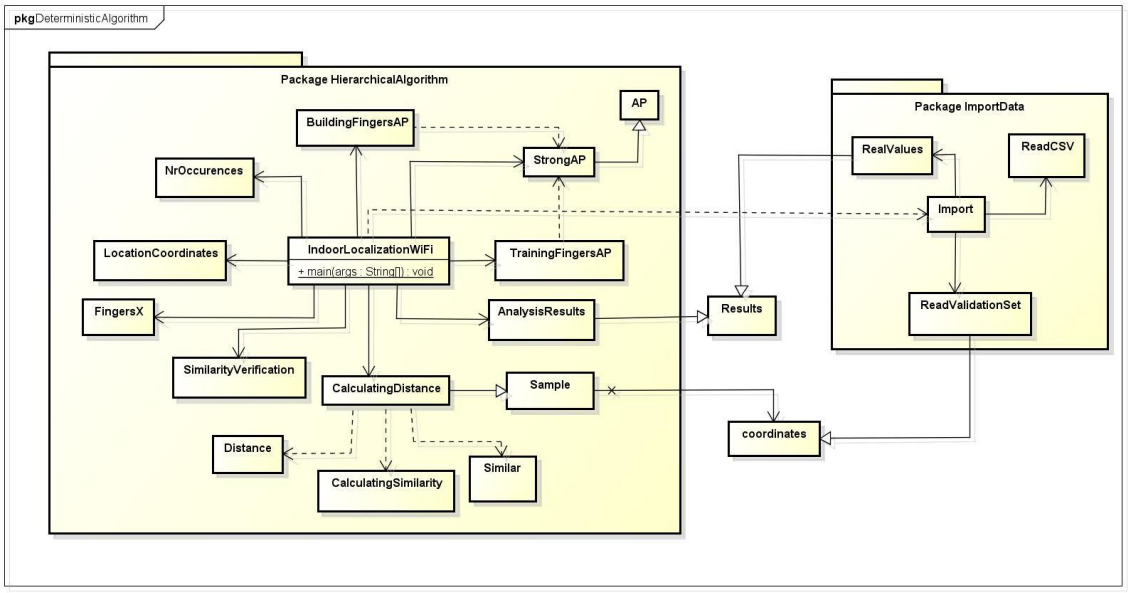

Figura 21 - Diagrama de classes algoritmo determinístico

Cada função implementada no algoritmo hierárquico tem um objetivo. Em seguida são apresentadas algumas das funções mais importantes do algoritmo. Em algumas funções são ainda apresentadas os seus diagramas de atividade em UML, de forma a compreender melhor o seu funcionamento.

Função "*strongAP*"

Esta é a função que recebe uma lista de valores RSS e determina qual o AP mais forte dessa lista (Figura 22).

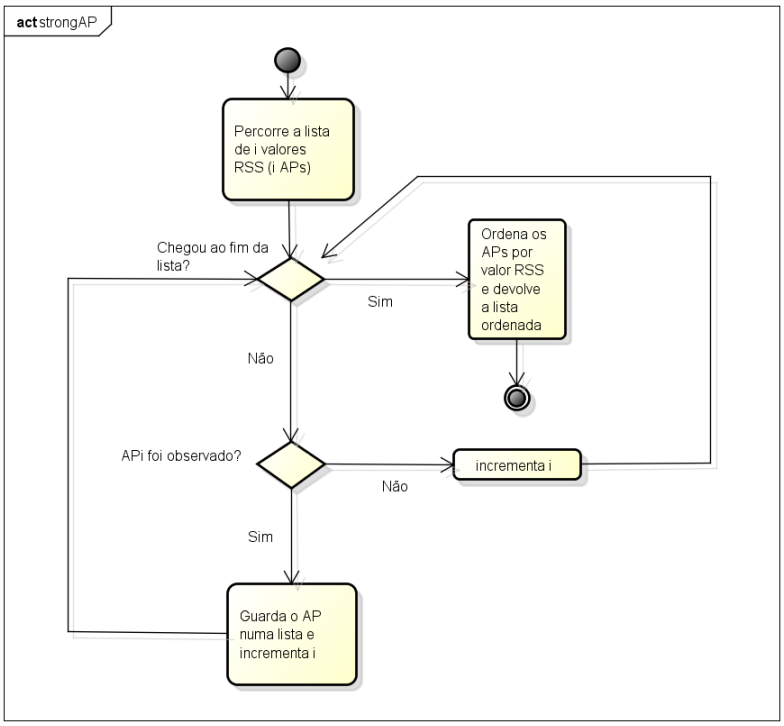

Figura 22 - Diagrama de atividade da função "strongAp"

Função "*trainingFingerAP*"

Esta função recebe o conjunto de *fingerprints* de treino e o AP mais forte de teste. É responsável por devolver um conjunto de *fingerprints*, cujo AP mais forte é igual ao AP mais forte de teste recebido (Figura 23).

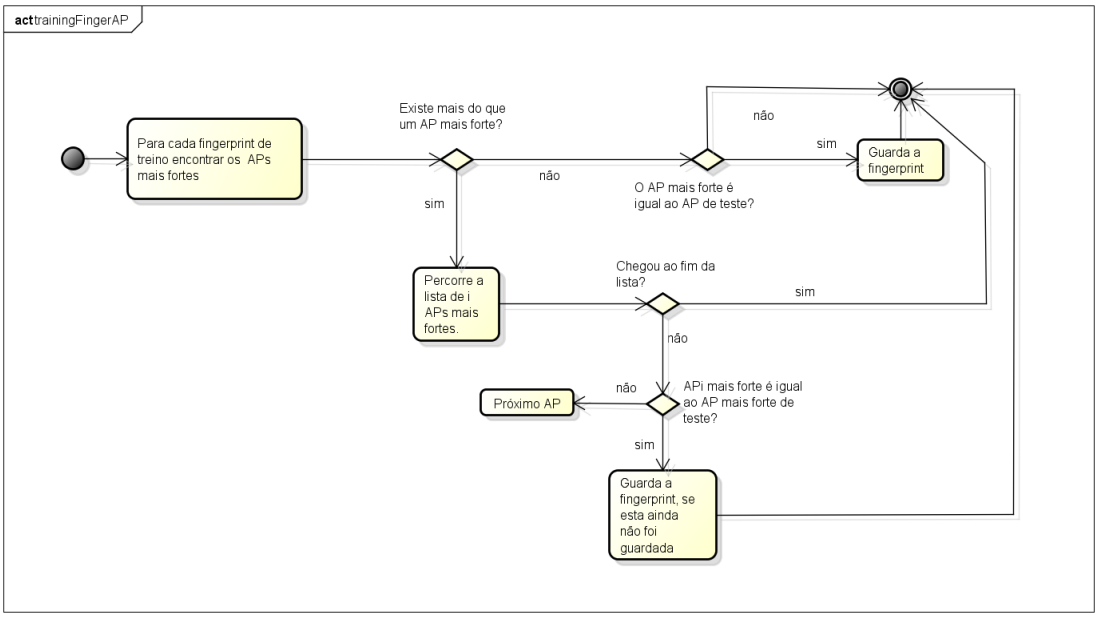

Figura 23 - Diagrama de atividade da função "trainingFingerAP"

Função "*fingerBuilding*" e "*fingerFloor*"

Cada uma destas funções recebe um conjunto de *fingerprints* e a estrutura estimada (edifício ou piso, consoante a função a utilizar). É responsável por devolver um conjunto de *fingerprints* que correspondem ao edifício/piso estimado recebido.

Função "*buildingFingersAP*"

Esta função recebe um conjunto de *fingerprints* e o primeiro, segundo e terceiro AP mais fortes da *fingerprint* de teste. É responsável por devolver as *fingerprints* de treino cujo AP mais forte é igual ao primeiro, segundo ou terceiro AP mais forte de teste.

Função "*calculatingDistance*"

Esta função recebe um conjunto de *fingerprints* e uma lista de valores RSS (correspondente à *fingerprint* de teste). É responsável por determinar a função de similaridade entre cada uma das *fingerprints* do conjunto e a *fingerprint* de teste. E ainda, responsável por ordenar as *fingerprints* pelo valor da similaridade calculada e devolvê-las.

Função "*kfinger*"

Esta função recebe um conjunto de *fingerprints* ordenadas por similaridade, e um valor *k*. É responsável por devolver as *k* primeiras *fingerprints* do conjunto recebido.

Função "*analysisRes*"

Esta função recebe os dados de validação e os dados estimados em cada *fingerprint*. É responsável por verificar a percentagem de acerto dos edifícios e dos pisos e ainda por calcular o erro médio de cada *fingerprint*.

Função "*simpleCentroid*"

Esta função recebe as coordenadas das *fingerprints*. É responsável por calcular a média dessas coordenadas recebidas, e devolver a longitude e a latitude (Figura 24).

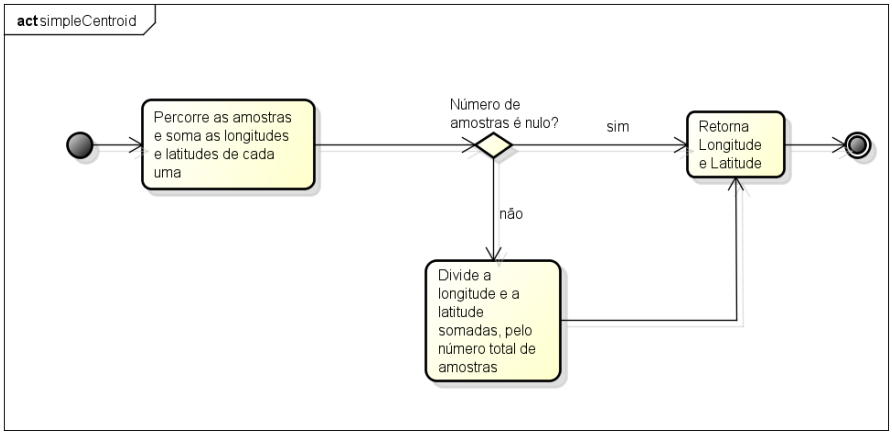

Figura 24 - Diagrama de atividade da função "simpleCentroid"

Função "*weightedCentroid*"

Esta função para além das cordenadas das *fingerprints*, recebe também as distâncias euclidianas de cada uma dessas *fingerprints*. É responsável por calcular a longitude e latitude através da atribuição de 'pesos'. No final devolve a longitude e a latitude calculadas (Figura 25**Erro! A origem da referência não foi encontrada.**).

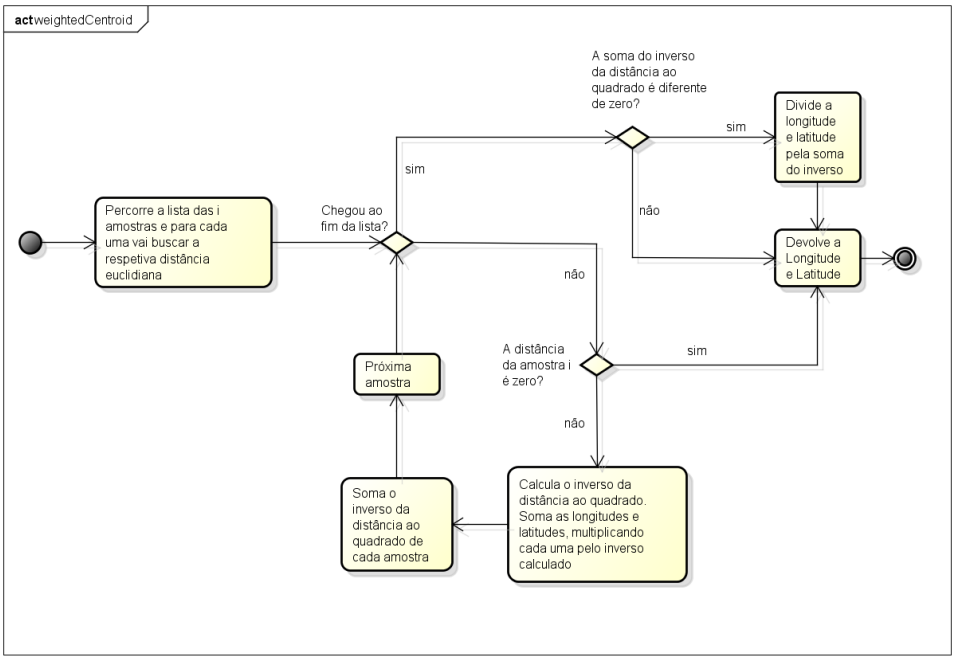

Figura 25 - Diagrama de atividade da função "weightedCentroid"

4.2.2. Implementação dos Algoritmos Probabilísticos

Os algoritmos probabilísticos foram também desenvolvidos em linguagem Java e a Figura 26 representa o diagrama de classes, com todas as classes desenvolvidas e utilizadas. De forma a simplificar o código, todas as classes utilizadas foram agrupadas nos seguintes *packages*.

- *ImportData:* contém as classes necessárias à importação dos dados de teste, de treino e de validação;
- *kPM_ProbabilityHistogram:* contém as classes necessárias para o desenvolvimento do algoritmo probabilístico que utiliza as probabilidades estimadas por histograma.
- *kPM_ProbabilityDetectedAPs:* contém as classes necessárias para o desenvolvimento do algoritmo probabilístico que utiliza as probabilidades de deteção dos APs.
- *kPM_ProbabilityKernel:* contém as classes necessárias para o desenvolvimento do algoritmo probabilístico que utiliza as probabilidades estimadas pela função gaussiana de *Kernel*.

