Universidade Federal
do Rio de Janeiro
Escola Politécnica

LOCALIZAÇÃO *INDOOR* VIA *KDE* EM ASSINATURAS DE RSSI

Eduardo Elael de Melo Soares

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Prof. Felipe Maia Galvão França, Ph.D.

Rio de Janeiro, RJ - Brasil

Agosto de 2013

LOCALIZAÇÃO INDOOR VIA KDE EM ASSINATURAS DE RSSI

Eduardo Elael de Melo Soares

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DOS GRAU DE ENGENHEIRO DE CONTROLE E AUTOMAÇÃO.

Examinados por:

_____

Prof. Felipe Maia Galvão França, Ph.D. (Orientador)

_____

Prof. Carlos Eduardo Pedreira, Ph.D.

_____

Prof. Priscila Machado Vieira Lima, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO de 2013

Soares, Eduardo Elael de Melo

Localização *Indoor* via *KDE* em Assinaturas de RSSI/ Eduardo Elael de Melo Soares – Rio de Janeiro: UFRJ/ ESCOLA POLITÉCNICA, 2013.

XI, 64 il. 29,7 cm

Orientador: Felipe Maia Galvão França

Projeto de Graduação – UFRJ/ POLI/ Engenharia de Controle e Automação, 2013.

Referências Bibliográficas: p. 63-65.

1. Localização *indoor*. 2. *KDE – Kernel Density Estimation*. 3. inferência bayesiana. 4. assinatura de RSSI. I. França, Felipe Maia Galvão. II. Universidade Federal do Rio de Janeiro, UFRJ, Engenharia de Controle e Automação. III. Localização *Indoor* via *KDE* em Assinaturas de RSSI.

AGRADECIMENTOS

Primeiramente gostaria de agradecer àqueles sem os quais nunca teria chegado à esse ponto de minha vida, ou seja, minha família. Sobretudo aos meus pais, Claudio Soares e João Henrique Taveira, e minha mãe, Lísia Melo, pelos valores morais, incentivo, carinho e direcionamento, tanto na vida pessoal como acadêmica; à minha avó, Therezinha Mello, por seu imensurável amor, dedicação e presença em minha vida; à minha dinda, Heloísa Melo, por me mostrar o valor da família, o verdadeiro sentido de dedicação ao próximo, e ser um grande exemplo de força de vontade e superação e ao meu avô, Nelson Mello, quem considero o meu mentor e ser humano no qual tento me espelhar. E sem dúvida a todos pelos ensinamentos, às vezes duros porém necessários, e apoio financeiro que me proporcionou a oportunidade de focar nos estudos, além de usufruir devidamente do meu tempo livre.

Não posso deixar de agradecer, também, a outras pessoas que foram essências na conclusão desse projeto, ao meu orientador, Felipe França, por me dar a oportunidade de trabalhar no projeto que veio a originar este trabalho, ao meu amigo e futuro empreendedor, Renan Salles de Freitas, que contribuiu fortemente para a conclusão desse trabalho e aos meus demais colegas de turma por todo os anos de apoio e amizade que possibilitaram minha caminhada nesse curso chegar até aqui.

Por fim, gostaria de agradecer aos demais amigos de fora do curso, que mesmo não agindo diretamente foram essenciais para que fosse capaz de manter a determinação, como o exímio designer, Diogo Camillo, que mesmo estando longe se manteve presente e sempre me proporcionou conversas engrandecedoras e o futuro engenheiro, Felipe Brasil Ramos, que depois de mais de quatorze anos estudando junto virá a se formar comigo.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos para a obtenção do grau de engenheiro de controle e automação.

Localização *Indoor* via *KDE* em Assinaturas de RSSI

Eduardo Elael de Melo Soares

Agosto/2013

Orientador: Prof. Felipe Maia Galvão França, Ph.D.

Curso: Engenharia de Controle e Automação

O projeto consiste em um sistema de localização *indoor* desenvolvido para rodar em *smartphones*, ele faz parte de um projeto maior de localização e rastreio robusto à variações do ambiente. Entretanto este projeto se restringe ao caso de localização do aparelho em estado imóvel, desconsiderando também variações do ambiente. O algoritmo se baseia no mapeamento das assinaturas RSSI, i.e. indicadores da potência do sinal, assim como suas variações, construindo, então, uma malha de pontos e associando à cada ponto uma distribuição de probabilidade. Esta distribuição deve ser estimada pelo método de *KDE – Kernel Density Estimation*. O algoritmo, então, cruza os dados de RSSI medidos em tempo real com as informações contidas nas distribuições de probabilidade, através de uma inferência Bayesiana, para determinar o local mais provável de origem dos vales de RSSI aferidos. O algoritmo foi capaz de obter uma acurácia de mais de 90%

na média para os testes realizados.

Palavras-chave: Localização *indoor*, *KDE – Kernel Density Estimation*, Inferência bayesiana, Assinatura de RSSI.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Control and Automation Engineer.