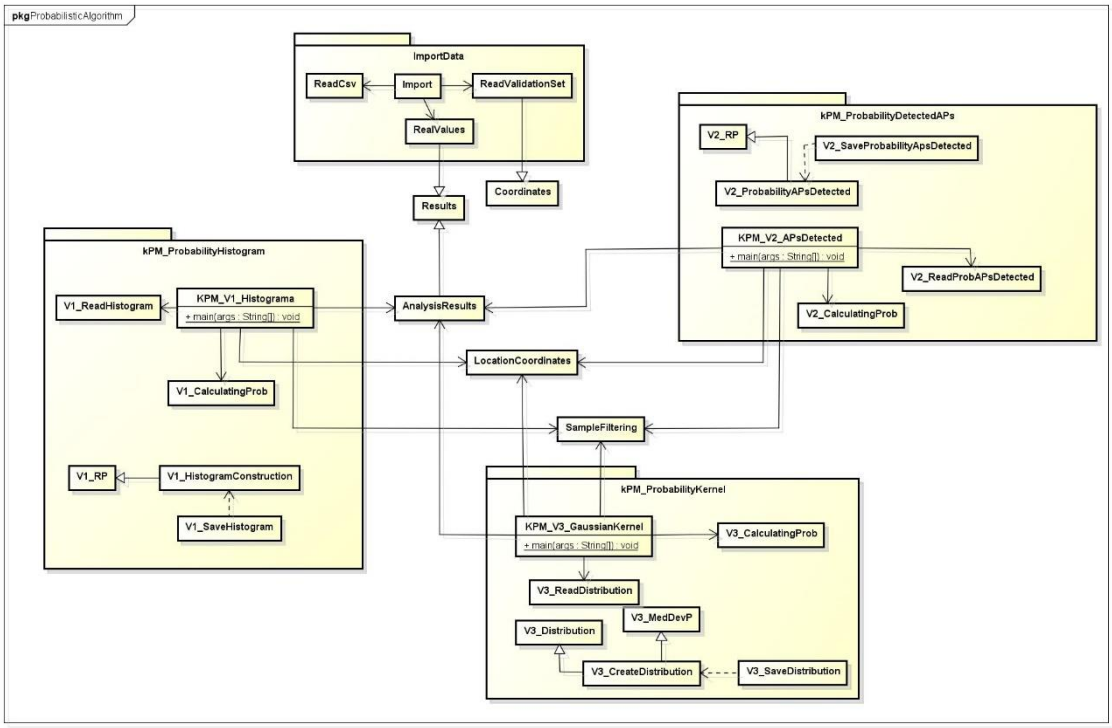

Figura 26 - Diagrama de classes do algoritmo probabilístico

Também nos algoritmos probabilísticos cada função implementada tem um objetivo. Grande parte das funções utilizadas na implementação do algoritmo determinístico foram reaproveitadas e utilizadas nos algoritmos probabilísticos. Algumas sofreram pequenas alterações na estrutura de dados. No algoritmo determinístico foi utilizada sempre a estrutura de uma lista de *coordinates* (classe que contem a lista de valores RSSI, a longitude, a latitude, o edifício e o piso), enquanto que nos algoritmos probabilísticos foi utilizada uma estrutura composta por uma lista de *position* (classe que contém a longitude, latitude, edifício e piso). Para além das funções que já foram apresentadas na secção anterior, é importante referir a função "S*imilarityCalculationProb*". Esta função é responsável pelo cálculo da probabilidade de cada posição. Embora possua o mesmo nome para os diferentes algoritmos probabilísticos, estas calculam a probabilidade de forma diferente, como já foi referido anteriormente.

Em seguida são apresentados os diagramas de atividades UML da distribuição composta pelas probabilidades estimadas por histograma (Figura 27) e do diagrama da distribuição composta pelas probabilidades de deteção dos APs (Figura 28).

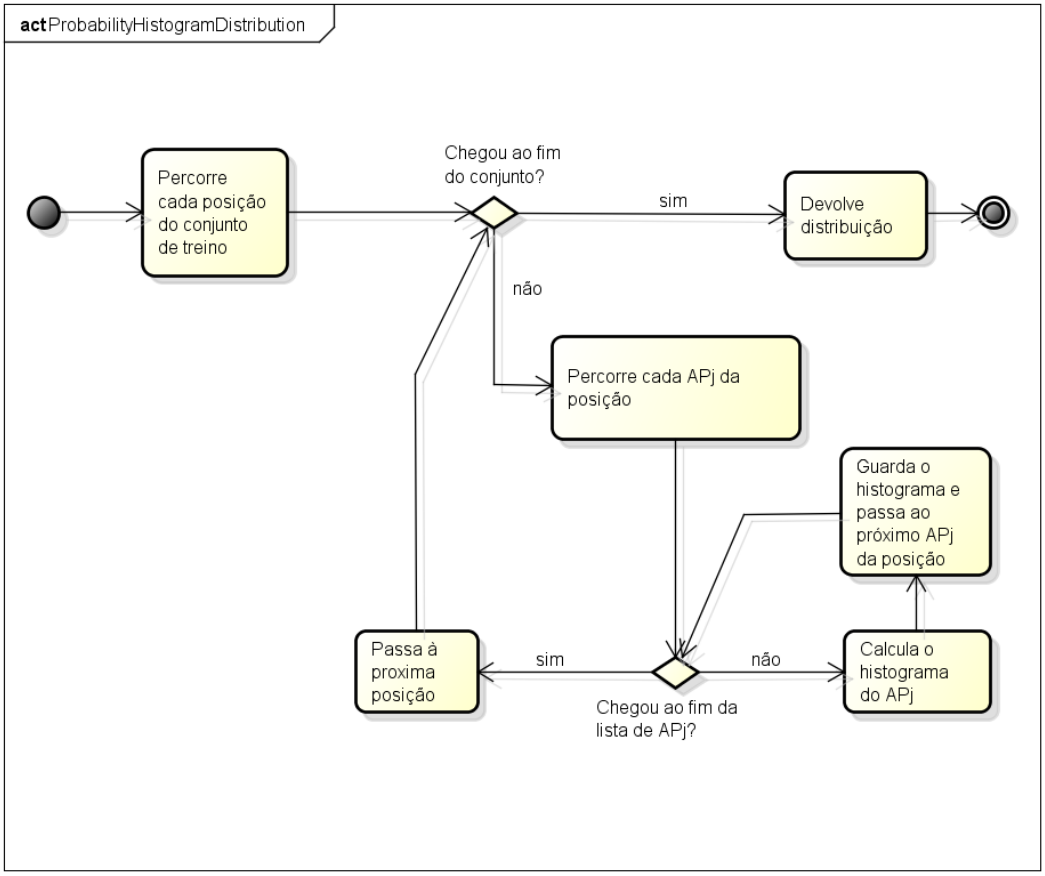

Figura 27- Distribuição composta pelas probabilidades estimadas por histograma

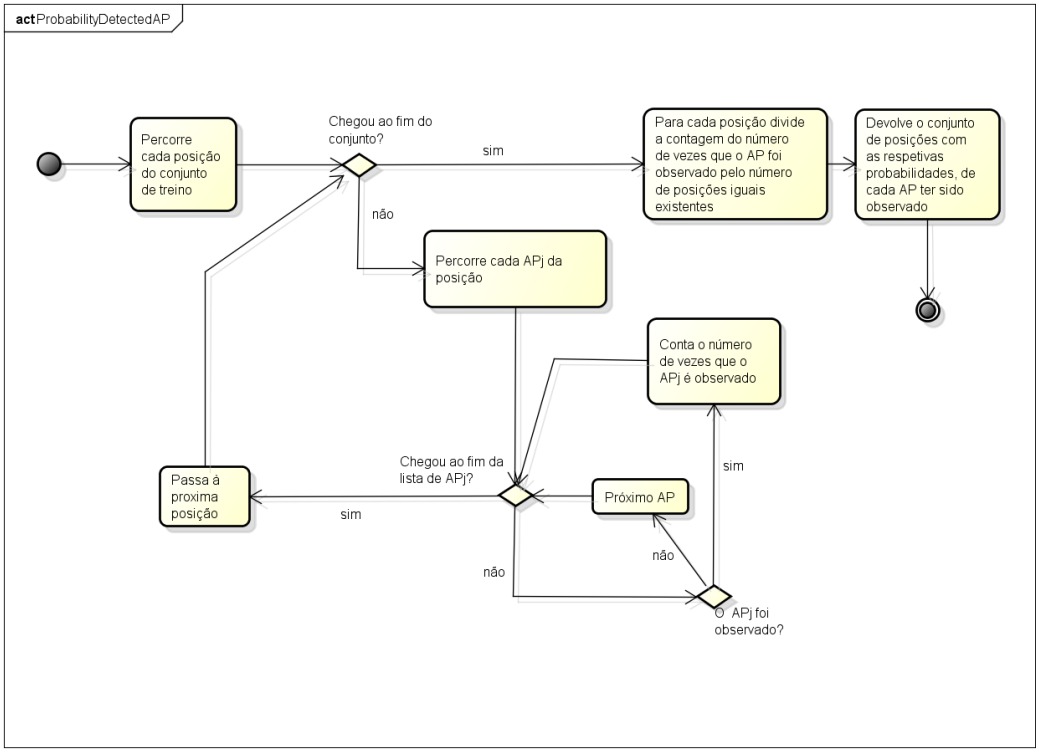

Figura 28 - Distribuição composta pelas probabilidades de deteção de APs

A distribuição composta pela média dos valores RSS de cada AP é semelhante à distribuição composta pelas probabilidades de deteção de APs. A diferença é que em vez de ser contado o número de vezes que o AP foi observado, são somados os valores RSS de cada AP numa posição P e depois dividido pelo número de amostras existentes para aquela posição P.

4.2.3. Tempos de execução

Os testes foram realizados num sistema *Windows 10* de 64 *bits* com um processador Intel® Core™ i5-3210M CPU@ 2.40GHz e com 6GB de RAM. O ambiente de desenvolvimento utilizado foi o *NetBeans* IDE 8.0.1, que permitiu escrever, compilar e realizar o *debug* dos algoritmos de localização.

Relativamente aos tempos de execução de cada um dos algoritmos, o que apresenta melhor tempo é o algoritmo probabilístico baseado na deteção dos APs (Figura 29), com um tempo de execução de exatamente 2 segundos. Seguido a este está o algoritmo probabilístico baseado na função de *Kernel* (Figura 30) com 43 segundos. Depois está o algoritmo determinístico (Figura 31) com um tempo de 3 minutos e 36 segundos. Por fim, está o algoritmo probabilístico baseado em histograma (Figura 32), com um tempo de 5 minutos e 44 segundos. Estes tempos de execução envolvem o tempo de carregar os dados dos *datasets* para memória, da estimativa total da localização e ainda da validação dos algoritmos (cálculo do respetivo erro médio).

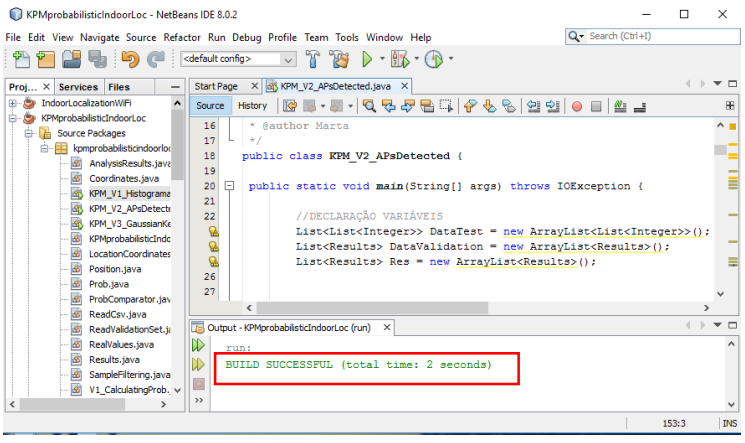

Figura 29 - Tempo de execução algoritmo kPM baseado nas probabilidades da deteção dos APs

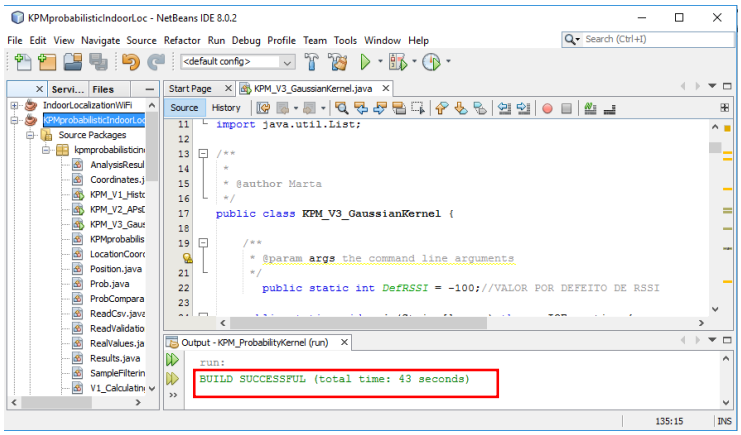

Figura 30 -Tempo de execução algoritmo kPM baseado nas probabilidades da função gaussiana de Kernel

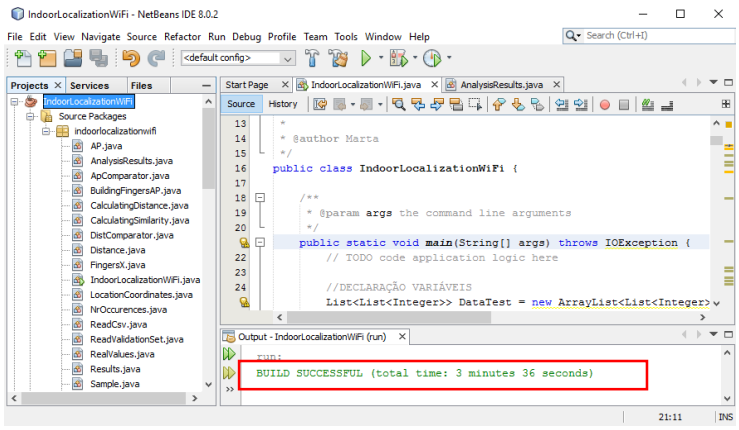

Figura 31 - Tempo de execução algoritmo determinístico

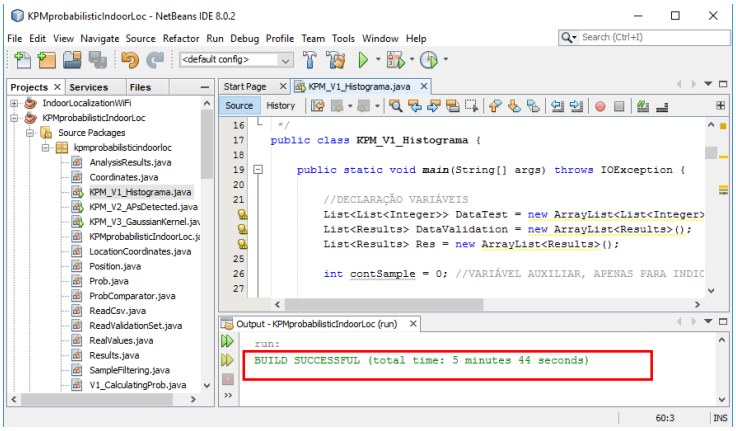

Figura 32 - Tempo de execução algoritmo kPM baseado nas probabilidades estimadas por histograma

Capítulo 5

Testes e Resultados

Após desenvolvidos e implementados os algoritmos de localização, procedeu-se à execução de uma série de testes. Foram efetuados diversos testes com parâmetros distintos, de forma a compreender melhor o comportamento dos algoritmos implementados. Neste capítulo são apresentados todos os testes realizados e os respetivos resultados obtidos.

5.1. Dados utilizados

Assim como foi referido no capítulo 1, neste trabalho foram utilizados dois *datasets* fornecidos pela competição EvALL [5], nomeadamente o *dataset training* e o *dataset validation.*

Estes dados de treino e de validação, foram recolhidos em três edifícios da Universidade de Jaume I, edifícios com 4 ou 5 andares cada. Algumas das características mais relevantes de cada um destes dois *datasets* é apresentada na Tabela 12.

Tabela 12 - Características dos dois datasets

	Training	*Validation*
Amostras	19937	1111
Edifícios distintos	3	3
Pisos distintos	13	13
Posições distintas	933	1053

O número de APs observados difere entre os dois *datasets* [6], o *dataset training* possui 465 APs observados enquanto que o *dataset validation* apenas possui 367.

No d*ataset training* existem 76 amostras onde não foi observado um único AP, estas 79 amostras não são úteis, logo poderiam ser removidas.

Neste projeto foram então utilizados os dois *datasets*, como já foi referido anteriormente. O *dataset training*, foi utilizado como mapa de rádio, contendo assim os dados de treino (dados recolhidos *offline*) e o *dataset validation*, foi usado como dados de teste (dados recolhidos *online*).

5.2. Testes aos algoritmos de localização

Foram realizados testes para cada um dos algoritmos implementados de forma a obter o menor erro médio possível. Este erro médio é dado pela média do erro de cada amostra, e o erro é dado pela fórmula utilizada na competição [6] (equação (25)).

$$Erro = Distancia(R_i, E_i) + pn1 \times bFail + pn2 \times fFail \qquad (25)$$

A $Distancia(R_i, E_i)$ é a distância Euclidiana entre a posição real e a posição estimada, *pn1* e *pn2* são as penalizações associadas ao edifício errado e ao piso errado respetivamente, assumindo os valores 50 e 4 metros. A variável *bFail* assume o valor 1 para estimativas incorretas do edifício e 0 para estimativas corretas. A variável *fFail* assume o valor da diferença absoluta entre o piso estimado e o piso correto.

5.2.1. Testes do algoritmo hierárquico

Nos testes realizados para este algoritmo, no que diz respeito à estimativa do edifício é obtido 100% de acerto.

No que diz respeito à estimativa do piso e da localização, foi necessário o ajuste de alguns parâmetros, tais como o valor por defeito de RSSI e o valor das *k1* amostras a

selecionar (amostras mais semelhantes à *fingerprint* de teste). O valor por defeito foi variando entre valores considerados fracos, -90dBm, -95dBm, -102dBm, -105dBm e -110dBm. Para cada valor d*efRSSI* o valor de *k1* foi variando, assumindo os valores entre os 30 e 70. Nos testes do erro médio é utilizado o valor de *defRSSI* em conjunto com o de *k1* que apresentar melhores resultados no acerto do piso. A Tabela 13 apresenta os resultados obtidos para a estimativa do piso, para os diferentes valores de *defRSSI*.

Tabela 13 - Resultados da estimativa do piso do algoritmo hierárquico

defRSSI	*-90*				
k1	30	35	**40**	50	
% Piso	92.979%	92.979%	**92.979%**	92.889%	
defRSSI	*-95*				
k1	30	35	40	**50**	60
% Piso	92.889%	92.979%	93.249%	**93.249%**	93.269%
defRSSI	*-102*				
k1	30	35	40	**50**	60
% Piso	92.709	92.799%	93.159%	**93.429%**	93.249%
defRSSI	*-105*				
k1	30	35	40	**50**	60
% Piso	92.709%	92.709%	93.159%	**93.429%**	93.249%
defRSSI	*-110*				
k1	30	35	40	**50**	60
% Piso	92.709%	92.709%	93.069%	**93.609%**	93.159%

Na Tabela 13 estão assinalados os melhores resultados para cada um dos valores de *defRSSI*. Identificado o melhor valor de *k1*, foram então realizados testes para a estimativa da localização. Para a estimativa da localização o valor de *k2* foi também ajustado, tomando valores entre 3 a 11. Para cada *k2*, existe ainda duas hipóteses, o cálculo do *centroid* das coordenadas através do algoritmo kNN e o WkNN. Assim é identificado o melhor valor de *k2* e ainda o algoritmo que permite obter melhores resultados, o kNN ou WkNN. A Tabela 14 apresenta os resultados obtidos para a estimativa da localização.

Tabela 14 - Resultados da estimativa da localização do algoritmo hierárquico

defRSSI = -90 *k1=40*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	*WkNN*	*kNN*	***WkNN***
Erro	8.670	8.360	8.432	8.091	8.365	7.944	**8.344**	7.883	8.351	**7.877**

defRSSI = -95 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.488	8.235	8.221	7.963	8.162	7.834	8.094	7.743	**8.086**	**7.687**
defRSSI = -102 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.341	8.176	8.202	8.025	8.118	7.910	8.043	7.818	**8.022**	**7.765**
defRSSI = -105 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	***WkNN***
Erro	8.343	8.197	8.196	8.047	8.130	7.943	8.031	7.869	**8.031**	**7.813**
defRSSI = -110 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.355	8.262	8.243	8.136	8.162	8.057	8.061	7.953	**8.014**	**7.863**

Os melhores resultados obtidos estão assinalados na Tabela 14. Mediante os resultados obtidos, os valores para WkNN deveriam ser melhores do que o kNN, e na tabela pode-se verificá-lo. No entanto os valores são muito próximos, a margem entre eles é pequena. Após a realização de análises, foi concluído que o facto deste acontecimento era devido às *k2* posições serem iguais. Isto é, o *dataset* de treino possui várias amostras para as mesmas coordenadas, desta forma as *k2* amostras mais similares serão iguais. Ao possuir as mesmas longitudes e as mesmas latitudes, a diferença entre utilizar o kNN ou o WkNN será mínima. Foram efetuados testes para um conjunto de amostras com diferentes longitudes e latitudes, de forma a confirmar que realmente o algoritmo WkNN obtém melhores resultados do que o kNN. Os resultados que foram obtidos confirmaram que o algoritmo WkNN possui melhores resultados do que o kNN.

Os mesmos testes foram realizados para as propostas apresentadas a este algoritmo, nomeadamente a função de similaridade euclidiana com parâmetros adicionais, a função de similaridade euclidiana e a função de similaridade de *Manhattan*. Primeiro foram realizados testes para o *k1* identificando o melhor valor, seguido dos testes para o *k2*. Os melhores resultados parciais dessas 3 funções de similaridades são apresentados na

Tabela 15, Tabela 16 e Tabela 17, respetivamente. Os resultados detalhados para estas 3 similaridades podem ser encontrados no Apêndice B.

Tabela 15 - Resultados parciais obtidos para a Similaridade Euclidiana com parâmetros adicionais

	Edifício	*Piso*	*Erro Médio(m)*	
			kNN	*WkNN*
			k2=11	*k2=11*
defRSSI=-90 e k1=50	100%	90.639%	9.571 m	8.822 m

Tabela 16 - Resultados parciais obtidos para a Similaridade Euclidiana

	Edifício	*Piso*	*Erro Médio(m)*	
			kNN	*WkNN*
			k2=9	*k2=11*
defRSSI=-90 e k1=50	100%	94.599%	7.746 m	7.440 m
			k2=11	*k2*=11
defRSSI=-95 e k1=50	100%	93.969%	7.473 m	7.428 m

Tabela 17 - Resultados parciais obtidos para a Similaridade de Manhattan

	Edifício	*Piso*	*Erro Médio(m)*	
			kNN	*WkNN*
			k2=9	*k2=11*
defRSSI=-90 e k1=50	100%	93.879%	7.689 m	7.657 m

Na Figura 33 estão presentes todos os resultados obtidos para a estimativa do piso, e como se pode observar a estratégia que apresenta maior percentagem, a nível geral, é a Similaridade Euclidiana, calculada apenas através da distância Euclidiana.

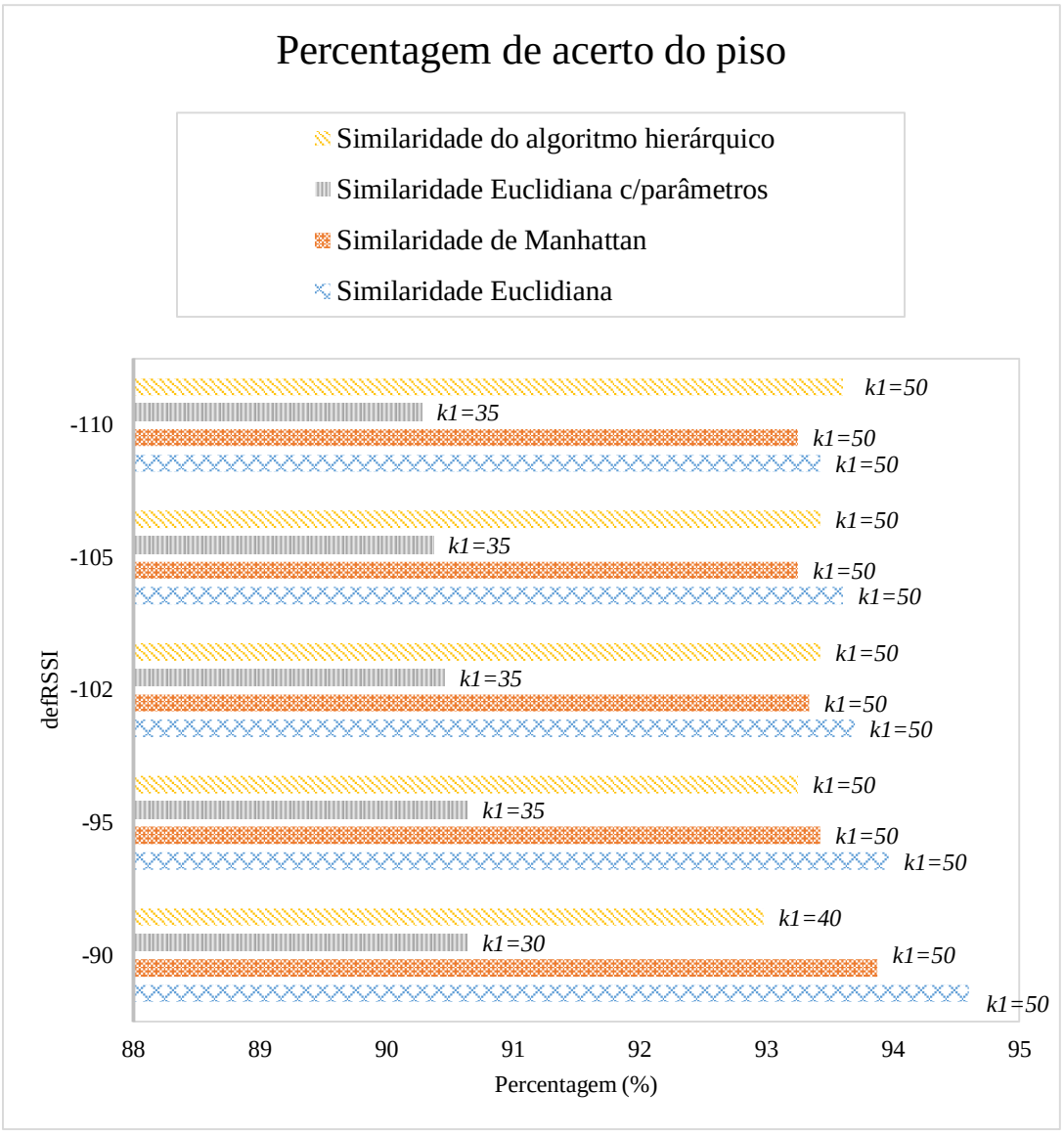

Figura 33- Percentagens de acerto no piso do algoritmo hierárquico e suas funções de similaridade

Segundo a Figura 34 e Figura 35 pode-se dizer que a melhor estratégia é a Similaridade Euclidiana, pois é a que apresenta o menor erro na estimativa total (edifício, piso e localização), tanto para kNN (Figura 34) como para WkNN (Figura 35), assim como era esperado.

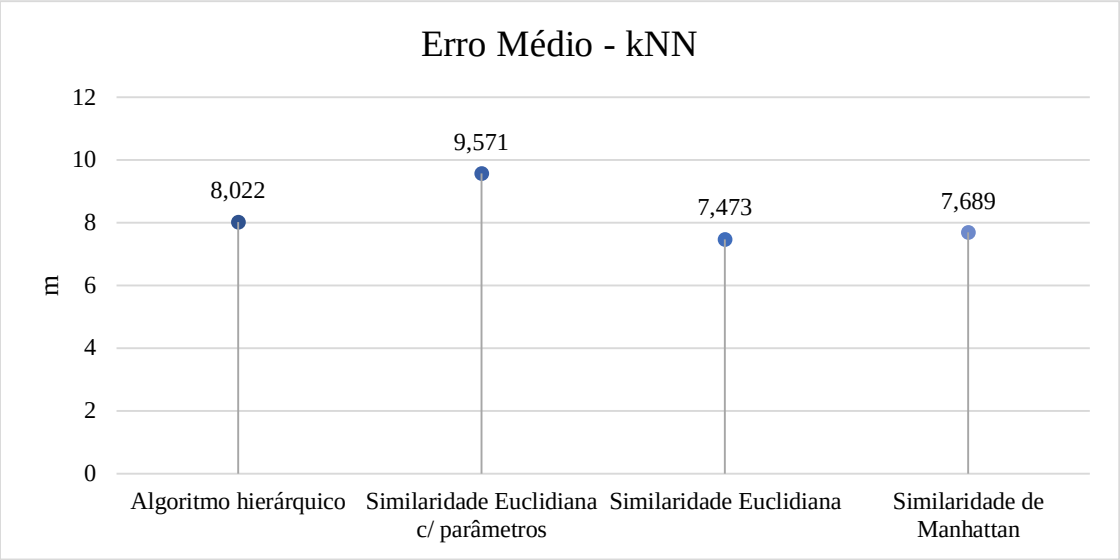

Figura 34 - Erro médio kNN da estimativa da posição do algoritmo hierárquico

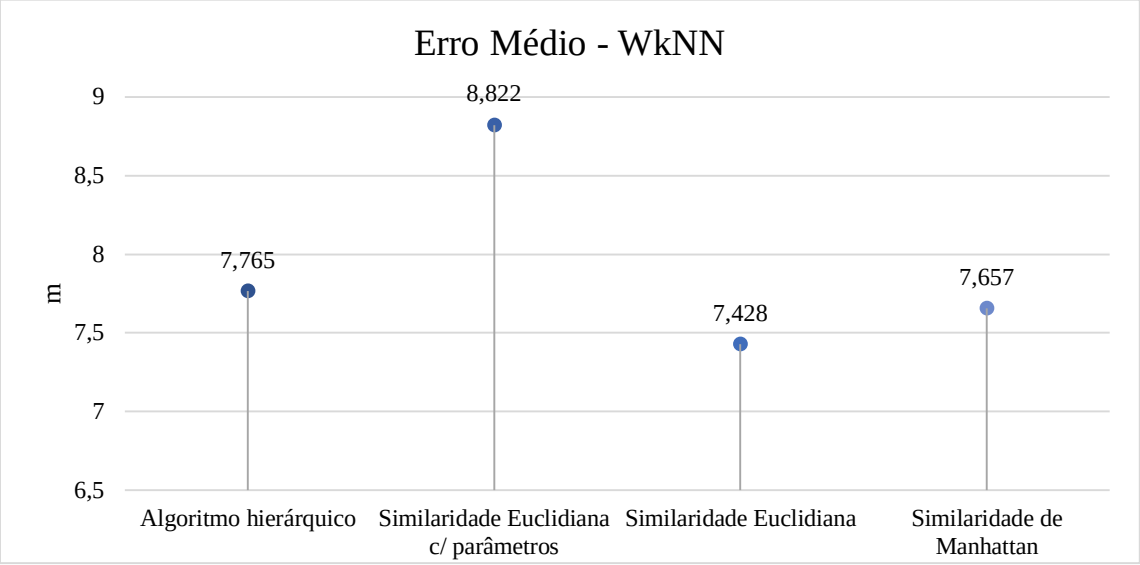

Figura 35 - Erro médio WkNN da estimativa da posição do algoritmo hierárquico

Apresentados os resultados deste algoritmo, na espectativa de tentar melhorar os resultados, foram propostos algoritmos baseados em métodos probabilísticos, que tal como o nome indica, faz uso das probabilidades para a estimação da posição.

5.2.2. Testes do algoritmo *kPM*

O algoritmo *kPM* possui três variantes, todas elas utilizam probabilidades, mas através de metodologias diferentes. Em seguida são apresentados os resultados obtidos para cada uma delas.

5.2.2.1. Testes do algoritmo *kPM*: Distribuição baseada em histograma

Para a realização dos testes desta proposta, no que diz respeito à estimativa do edifício, também foi necessário o ajuste de alguns parâmetros. Foram ajustados os valores por defeito de RSSI (*defRSSI*), o valor das *k1* amostras a selecionar e ainda o valor de *r* que define o intervalo de valores a utilizar. O valor por defeito foi variando entre valores considerados fracos, -85dBm, -90dBm, -95dBm, -100dBm, -105dBm e -110dBm. Para cada valor d*efRSSI* o valor de *r* foi variando entre 5 a 40, ou seja, entre um intervalo de 10 valores e um intervalo de 80 valores (acaba por incluir quase todos). O valor de *k1* também foi variando entre valores pequenos e maiores de forma a compreender o comportamento do algoritmo.

Na Tabela 18 são apresentados os resultados para *k1*=3. As tabelas para os restantes valores de *k1* podem ser consultadas no apêndice C. A maior percentagem de acerto obtida para o piso em todos os testes realizados é apresentada nesta tabela. O melhor resultado obtido foi, para um valor por defeito de -105 e um intervalo de 80 valores, sendo essa mesma percentagem de 95.591%.

Tabela 18 - Estimativa do edifício algoritmo kPM (Distribuição baseada em histograma) para k1=3

	-5 … +5	**-10 … +10**	**-15 … +15**	**-20 … +20**	**-25 … +25**	**-30 … +30**	**-35 … +35**	**-40 … +40**
-85	60.036%	79.928%	84.878%	80.558%	70.117%	64.356%	55.806%	33.213%
-90	60.216%	76.508%	86.139%	87.039%	88.929%	72.097%	60.576%	46.625%
-95	58.416%	70.837%	82.088%	89.019%	87.849%	87.579%	72.367%	56.706%
-100	52.926%	64.356%	74.977%	83.798%	90.189%	89.469%	95.231%	72.997%
-105	50.945%	57.786%	68.137%	77.408%	85.779%	90.549%	90.369%	**95.591%**
-110	50.945%	54.905%	60.666%	69.937%	79.028%	86.319%	90.729%	90.549%

Não sendo este um valor aceitável para a estimativa do edifício foram realizadas análises complementares de forma a entender o porquê de falhar no acerto total dos edifícios. Estas análises podem ser encontradas no Apêndice D.

Após efetuadas análises, foi verificado que existiam APs, que mesmo utilizando intervalos obtinham probabilidade 0.0, que ao multiplicar tornava a probabilidade final da posição 0.0. Com o objetivo de contornar o dilema das probabilidades a zero, foi atribuída uma probabilidade uniforme diferente de zero (a probabilidade uniforme é utilizada nos APs cujos valores RSSI têm probabilidade nula). Assim, a probabilidade será sempre diferente de zero, no entanto esse valor deve ser baixo pois tratam-se de valores RSSI não observados (foi utilizado o valor de 0.01, para contornar os 0's). Mas mesmo com esta questão dos 0's resolvida, o valor máximo obtido foi 97,93%. Não sendo ainda um resultado satisfatório, foi necessário pensar noutra metodologia, e assim surgiu a variante probabilística baseada na visibilidade dos APs.

5.2.2.2. Testes do algoritmo *kPM*: Distribuição de Visibilidade dos APs

Nos testes realizados para esta variante, no que diz respeito à estimativa do edifício, é obtido 100% de acerto.

Para a realização dos testes para a estimativa do piso e da localização, foi necessário o ajuste de alguns parâmetros, nomeadamente as k1 e k2 amostras a selecionar.

A Tabela 19, apresenta os resultados obtidos para a estimativa do piso para os diferentes valores de k1, este valor foi variando entre 1 e 11.

Tabela 19 - Percentagem de acerto no piso para a algoritmo kPM baseado na visibilidade dos APs

	k1=1	*k1=3*	*k1=5*	*k1=7*	*k1=9*	*k1=11*
% Piso	78.488%	80.018%	82.718%	**84.608%**	83.978%	83.618%

Identificado o melhor valor de k1 para a estimativa do piso (k1=7), foram realizados testes para a estimativa das coordenadas. Para esta estimativa, o valor de k2 foi também variando Tabela 20, de forma a identificar qual o valor de k2 que apresenta melhores resultados dentro dos possíveis.

Tabela 20 - Erro médio final com a estimativa do edifício, piso e localização do algoritmo kPM baseado na visibilidade dos APs

	k2=2	*k2=3*	*k2=4*	*k2=5*	*k2=6*
Erro (m)	11.934	11.569	**11.447**	11.451	11.607

Assim sendo para esta variante, os melhores resultados obtidos são de 100% de acerto para o edifício e 84,608% para o piso, com um erro final de 11,447 metros. Este erro é afetado logo de início pela percentagem de acerto do piso, este diminuiria se essa percentagem aumentasse. Análises complementares a este algoritmo podem ser encontradas no Apêndice E.

5.2.2.3. Testes ao algoritmo *kPM*: Distribuição de *Kernel*

Para a realização dos testes para esta terceira variante dos algoritmos probabilísticos, foram ajustados os valores de alguns parâmetros, tal como o *defRSSI* que assumiu valores de -95, -100, -105 e -110, o valor de *k1* e de *k2*.

No que diz respeito à estimativa do edifício para este algoritmo, foram obtidos os resultados presentes em Figura 36 .

	defRSSI=-95	*defRSSI=-100*	*defRSSI=-105*	*defRSSI=-110*
% Edifício	98.92%	**100%**	99.37%	97.75%

Figura 36 - Percentagem de acerto no edifício, algoritmo kPM baseado na distribuição de Kernel

Identificado o valor *defRSSI* com melhores resultados para o edifício, foram realizados testes para estimativa do piso para diferentes valores de *k1*, Figura 37.

	k1=1	*k1=3*	*k1=5*	*k1=7*	*k1=9*
% Piso	90.999%	**91.179%**	90.279%	89.919%	89.109%

Figura 37- Percentagem de acerto no piso, algoritmo kPM baseado na distribuição de Kernel

Por fim, foram realizados testes para diferentes valores de *k2*, de forma a verificar qual o menor erro médio obtido, Figura 38.

	k2=1	*k2=2*	*k2=3*	*k2=4*	*k2=5*
Erro (m)	9.267	**8.736**	8.936	8.984	9.063

Figura 38- Erro médio, algoritmo kPM baseado na distribuição de Kernel

Para esta terceira variante, o melhor erro obtido é para um *defRSSI* de -100dBm, um *k1* de 3 e um *k2* de 2.

5.2.3. Comparação dos resultados obtidos para os diferentes algoritmos

Realizados os testes para todos os algoritmos, são agora analisados os resultados gerais obtidos para cada um deles. A Figura 39 apresenta os melhores resultados obtidos na estimativa do edifício e do piso, para cada um dos algoritmos apresentados neste projeto.

Como se pode verificar, em termos de acerto no piso, apenas o algoritmo probabilístico que faz uso da distribuição baseada em histograma não apresenta os 100% de acerto. Esta estratégia do histograma, assim como já foi referido, não é a melhor. Nesta estratégia, o valor RSSI de teste pode não estar entre a gama de valores RSSI presentes no histograma, no entanto este pode conter valores muito próximos. A solução dos intervalos resolve alguns casos, mas não todos, não sendo então esta estratégia a melhor a nível de métodos probabilísticos.

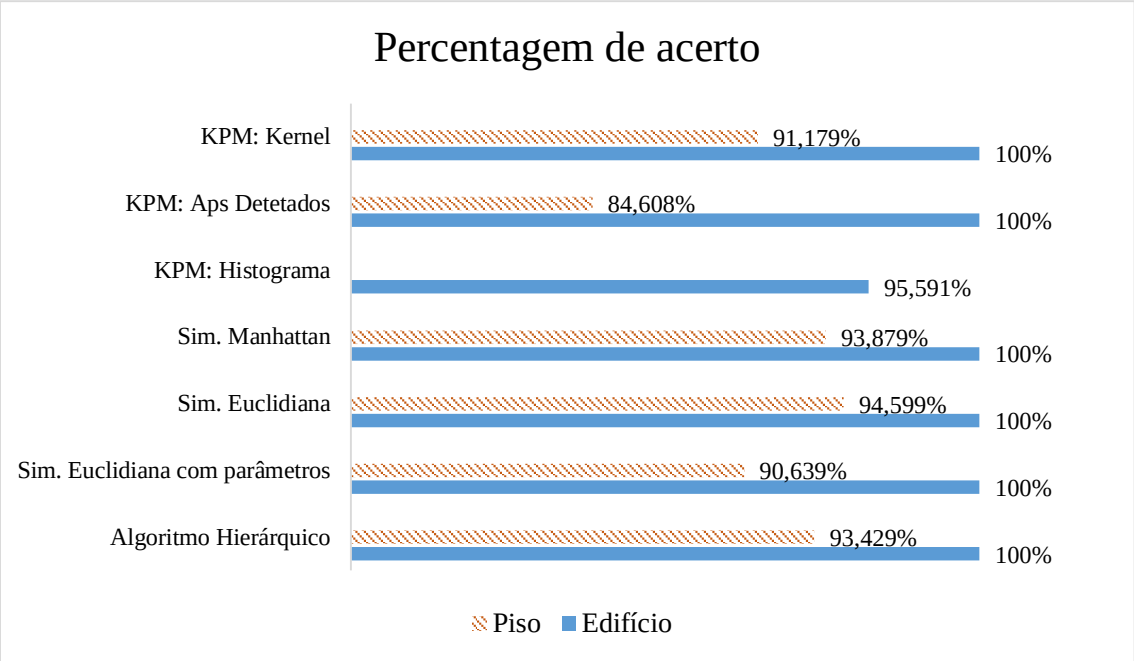

Figura 39 - Percentagem de acerto geral do edifício e do piso

No que diz respeito à estimativa do piso, é perfeitamente visível que os métodos determinísticos são os que detêm as melhores percentagens de acerto. No entanto os métodos probabilísticos não ficam muito longe, especialmente o algoritmo probabilístico baseado na distribuição de *Kernel*. Este apresenta uma percentagem de acerto perto das dos métodos determinísticos, o que é um bom resultado.

Por fim, na Figura 40 é apresentado o erro médio obtido para cada um dos algoritmos apresentados. Como é possível verificar, os métodos determinísticos (Algoritmo hierárquico, Similaridade Euclidiana com parâmetros adicionais, Similaridade Euclidiana e Similaridade de *Manhattan*) apresentam menor erro médio na estimativa da posição em relação aos métodos probabilísticos, no entanto entre os métodos probabilísticos, existe um que se destaca e que se aproxima dos métodos determinísticos, nomeadamente o algoritmo de *kPM* baseado na distribuição de *Kernel*.

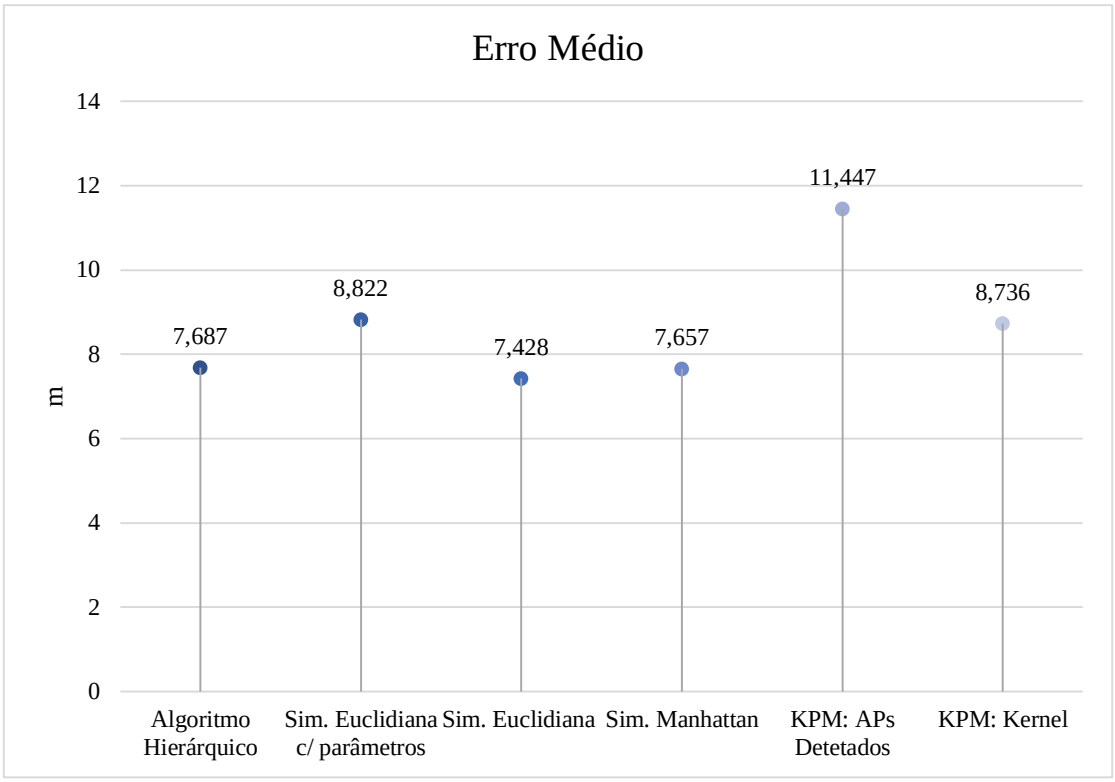

Figura 40 - Erro médio geral

Capítulo 6

Conclusão e trabalho futuro

O principal foco desta dissertação era obter melhorias nos algoritmos de localização *indoor*.

Para o desenvolvimento desta dissertação foram estudadas tecnologias e técnicas utilizadas para a estimação da localização em ambientes *indoor*. Estes estudos permitiram uma familiarização com esta área, compreendendo o estado do desenvolvimento atual destes algoritmos de localização e ainda as dificuldades que são encontradas na sua implementação.

Neste capítulo são apresentadas as conclusões deste trabalho, onde é especificado o trabalho desenvolvido e os resultados finais obtidos. Na última secção são apresentados quais os passos futuros para a evolução deste trabalho.

6.1. Conclusões

Este projeto começou com a implementação de um algoritmo determinístico, já existente, para ser usado como implementação de referência. Foi necessário despender algum tempo na sua conceção e ajuste de parâmetros de funcionamento. Após a compreensão completa deste algoritmo já existente, um dos objetivos era melhorar os resultados deste e ainda integrar novos algoritmos, nomeadamente, algoritmos probabilísticos. Foram então delineadas algumas propostas de novos algoritmos probabilísticos de forma a comparar o desempenho destes com os algoritmos determinísticos. Foi concebido e implementado um algoritmo fundamentado numa distribuição baseada em histograma, sendo criado um histograma de valores RSSI por cada AP de uma determinada posição, e foi também um algoritmo baseado numa distribuição de visibilidade dos APs, onde cada AP de uma posição possuía a respetiva probabilidade de este ter sido observado. Finalmente um outro algoritmo baseado numa distribuição da função *Gaussiana* de *Kernel*.

Deste projeto resultaram assim seis implementações de algoritmos de localização: um algoritmo determinístico baseado nas distâncias e nos *k* vizinhos mais próximos, que deu origem a 3 "variantes" no que toca à similaridade, e três probabilísticos (distribuição baseada em histograma, distribuição de visibilidade dos APs e distribuição de *kernel*). No que diz respeito às 3 'variantes' do algoritmo determinístico, estas apresentam resultados satisfatórios em linha com os valores de referência, sendo que uma delas apresentou melhores resultados do que a variante original. Em relação aos algoritmos probabilísticos propostos não foi possível tirar conclusões a nível de estimativa de piso, para o algoritmo fundamentado na distribuição baseada em histograma pois os resultados obtidos para a estimativa do edifício não eram os esperados, tendo sido necessário explorar outras metodologias. Para o algoritmo baseado na distribuição de visibilidade dos APs e na distribuição de *Kernel* foram obtidos bons resultados para a estimativa do edifício, nomeadamente 100%. No que diz respeito à estimativa do piso, o algoritmo da distribuição *Kernel* é o que apresenta melhores resultados de entre os métodos probabilísticos.

Os objetivos propostos inicialmente foram atingidos, resultando numa implementação Java de algoritmos determinísticos e probabilísticos. A ferramenta poderá ser expandida e usada no futuro na melhoria dos algoritmos propostos. Os métodos probabilísticos propostos não apresentaram resultados melhores que os determinísticos, mas tal não significa que não possam ser melhorados a ponto de poderem superá-los. Nomeadamente com a inclusão de informação histórica e não foi possível, por falta de tempo, explorar essa via, que aponta para trabalho futuro.

6.2. Trabalho futuro

A localização *indoor* é um tema atual, a merecer atenção da comunidade científica, para o qual surgem cada vez mais propostas. Há espaço para melhorias dos resultados obtidos.

Os algoritmos propostos poderão ter no futuro aperfeiçoamentos, nomeadamente no que diz respeito à estimativa do piso dos algoritmos probabilísticos. Uma melhoria na estimativa do piso, permitiria melhorar em muito o desempenho do algoritmo de

localização na medida em que possibilita a melhoria do erro médio final da estimativa da posição.

Os métodos probabilísticos fazem, essencialmente, uso do teorema de *Bayes* para o cálculo da probabilidade. Este teorema de *Bayes*, permite assim calcular a probabilidade de uma determinada *fingerprint* estar naquela posição. A probabilidade é dada através da razão entre a probabilidade da *fingerprint* pertencer àquela posição multiplicada pela probabilidade à *priori* daquela posição, e a probabilidade da *fingerprint.* Neste projeto, não tendo a probabilidade à *priori* da posição, no teorema de *Bayes* foi utilizado apenas a probabilidade da *fingerprint* pertencer àquela posição, considerando todas as posições como equiprováveis à partida.

Assim, fica como proposta para trabalho futuro a utilização de informação histórica nos algoritmos probabilísticos, desta forma é possível obter a probabilidade à *priori* daquela posição, permitindo calcular a probabilidade *bayesiana*. Esta proposta irá certamente melhorar os resultados, pois para além da informação atual, tem ainda acesso a informação anterior a essa.

Nos algoritmos probabilísticos apresentados nesta dissertação, uns apenas utilizavam os valores RSSI e outros apenas a deteção dos APs. Outra proposta para trabalho futuro seria, a junção destes dois. O desenvolvimento de um algoritmo, que não só dependesse dos valores RSSI como também, de um AP ter sido ou não detetado.

Apêndice A

Análise detalhada- kPM baseado na visibilidade dos APs

Aquando da implementação do algoritmo probabilístico baseado na visibilidade dos APs, foi verificado que para algumas *fingerprints* de teste as probabilidades das posições eram nulas, influenciando na estimativa da posição. Estas probabilidades nulas resultavam do facto de um AP ser observado na *fingerprint* de teste e não ser na *fingerprint* de treino, o que tornava de imediato a probabilidade dessa posição nula.

De forma a comprovar que as probabilidades eram nulas em todas as posições, foi selecionada uma das *fingerprint* de teste onde tal acontecia, Tabela 21.

Tabela 21- Fingerprint de teste (Distribuição de Visibilidade dos APs)

APs	*Longitude*	*Latitude*	*Floor*	*Building*
RSSI	-7383,867221	4864839,74	4	2

Esta *fingerprint* de teste, possui 12 APs observados, isto é 12 APs onde o valor RSSI é diferente de 100. A Tabela 22 contém esses 12 APs observados e os respetivos valores RSSI.

Tabela 22 - APs observados na fingerprint de teste

AP	*RSSI*	*AP*	*RSSI*	*AP*	*RSSI*	*AP*	*RSSI*
11	-87	**84**	-88	**139**	-91	**248**	-93
12	-87	**85**	-87	**144**	-57	**277**	-95
65	-94	**114**	-95	**145**	-56	**342**	-89

Foram selecionadas 5 posições que se encontrassem próximas dessa *fingerprint* de teste embora algumas pertençam a um piso diferente, Tabela 23.

Tabela 23 - Posições selecionadas

Posição	*Longitude*	*Latitude*	*Edifício*	*Piso*	*APs*
P1	-7384.3934	4864778.287	2	3	Probabilidade para cada AP
P2	-7383.2656	4864777.661	2	3	Probabilidade para cada AP
P3	-7384.3934	4864778.287	2	4	Probabilidade para cada AP
P4	-7383.7331	4864842.192	2	3	Probabilidade para cada AP
P5	-7384.7291	4864840.396	2	4	Probabilidade para cada AP

Na Figura 41 encontram-se representadas as coordenadas das 7 posições selecionadas (pontos a azul- posições do piso 4, pontos amarelos- posições do piso 3) e as coordenadas da *fingerprint* de teste (ponto cinza com padrão xadrez).

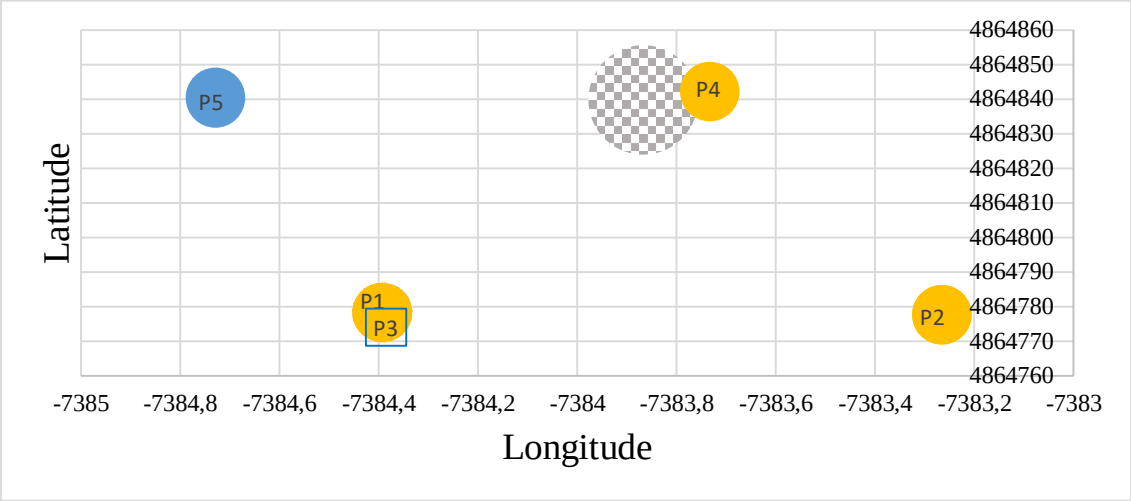

Figura 41 - Representação das coordenadas no plano cartesiano

Cada posição observa um certo número de APs, tendo uma certa probabilidade associada. Os APs não observados possuem probabilidade de 0.0. Entre estes APs que observa, apenas alguns são comuns aos APs observados na *fingerprint* de teste.

- Posição 1 observa 7 APs dos 12 observados na *fingerprint* de teste;
- Posição 2 observa 7 APs dos 12 observados na *fingerprint* de teste;
- Posição 3 observa 8 APs dos 12 observados na *fingerprint* de teste;
- Posição 4 observa 9 APs dos 12 observados na *fingerprint* de teste;
- Posição 5 observa 8 APs dos 12 observados na *fingerprint* de teste;

Nos APs observados na *fingerprint* de teste e que não constam na tabela dos APs observados das posições, terão uma probabilidade de 0.0. No cálculo da probabilidade destas 5 posições, ao multiplicar as probabilidades dos APs, esta fica com o valor 0.0, Figura 42. Tal acontece, pois nenhuma destas posições possui exatamente os mesmos APs observados do que a *fingerprint* de teste.

Sendo as probabilidades todas de 0.0, tem grande influência nos resultados visto que as *fingerprints* não serão ordenadas.

```
-7383.2656 4864777.661 2 3 0.0
-7384.3934 4864778.287 2 4 0.0
-7383.7331 4864842.192 2 3 0.0
-7384.7291 4864840.396 2 4 0.0
-7384.3934 4864778.287 2 3 0.0
```

Figura 42 - Resultado obtido para as 5 posições

Com o intuito de resolver esta questão dos 0.0, foi utilizado um valor significativamente baixo, de 0.01 para substituir os APs de teste que não foram observados nas posições. Usando este baixo valor como alternativa, foram obtidas as probabilidades presentes na Figura 43. Desta forma é possível obter 100% de acerto nos edifícios, selecionando a primeira posição.

```
-7383.7331 4864842.192 2 3 8.238683127572017E-10
-7384.7291 4864840.396 2 4 6.250000000000001E-10
-7384.3934 4864778.287 2 4 3.5000000000000003E-10
-7384.3934 4864778.287 2 3 4.590000000000001E-12
-7383.2656 4864777.661 2 3 3.6000000000000004E-12
```

Figura 43 - Probabilidades ordenadas

Através da Figura 43 é possível verificar que a posição 4 pertencente ao piso 3 apresenta maior probabilidade do que a posição 5 e 3 que pertencem ao mesmo piso da *fingerprint* de teste, piso 4. Isto deve-se ao facto da posição 4 observar 9 dos 12 APs de teste, enquanto que a posição 5 e 3 observam ambas 8 APs. O número de APs não observados será maior, o que faz com que a probabilidade seja menor, pois multiplicando valores baixos (valor utilizado 0.01), obtêm-se valores ainda mais baixos.

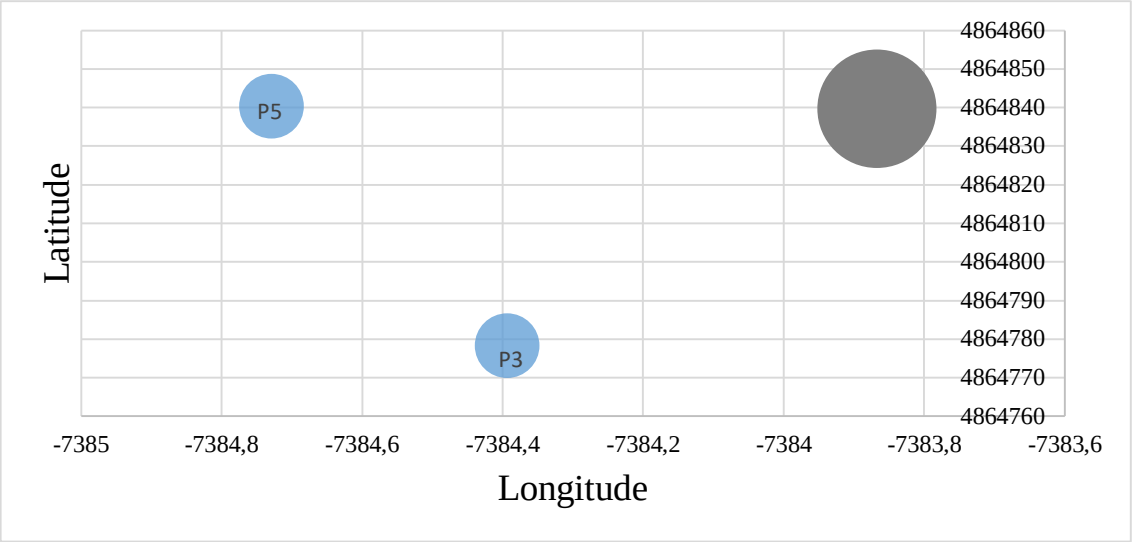

Figura 44 - Posição 3 e 5 pertencentes ao mesmo edifício e piso da fingerprint de teste

Na Figura 44 são apresentadas a posição 3 e 5 que correspondem às *fingerprints* de treino com o mesmo piso da *fingerprint* de teste. Na figura a posição 3 encontra-se mais próxima da *fingerprint* de teste do que a posição 5, logo por norma esta deveria ser escolhida. No entanto a posição 3 apresenta menor probabilidade do que a posição 5, o que faz com que, na prática, a posição 5 seja a escolhida. Este acontecimento deve-se à multiplicação das probabilidades dos APs observados em teste. Cada uma destas posições, 3 e 5, tem exatamente 8 APs observados que correspondem aos APs observados em teste, a multiplicação das probabilidades desses APs para a posição 3 dá 0.035, enquanto que para a posição 5 dá 0.0625 (valores aproximados), e como consequência terá mais probabilidade a posição 5.

Apêndice B

Resultados detalhados do Algoritmo Determinístico

Nas tabelas abaixo apresentadas encontram-se todos os resultados obtidos para as variantes propostas à função de similaridade do algoritmo determinístico. A Tabela 24 e a Tabela 25 apresentam os resultados da estimativa do piso e erro médio, respetivamente, para a similaridade Euclidiana com parâmetros adicionais. A Tabela 26 e Tabela 27 apresentam os resultados da estimativa do piso e erro médio, respetivamente, para a similaridade Euclidiana. A Tabela 28 e a Tabela 29 apresentam os resultados da estimativa do piso e erro médio, respetivamente, para a similaridade de *Manhattan*.

Tabela 24 - Resultados para a estimativa do piso para a Similaridade Euclidiana com parâmetros adicionais

defRSSI	*-90*			
k1	**30**	35	40	50
% Piso	**90.639%**	90.369%	90.369%	90.279%
defRSSI	*-95*			
k1	30	**35**	40	50
% Piso	90.459%	**90.639%**	90.459%	90.369%
defRSSI	*-102*			
k1	30	**35**	40	50
% Piso	90.189%	**90.459%**	90.279%	90.099%
defRSSI	*-105*			
k1	30	**35**	40	50
% Piso	90.099%	**90.369%**	90.279%	90.099%
defRSSI	*-110*			
k1	30	**35**	40	50
% Piso	89.829%	**90.279%**	90.009%	89.829%

Tabela 25 - Resultados para estimativa das coordenadas para a Similaridade Euclidiana com parâmetros adicionais

defRSSI = -90 *k1=30*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	10.15	9.676	9.869	9.327	9.680	9.026	9.603	8.886	**9.571**	**8.822**

defRSSI = -95 *k1=35*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	10.02	9.604	9.779	9.296	9.612	9.019	9.601	8.943	**9.591**	**8.896**
defRSSI = -102 *k1=35*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	10.11	9.789	9.874	9.476	9.824	9.368	9.765	9.263	**9.700**	**9.191**
defRSSI = -105 *k1=35*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	10.17	9.872	9.963	9.606	9.916	9.489	9.855	9.400	**9.815**	**9.351**
defRSSI = -110 *k1=35*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	10.28	10.003	10.022	9.699	9.954	9.551	9.911	9.509	**9.848**	**9.431**

Tabela 26 - Resultados para a estimativa do piso para a Similaridade Euclidiana

defRSSI	*-90*				
k1	30	35	40	**50**	60
% Piso	93.069%	92.979%	93.729%	**94.599%**	94.059%
defRSSI	*-95*				
k1	30	35	40	**50**	60
% Piso	91.989%	92.079%	92.889%	**93.969%**	93.609%
defRSSI	*-102*				
k1	30	35	40	**50**	60
% Piso	91.449%	91.449%	92.349%	**93.699%**	93.609%
defRSSI	*-105*				
k1	30	35	40	**50**	60
% Piso	91.269%	91.269%	92.079%	**93.609%**	93.519%
defRSSI	*-110*				
k1	30	35	40	**50**	60
% Piso	90.819%	90.909%	91.989%	**93.429%**	93.339%

Tabela 27 - Resultados para estimativa das coordenadas para a Similaridade Euclidiana

defRSSI = -90 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	*WkNN*	*kNN*	***WkNN***
Erro	7.832	7.834	7.663	7.652	7.568	7.551	**7.476**	7.453	7.477	**7.440**
defRSSI = -95										

k1=50										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	7.884	7.866	7.585	7.593	7.535	7.524	7.481	7.461	**7.473**	**7.428**
defRSSI = -102										
k1=50										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.156	8.185	7.983	7.985	7.922	7.912	7.885	7.858	**7.871**	**7.826**
defRSSI = -105										
k1=50										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.317	8.317	8.155	8.152	8.102	8.090	8.099	8.071	**8.011**	**7.973**
defRSSI = -110										
k1=50										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.613	8.613	8.479	8.471	8.417	8.403	8.379	8.354	**8.360**	**8.323**

Tabela 28 - Resultados para a estimativa do piso para a Similaridade de Manhattan

defRSSI	*-90*				
k1	30	35	40	**50**	60
% Piso	92.719%	91.72%	92.799%	**93.879%**	93.789%
defRSSI	*-95*				
k1	30	35	40	**50**	60
% Piso	90.639%	90.909%	92.169%	**93.429%**	93.249%
defRSSI	*-102*				
k1	30	35	40	**50**	60
% Piso	90.909%	91.179%	92.259%	**93.339%**	93.159%
defRSSI	*-105*				
k1	30	35	40	**50**	60
% Piso	90.819%	91.089%	92.349%	**93.249%**	93.159%
defRSSI	*-110*				
k1	30	35	40	**50**	60
% Piso	90.819%	91.089%	92.439%	**93.249%**	92.979%

Tabela 29 - Resultados para estimativa das coordenadas para a Similaridade Manhattan

defRSSI = -90										
k1=50										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	*WkNN*	*kNN*	***WkNN***
Erro	8.077	8.083	7.834	7.876	7.741	7.726	**7.689**	7.659	7.701	**7.657**
defRSSI = -95										
k1=50										

k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*kKNN*	***kNN***	*WkNN*	*kNN*	***WkNN***
Erro	8.051	8.064	7.824	7.821	7.761	7.739	**7.743**	7.705	7.743	**7.682**
defRSSI = -102 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***	*kNN*	*WkNN*
Erro	8.331	8.347	8.156	8.147	8.065	8.049	**8.040**	**8.006**	8.094	8.025
defRSSI = -105 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.729	8.423	8.267	8.249	8.268	8.241	8.222	8.181	**8.185**	**8.123**
defRSSI = -110 *k1=50*										
k2	3		5		7		9		11	
	kNN	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	*kNN*	*WkNN*	***kNN***	***WkNN***
Erro	8.590	8.609	8.395	8.392	8.438	8.407	8.396	8.357	**8.372**	**8.308**

Apêndice C

Resultados detalhados do Algoritmo kPM: Distribuição baseada em Histograma

As tabelas abaixo apresentadas, Tabela 30, Tabela 31 e Tabela 32, apresentam os resultados obtidos para a estimativa do piso para diferentes valores de k1 (amostras mais prováveis) do algoritmo probabilístico fundamentado numa distribuição baseada em histograma.

Tabela 30 - Estimativa do edifício algoritmo kPM (Distribuição baseado em histograma) para k1=1

	-5 … **+5**	**-10** … **+10**	**-15** … **+15**	**-20** … **+20**	**-25** … **+25**	**-30** … **+30**	**-35** … **+35**	**-40** … **+40**
-85	62.736%	82.538%	86.229%	79.388%	72.367%	50.945%	34.833%	25.653%
-90	63.096%	78.758%	85.959%	87.129%	84.968%	72.097%	53.195%	37.894%

-95	61.656%	73.177%	83.528%	88.209%	88.929%	86.859%	72.187%	56.436%
-100	55.896%	66.877%	77.228%	85.149%	89.469%	88.389%	87.219%	60.036%
-105	53.555%	60.396%	70.567%	79.298%	86.769%	89.919%	88.389%	87.309%
-110	53.555%	57.336%	63.726%	72.457%	80.918%	87.399%	**90.189%**	88.569%

Tabela 31 - Estimativa do edifício algoritmo kPM (Distribuição baseado em histograma) para k1=6

	-5 … +5	**-10 … +10**	**-15 … +15**	**-20 … +20**	**-25 … +25**	**-30 … +30**	**-35 … +35**	**-40 … +40**
-85	51.665%	79.748%	86.589%	79.118%	69.757%	50.855%	42.754%	39.244%
-90	52.565%	75.247%	85.959%	87.849%	85.869%	75.428%	60.576%	48.785%
-95	48.155%	65.797%	79.118%	87.309%	89.649%	85.059%	71.737%	56.665%
-100	39.784%	65.797%	70.297%	80.828%	87.759%	91.179%	90.909%	71.647%
-105	36.994%	54.995%	58.776%	71.467%	81.368%	88.029%	91.629%	91.629%
-110	36.904%	44.104%	49.685%	59.766%	72.367%	81.728%	88.119%	**91.719%**

Tabela 32 - Estimativa do edifício algoritmo kPM (Distribuição baseado em histograma) para k1=30

	-5 … +5	**-10 … +10**	**-15 … +15**	**-20 … +20**	**-25 … +25**	**-30 … +30**	**-35 … +35**	**-40 … +40**
-85	60.667%	80.828%	88.929%	84.878%	67.417%	57.516%	42.754%	48.785%
-90	62.016%	77.228%	87.759%	90.099%	85.059%	72.187%	73.177%	59.676%
-95	60.036%	71.917%	82.628%	90.099%	90.369%	86.499%	75.608%	73.807%
-100	54.995%	65.077%	76.148%	85.059%	91.179%	90.819%	87.399%	78.308%
-105	52.925%	59.226%	68.947%	78.848%	86.499%	91.629%	90.819%	88.119%
-110	52.925%	56.166%	62.466%	71.287%	80.468%	87.219%	**91.899%**	90.819%

Apêndice D

Análise complementar ao Algoritmo kPM: Distribuição baseada em Histograma

Foi selecionada uma *fingerprint* de teste, Tabela 33.

Tabela 33- Fingerprint de teste Distribuição baseada em histograma

APs	*Longitude*	*Latitude*	*Floor*	*Building*
RSSI	-7345,08517	4864830,817	0	2

Esta *fingerprint* de teste, aleatoriamente selecionada, possui 23 APs observados, isto é 23 APs onde o valor RSSI é diferente de 100. A Tabela 34 contém esses 23 APs observados e os respetivos valores RSSI.

Tabela 34 - APs observados na fingerprint de teste

AP	*RSSI*	*AP*	*RSSI*	*AP*	*RSSI*	*AP*	*RSSI*
59	-87	**70**	-83	**98**	-79	**360**	-91
60	-86	**73**	-88	**99**	-79	**489**	-77
63	-88	**79**	-89	**121**	-91	**495**	-72
64	-89	**87**	-92	**203**	-66	**496**	-89
67	-75	**96**	-40	**204**	-65	**501**	-54
69	-90	**97**	-41	**303**	-89		

Foram selecionadas 7 posições que se encontrassem próximas da *fingerprint* de teste, todas elas pertencentes ao mesmo edifício e piso, Tabela 35.

Tabela 35 - Posições selecionadas

Posição	*Longitude*	*Latitude*	*Edifício*	*Piso*	*APs*
P1	-7343.870905	4864745.745	2	0	Histograma para cada AP
P2	-7346.1826	4864759.053	2	0	Histograma para cada AP
P3	-7342.4353	4864756.958	2	0	Histograma para cada AP
P4	-7345.4118	4864829.551	2	0	Histograma para cada AP
P5	-7351.2667	4864837.355	2	0	Histograma para cada AP
P6	-7351.981125	4864843.493	2	0	Histograma para cada AP
P7	-7360.5393	4864837.944	2	0	Histograma para cada AP

Na Figura 45 encontram-se representadas as coordenadas das 7 posições selecionadas (pontos a azul) e as coordenadas da *fingerprint* de teste (ponto cinza com padrão xadrez).

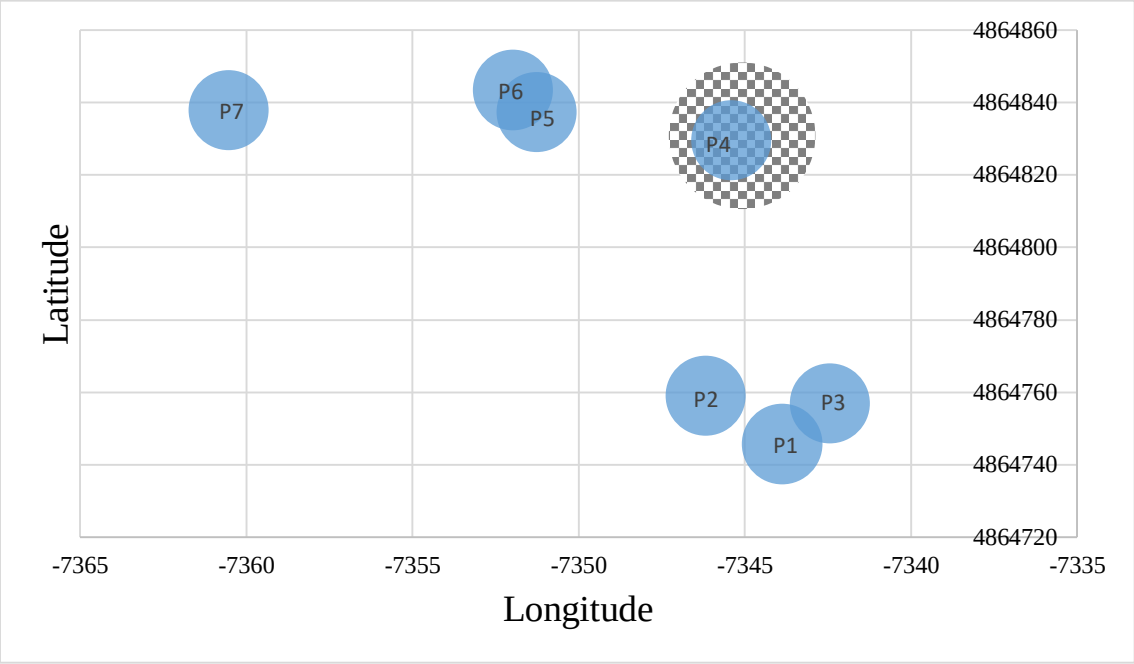

Figura 45 - Representação das coordenadas no plano cartesiano, Distribuição baseada em histograma

Analisando a Figura 45, supõe-se que a posição mais provável a ser escolhida seria a posição 4.

Foram realizados testes para *defRSSI*=-105 pois este é considerado como uma intensidade de sinal bastante fraca.

Na posição 1 existem 8 APs (AP 67, AP 68, AP 73, AP 74, AP 277, AP 332, AP 389 e AP 486) em que a probabilidade para o *defRSSI* é 0.0, o que significa que estes APs foram sempre observados. Existem 13 APs onde o histograma possui probabilidade para o *defRSSI* e para outros valores RSSI, sendo que a médias das probabilidades para *defRSSI* ronda os 59,6%. Os restantes 499 APs, no seu histograma, apenas possuem probabilidade para o *defRSSI*, sendo a probabilidade sempre de 100%, o que significa que estes 499 APs nunca foram observados.

Para um intervalo de RSSI-5 até RSSI+5,

O AP mais forte de teste detém um valor de RSSI de -40dBm e corresponde ao AP 96, este AP para a posição 1 possui o histograma presente na Figura 46, ('…' representa no histograma os restantes valores RSSI).

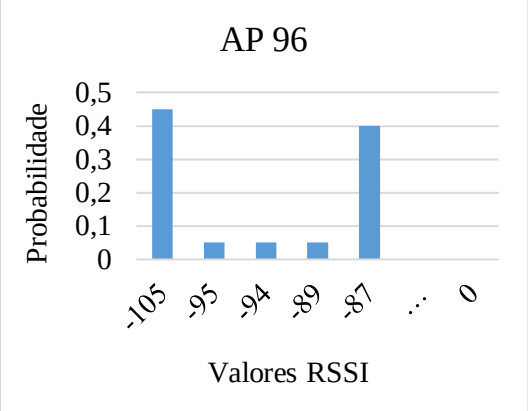

Figura 46- Histograma do AP 96 da Posição 1

Independentemente do intervalo utilizado, a probabilidade desta posição será sempre zero, pois no histograma deste AP não existe probabilidade para um valor abrangido pelo intervalo, Tabela 36. Como a probabilidade final da posição é dada pela multiplicação das probabilidades dos 520 APs, basta que a probabilidade de um deles seja zero, para que a probabilidade final seja 0.0, independentemente das probabilidades dos restantes APs. Se os APs mais fortes de cada posição não tiverem probabilidade no histograma, a probabilidade desta posição será automaticamente zero.

Tabela 36- Intervalos de valores para o AP 96

	RSSI-r a RSSI+r	*Histograma AP 96*
r=5	-45 a -35	Não abrange os valores do histograma
r=10	-50 a -30	Não abrange os valores do histograma
r=15	-55 a -25	Não abrange os valores do histograma
r=20	-60 a -20	Não abrange os valores do histograma
r=25	-65 a -15	Não abrange os valores do histograma
r=30	-70 a -10	Não abrange os valores do histograma
r=35	-75 a -5	Não abrange os valores do histograma
r=40	-80 a 0	Não abrange os valores do histograma

O valor de *r* vai até 40, valor utilizado em testes realizados, no entanto este valor faz com que seja criado um intervalo de 80 valores.

O mesmo acontece para a posição 2 e 3, pois o intervalo também não abrange os valores do histograma cuja probabilidade é não nula, como se pode verificar na Figura 47.

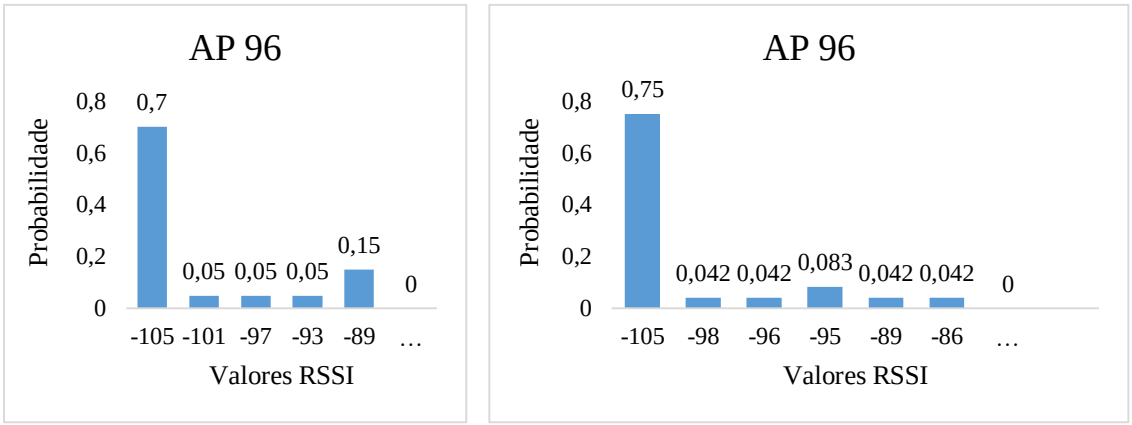

Figura 47 - Histograma do AP 96 para a Posição 2 (Esquerda) e Posição 3 (Direita)

As restantes 4 posições possuem valores RSSI, no AP 96, que abrangem os intervalos de valores, no entanto acaba por acontecer o mesmo, não propriamente neste AP mas noutros. Desta forma, para um intervalo de 10 valores, vai tornar todas as probabilidades a 0.0, assim como foi testado, Figura 48.

```
-7351.981125 4864843.493 2 0 0.0
-7342.4353 4864756.958 2 0 0.0
-7345.4118 4864829.551 2 0 0.0
-7351.2667 4864837.355 2 0 0.0
-7343.870905 4864745.745 2 0 0.0
-7360.5393 4864837.944 2 0 0.0
-7346.1826 4864759.053 2 0 0.0
```

Figura 48 - Resultados obtidos para as 10 posições com um intervalo de 10 valores (RSSI-5 a RSSI+5)

Como as probabilidades são todas a zero, a ordenação das posições por probabilidade não vai alterar o aspeto apresentado na Figura 48. Ao selecionar as *k* amostras, supostamente as *k* mais semelhantes, vão ser escolhidas as primeiras, que neste caso não há garantia que sejam as melhores, o que influencia os resultados finais.

Visto que com intervalos de valores pequenos, as probabilidades são todas nulas, foi utilizado um intervalo de 80 (RSSI-40 a RSSI+40), e desta forma foram obtidas 4 probabilidades não nulas, Figura 49. A quarta é exatamente a posição 4, a posição mais próxima da *fingerprint* de teste, segundo a Figura 45, e que supostamente seria a escolhida por ser a mais próxima.

```
-7360.5393 4864837.944 2 0 0.7735555555555594
-7351.981125 4864843.493 2 0 0.7666666666666657
-7351.2667 4864837.355 2 0 0.4999999999999994
-7345.4118 4864829.551 2 0 0.2771352284324518
-7342.4353 4864756.958 2 0 0.0
-7343.870905 4864745.745 2 0 0.0
-7346.1826 4864759.053 2 0 0.0
```

Figura 49 - Resultados obtidos para as 10 posições com um intervalo de 80 valores (RSSI-40 a RSSI+40)

A posição 4, apesar de ser a posição mais próxima, é a posição que apresenta menor probabilidade em relação às restantes.

Segundo a Figura 49, as posições que apresentam probabilidade é a posição 7, a posição 6, a posição 5 e a posição 4, respetivamente. As restantes posições, posição 3,1 e 2 apresentam probabilidade zero, o que comprova aquilo que foi dito anteriormente, que independentemente do intervalo a sua probabilidade seria zero.

Na posição 7 existem 3 APs (AP 203, AP 204 e AP 501) em que a probabilidade para o *defRSSI* é 0.0, o que significa que estes APs foram sempre observados. Existem 35 APs onde o histograma possui probabilidade para o *defRSSI* e para outros valores RSSI, sendo que a médias das probabilidades para *defRSSI* ronda os 50.4%. Os restantes 482 APs, no seu histograma, apenas possui probabilidade para o *defRSSI*, sendo a probabilidade sempre de 100%, o que significa que estes 482 APs nunca foram observados.

Na posição 6 existem 4 APs (AP 96, AP 97, AP 203 e AP 204) em que a probabilidade para o *defRSSI* é 0.0, o que significa que estes APs foram sempre observados. Existem 17 APs onde o histograma possui probabilidade para o *defRSSI* e para outros valores RSSI, sendo que a médias das probabilidades para *defRSSI* ronda os 57.9%. Os restantes 499 APs, no seu histograma, apenas possui probabilidade para o *defRSSI*, sendo a probabilidade sempre de 100%, o que significa que estes 499 APs nunca foram observados.

Na posição 5 existem 7 APs (AP 96, AP 97, AP 203, AP 204, AP 495, AP 501 e AP 516) em que a probabilidade para o *defRSSI* é 0.0, o que significa que estes APs foram sempre observados. Existem 21 APs onde o histograma possui probabilidade para o *defRSSI* e para outros valores RSSI, sendo que a médias das probabilidades para *defRSSI* ronda os 56.6%. Os restantes 492 APs, no seu histograma, apenas possui probabilidade para o *defRSSI*, sendo a probabilidade sempre de 100%, o que significa que estes 492 APs nunca foram observados.

Por fim, na posição 4 não existem APs onde a probabilidade para o *defRSSI* é 0.0, o que significa que não existem APs que foram sempre observados. Existem 37 APs onde o histograma possui probabilidade para o *defRSSI* e para outros valores RSSI, sendo que a médias das probabilidades para *defRSSI* ronda os 45.8%. Os restantes 483 APs, no seu histograma, apenas possui probabilidade para o *defRSSI*, sendo a probabilidade sempre de 100%, o que significa que estes 483 APs nunca foram observados.

Tabela 37 - Análise das 3 posições com probabilidades não nula

	APs sempre observados Prob(-105)=0.0	*APs com probabilidade para defRSSI e outros valores de RSSI*	*APs nunca observados Prob(-105)=1.0*
Posição 7	3	35	482
Posição 6	4	17	499
Posição 5	7	21	492
Posição 4	0	37	483

Os dados recolhidos presentes na Tabela 37 permitem concluir o porque da posição 7 possuir maior probabilidade, seguida da posição 6 e depois a posição 5 e finalmente a posição 4.

Em primeiro lugar, a posição 4 não apresenta qualquer AP onde a probabilidade para -105 seja 0.0, não existe nenhum AP que tenha sido observado. Para além disso possui 37 APs com probabilidade para o valor RSSI de defeito, para além dos outros valores. Isto, juntamente com os 483 APs não observados uma única vez condiciona o valor final da probabilidade desta posição.

Para além disso, aquando da averiguação destes dados foi verificado que a posição 7, nos APs com probabilidades para o valor *defRSSI* e outros APs, a diversidade e quantidade de valores RSSI era maior do que para a posição 4. No caso do AP 501 por exemplo, para posição 4 o histograma contém 13 valores (Figura 50) e para a posição 7 o histograma contém 18 valores (Figura 51). Para intervalos grandes, quando é efetuado a soma dos valores desse intervalo para obter a probabilidade desse AP, será maior para a posição 7 do que a 4, pois possui mais valores para somar. (histogramas da Figura 50 e Figura 51 apresentam valores de probabilidades arredondados)

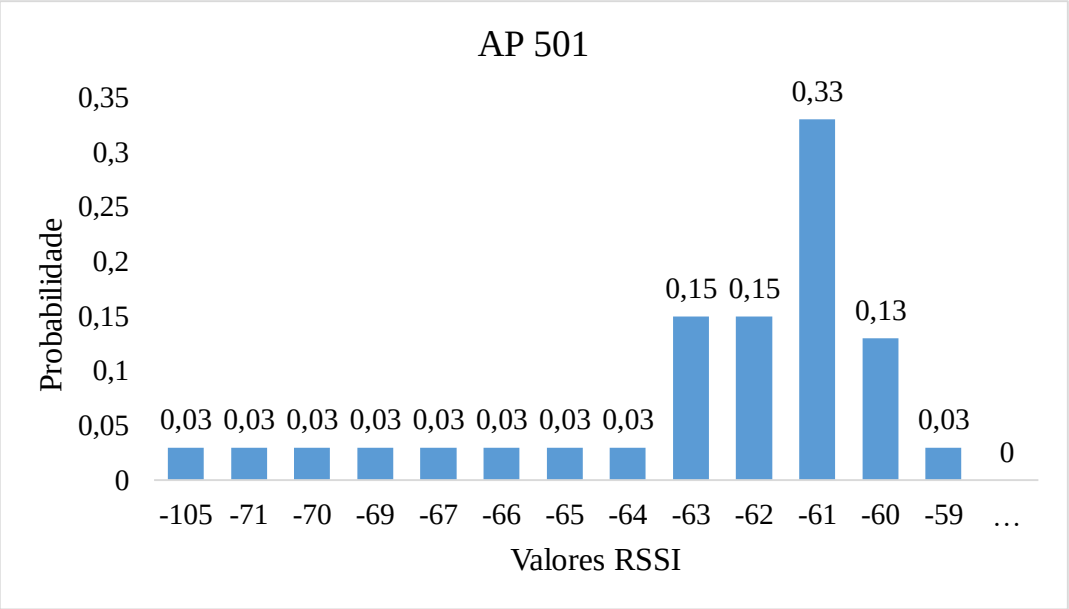

Figura 50- Histograma AP 501 Posição 4

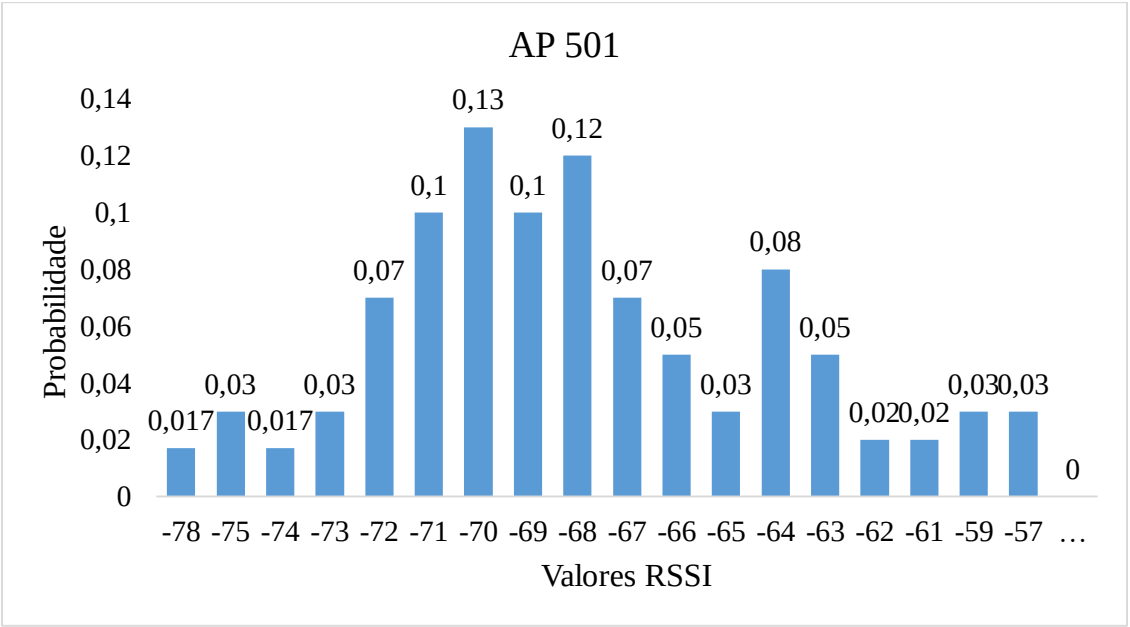

Figura 51 - Histograma AP 501 Posição 7

Para o caso deste AP, o valor RSSI do AP 501 na *fingerprint* de teste é -54dBm, para um intervalo de 80 valores (RSSI-40 a RSSI+40), obtendo assim um intervalo entre -94dBm a -14dBm. A probabilidade da posição 4 acaba por ser ligeiramente menor do que a probabilidade da posição 7.

Apêndice E

Análise complementar ao Algoritmo kPM: Distribuição de Visibilidade dos APs

A *fingerprint* de teste 14, Tabela 38, é uma das amostras cuja estimativa do piso para este algoritmo é falhada. Esta *fingerprint* possui 20 APs observados.

Tabela 38- Fingerprint de teste 14, algoritmo kPM baseado na visibilidade dos APs

APs	*Longitude*	*Latitude*	*Floor*	*Building*
RSSI	-7331,1001	4864766,922	2	2

Baseada nesta *fingerprint* de teste são calculadas as probabilidades para todas as posições, e estas após serem ordenadas por probabilidade apresentam a ordem visível na Figura 52.

```
-7370.775379 4864770.923 2 1 0.9357425742574257
-7363.159 4864768.472 2 1 0.8737760327756913
-7362.7832 4864764.929 2 1 0.8674257425742573
-7366.4499 4864770.297 2 1 0.8657284299858556
-7349.4756 4864760.88 2 1 0.8570297029702969
-7352.9408 4864762.803 2 1 0.851089108910891
-7359.4401 4864766.409 2 2 0.8476237623762375
-7331.4638 4864768.514 2 1 0.8430039525691698
-7371.374951 4864761.285 2 1 0.8275542406311636
-7330.097967 4864762.291 2 1 0.8180708661417322
-7344.375 4864757.61 2 2 0.8151181102362204
-7335.376828 4864753.121 2 1 0.8034782608695651
-7341.3081 4864756.174 2 2 0.8003543307086612
-7356.4576 4864764.754 2 2 0.7995256916996046
-7404.4643 4864809.153 2 3 0.7964173228346455
-7375.1481 4864773.737 2 1 0.7928382838283827
-7339.191617 4864755.251 2 1 0.7913725490196076
-7340.1914 4864758.186 2 1 0.7851427316644707
-7333.861493 4864760.14 2 1 0.7826377952755904
-7343.7335 4864754.511 2 2 0.7796850393700786
-7331.4005 4864768.479 2 2 0.7787747035573122
```

Figura 52- Posições ordenadas por probabilidade na estimativa do piso, algoritmo kPM baseado na visibilidade dos APs

Como é possível verificar na Figura 52, as primeiras posições, as que apresentam maior probabilidade não pertencem ao piso 2, mas sim ao 1. Entre o conjunto de 21 posições acima apresentado apenas existem 6 posições cujo piso é o 2, mas estas encontram-se em 7º, 11º, 13º, 14º, 20º e 21º lugar, respetivamente.

Foram selecionadas um conjunto de posições e realizada uma análise complementar, de forma a perceber o porquê destes resultados. Foi então selecionada a primeira posição do conjunto apresentado na Figura 52, e as posições presentes no 7º, 11º e 21º lugar que pertencem ao piso certo, Tabela 39.

Tabela 39 - Posições selecionadas, algoritmo kPM baseado na visibilidade dos APs

Posição	*Longitude*	*Latitude*	*Edifício*	*Piso*	*APs*
P1	-7370.775379	4864770.923	2	1	Probabilidade para cada AP
P2	-7359.4401	4864766.409	2	2	Probabilidade para cada AP
P3	-7344.375	4864757.61	2	2	Probabilidade para cada AP
P4	-7331.4005	4864768.479	2	2	Probabilidade para cada AP

Na Figura 53 encontram-se representadas as coordenadas das 4 posições selecionadas (pontos a azul – posições pertencentes ao mesmo piso, ponto azul com padrão às riscas – posição pertencente a um piso diferente) e as coordenadas da *fingerprint* de teste (ponto cinza com padrão xadrez).

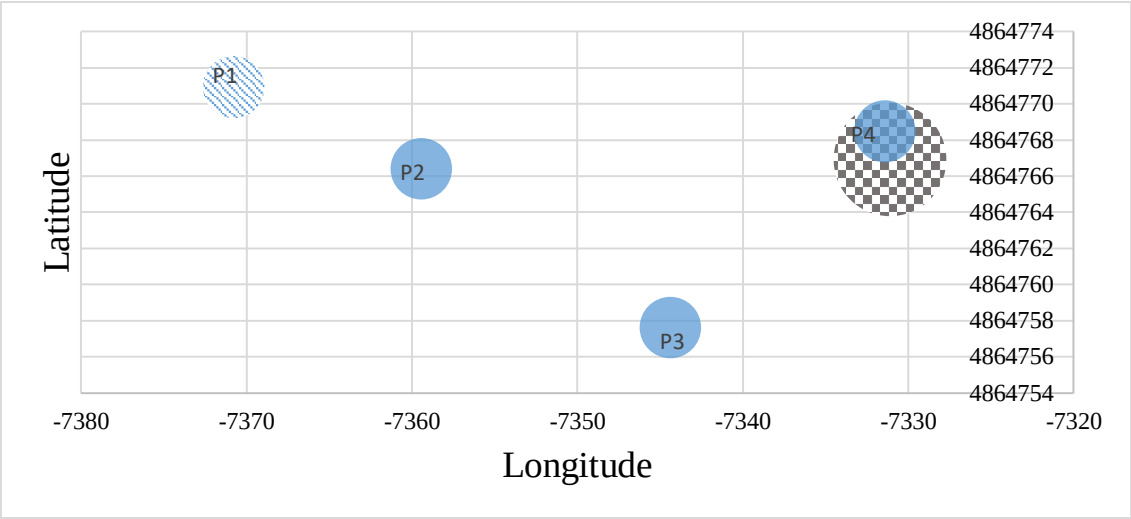

Figura 53 - Representação das coordenadas no plano cartesiano, algoritmo kPM baseado na visibilidade dos APs

Tal como seria de esperar, a P1 é a que se encontra mais afastada da *fingerprint* de teste, pois pertence a um piso diferente, as restantes como pertencem ao mesmo piso que a *fingerprint* de teste, encontram-se mais próximas. A P4 é a que se encontra mais próxima da *fingerprint* de teste, no entanto do conjunto apresentado na Figura 52, é das que tem menor probabilidade.

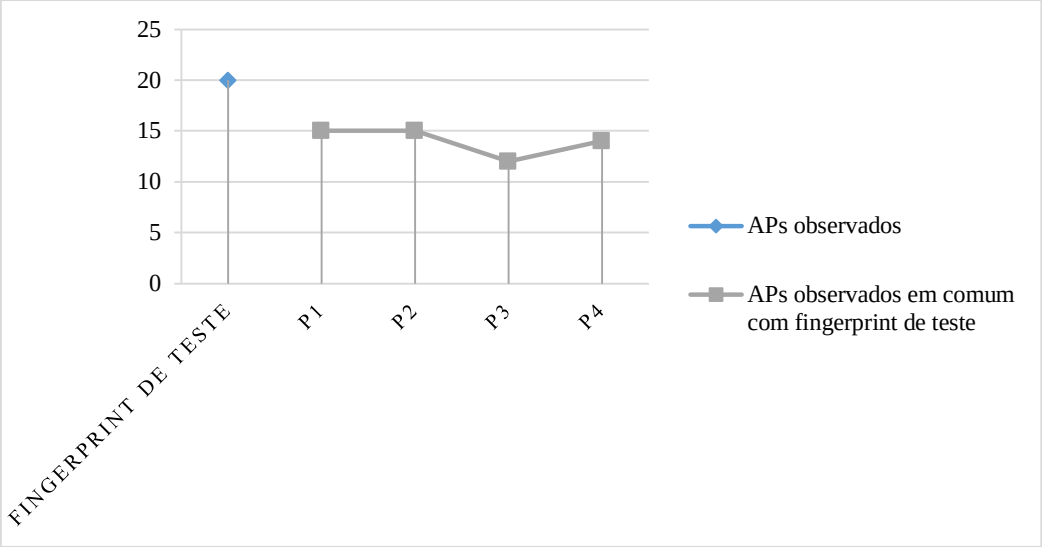

Figura 54 - APs observados na fingerprint de teste e nas posições

A P1 e a P2 contêm 15 dos 20 APs observados na *fingerprint* de teste, a P3 contém 12 e a P4 contém 14, Figura 54 . Como já foi referido, a probabilidade de cada posição é obtida através da razão entre o somatório das probabilidades de todos os APs observados na *fingerprint* de teste e o número de APs da posição P que não foram observados em comum com a *fingerprint* de teste (520 – APs observados em comum). Em primeiro lugar, a P1 e a P2 ao possuírem mais APs observados em comum com a *fingerprint* de teste, diminuem o denominador e como consequência aumentam o valor final da probabilidade. Por outro lado, a soma das probabilidades dos APs observados em comum com a *fingerprint* de teste (já utilizando os pesos mediante o valor RSSI observado na *fingerprint* de teste), é significativamente maior para a P1 do que a P4 (a que se encontra mais perto). Ou seja, a P1 possui um numerador maior e denominador menor, o que resulta numa maior probabilidade. Para a P1 a soma das probabilidades dá aproximadamente 472.5, que dividindo pelos 520-15, dá aproximadamente uma probabilidade de 0.9356. Para a P4 a soma das probabilidades dá aproximadamente 394, que dividindo pelos 520-14, dá

aproximadamente uma probabilidade final de 0.7787. As probabilidades resultantes são exatamente as que são visíveis na Figura 52.

Ainda na tentativa de obter melhorias neste algoritmo, foi realizada uma outra experiência. Antes de calcular a probabilidade para cada posição baseado na *fingerprint* de teste, foram retirados da *fingerprint* de teste todos os APs cuja força de sinal é muito fraca (por exemplo -90dBm ou -95dBm), ou seja, considerar que esses APs não foram observados. Depois de remover esses APs era então calculada a probabilidade através dos APs observados na *fingerprint* de teste. Esta experiência não apresentou melhorias em relação à proposta em questão, acabando por ser descartada.

Referências Bibliográficas

[1] António Carlos Macedo de Sousa, " Localization as a Network Service," 2015.

[2] B. Miguel Torres Lopes, "Algoritmos de localização com informação histórica e realimentação dos utilizadores," 2014.

[3] João André Félix Soares Torres da Silva , " Localização Em Redes Wi-Fi .," 2010.

[4] João Paulo da Fonseca Fernandes, " Localização em Redes Wi-Fi," 2012.

[5] "Current competition - Evaluating AAL Systems - EvAAL ..::Evaluating AAL Systems - EvAAL::.." [Online]. Available: http://evaal.aaloa.org/. [Accessed: 26-Apr-2016].

[6] A. Moreira, M. J. Nicolau, F. Meneses, and A. Costa, "Wi-Fi Fingerprinting in the Real World – RTLS@UM At the EvAAL Competition", 2015.

[7] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*. 2007.

[8] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*. 2013.

[9] M. Bouet and A. L. dos Santos, "RFID Tags : Positioning Principles and Localization Techniques." *1st IFIP Wireless Days*, 2008.

[10] X. Mingzhe, C. Jiabin, S. Chunlei, L. Nan, and C. Kong, "The indoor positioning algorithm research based on improved location fingerprinting," *The 27th Chinese Control and Decision Conference*, 2015.

[11] T. Roos, P. Myllymäki, H. Tirri, and P. Misikangas, "A Probabilistic Approach to WLAN User Location Estimation," *Int. J. Wirel. Inf. Networks*, vol. 9, 2002.

[12] J. D. Kelly, Jr. & L. Davis, "A Hybrid Genetic Algorithm for Classification." In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[13] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat, “Location determination of a mobile device using IEEE 802.11b access point signals,” *2003 IEEE Wirel. Commun. Networking, 2003. WCNC 2003.*, vol. 3, 2003.

[14] T. N. Lin and P. C. Lin, “Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks,” *Wirel. Networks, Commun. Mob. Comput. 2005 Int. Conf.*, vol. 2, 2005.

[15] M. Brunato and R. Battiti, “Tian Yingjie - Statistical learning theory for location fingerprinting in wireless LANs - ResearchGate,” no. October 2004.

[16] U. Ahmad, A. Gavrilov, S. Lee, and Y.-K. Lee, “Modular Multilayer Perceptron For WLAN Based Localization.” In *International Joint Conference on Neural Networks*, 2006.

[17] M. W. Gardner and S. R. Dorling, “Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences,” *Atmos. Environ.*, 1998.

[18] “Financial Predictor via Neural Network - CodeProject.” [Online]. Available: http://www.codeproject.com/Articles/175777/Financial-predictor-via-neural-network. [Accessed: 26-Apr-2016].

[19] M. A. Youssef and A. Agrawala, “On the Optimality of WLAN Location Determination Systems University of Maryland,” *Analysis*, 2003.

[20] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, “Robotics-based location sensing using wireless ethernet,” *Wirel. Networks*, vol. 11 , 2005.

[21] Mónica Beltrami, “Precificação de opções sobre ações por modelos de Support Vector Regression,” 2009.

[22] J. Barnes, C. Rizos, J. Wang, N. Gambale, “Locata : the positioning technology of the future,” *The 6th International Symposium on Satelite Navigation Technology Including Mobile Positioning & Location Services*, 2003.

[23] J. Barnes, C. Rizos, J. Wang, D. Small, G. Voigt, and N. Gambale, “Locata: A new positioning technology for high precision indoor and outdoor positioning,” *Ion Gpsgnss 2003*, 2004.

[24] L. M. Ni and A. P. Patil, “LANDMARC: indoor location sensing using active RFID,” *Proc. First IEEE Int. Conf. Pervasive Comput. Commun.*, 2003.

[25] V. Otsason, A. Varshavsky, A. Lamarca, and E. De Lara, “Accurate GSM Indoor Localization,” *Pervasive Mob. Comput.*, vol. 3, 2007.

[26] 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, 2000.

[27] M. Youssef and A.Agrawala, " The Horus WLAN location determination system." In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, 2005.

[28] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg, “COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses.” WiNTECH, 2006.

[29] M. Paciga and H. Lutfiyya, “Herecast : An Open Infrastructure for Location-Based Services Using WiFi,” *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, vol. 4, 2005.

[30] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, “ARIADNE: a dynamic indoor signal map construction and localization system,” *Proc. 4th Int. Conf. Mob. Syst. Appl. Serv. - MobiSys 2006*, 2006.

[31] “Trying to understand RSSI levels and how to read them.” [Online]. Available: http://www.metageek.com/training/resources/understanding-rssi.html. [Accessed: 20-Apr-2016].

[32] “Mathwords: Centroid.” [Online]. Available: http://www.mathwords.com/c/centroid.htm. [Accessed: 26-Apr-2016].

[33] “Dictionary of Algorithms and Data Structures.” [Online]. Available: https://xlinux.nist.gov/dads/.[Accessed: 2-May-2016].

[34] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, vol. 2 edition. Morgan Kaufman, 2005.

[35] T. Bagosi and Z. Baruch, “Indoor localization by WiFi,” *Proc. - 2011 IEEE 7th Int. Conf. Intell. Comput. Commun. Process. ICCP 2011*, 2011.

[36] “Java.lang.Math.exp() Method.” [Online]. Available: https://www.tutorialspoint.com/java/lang/math_exp.htm. [Accessed: 16-Sept-2016].

[37] H. X. Liu, B. A. Chen, P. H. Tseng, K. Ten Feng, and T. S. Wang, “Map-aware indoor area estimation with shortest path based on RSS fingerprinting,” *IEEE Veh. Technol. Conf.*, vol. 2015, 2015.

[38] “Introduction to Support Vector Machines — OpenCV 2.4.13.0 documentation.” [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed: 26-Apr-2016].