Indoor Localization via KDE of RSSI Fingerprints

Eduardo Elael de Melo Soares

August/2013

Advisor: Felipe Maia Galvão França

Course: Control and Automation Engineering

This project consists on the development of an *indoor* localization system targeting *smartphones*; it is part of a broader effort to create a localization and tracking system robust enough to overcome the environmental variability. However this project focus only on the steady case, i.e. no moving targets, it also disregards environmental mutability. The algorithm rely on the RSSI fingerprint mapping, which indicates de signal strength at a given point, and its fluctuations, setting up a grid with a probability distribution associated with each point. The distribution is to be estimated using *KDE – Kernel Density Estimation*. The real time acquired RSSI data is then crossed with every distribution on the database, through a Bayesian inference, to find out the most likely source point for the measured RSSI values. The algorithm scored a 90% mean accuracy for the performed experiments.

Keywords: Indoor localization, *KDE – Kernel Density Estimation*, Bayesian inference, RSSI fingerprint.

Sumário

1 Introdução 1
1.1 Motivação 1
1.2 Objetivos 1
1.3 Estrutura do Projeto 2
2 Sistemas de Localização 3
2.1 Panorama 3
2.1.1 Outdoor vs Indoor 3
2.2 Localização Indoor 5
2.2.1 Características do Ambiente 5
2.2.2 Características de um sinal eletromagnético 6
2.2.3 Localização vs Rastreio 13
2.2.4 Dificuldades práticas 15
2.3 Trabalhos Correlatos 17
3 Um Solução de Localização *Indoor* 19
3.1 Premissas do Projeto 19
3.1.1 Antena 19
3.1.2 Receptor estático 21
3.1.3 Desconhecimento de redes vizinhas 21
3.1.4 Não variabilidade do ambiente 22
3.1.5 Portabilidade 23
3.2 Algoritmo de Localização 23
3.2.1 Treinamento 24
3.2.2 KDE – *Kernel Density Estimation* 29
3.2.3 Comparação de probabilidades 38
3.3 Programação *Android*® 39
3.3.1 Java e uso de API's 39
3.3.2 Interface e funcionamento 39
4 Resultados Experimentais 43
4.1 Metodologia 43
4.1.1 Servidor *Matlab*® 44
4.1.2 Comportamento do canal 44
4.1.3 Variação temporal 45
4.1.4 Posição da Antena 45
4.2 Qualidade de Localização 45

4.2.1 Acurácia e variância 46
4.2.2 Sensibilidade à variação do número de modems 47
5 Conclusão e Trabalhos Futuros 50
5.1 Discussão 50
5.2 Possíveis Melhorias do Sistema Atual 50
5.2.1 Filtro da base de dados 50
5.2.2 Aprendizado 51
5.2.3 Uso da predição de movimento 51
5.2.4 Variação da Largura de Banda 51
5.2.5 Solução discreta 51
5.2.6 Transformação de espaços 52
5.2.7 *Clusterização* 52
5.2.8 Integração com outros sensores 52
6 Referências Bibliográficas 53
7 Anexo 56

ÍNDICE DE FIGURAS

Figura 1-1 Visão geral simplificada do projeto de localização. 1
Figura 2-1 Exemplo de classificação por KNN, onde sobre a nova amostra são avaliados os cinco vizinhos mais próximos (k=5). 11
Figura 2-2 Variação dos valores de potência medidos para uma rotação de 90º do receptor ao cuso de aproximadamente 12 horas. 12
Figura 2-3 Relação entre a potência e número de vezes que o sinal não é captado, em 60 mil amostras de 60 sinais. (p-valor aprox. -0,8) 16
Figura 3-1 Comparação entre o os ganhos de uma antena isotrópica ideal e uma omnidirecinal, posicionadas verticalmente no eixo Z. 20
Figura 3-2 Maior parte dos sinais recebidos em um ambiente comum não pertencem ao proprietário do sistema de localização. 22
Figura 3-3 Fluxograma do Sistema de Localização 24
Figura 3-4 Modelo lógico da base de dados, que irá conter as informações sobre os valores de RSSI amostrados. 25
Figura 3-5 Tabela local após quatro posições amostradas em cômodos diferentes. 25
Figura 3-6 Tabela Access_Point exibindo alguns dos BSSID amostrados. 26
Figura 3-7 Tabela KSD após a etapa intermediária de preparadação dos dados, já contendo o nome dos arquivos com as distribuições de probabilidade. 27
Figura 3-8 Tabela Sample contendo dados de diversos locais e pontos de acesso. 28
Figura 3-9 Exemplo de histograma dos valores de RSSI em um local, h=1 30
Figura 3-10 Exemplo de histograma dos valores de RSSI em um local, h=1 31
Figura 3-11 Estimação por *kernel* sem levar em consideração o suporte da distribuição subjacente. 36
Figura 3-12 Estimação após truncamento e renormalização sobre o suporte (-35,-100). 37
Figura 3-13 Estimação utilizando transformação para um suporte igual à (-35,-100). 37
Figura 3-14 Dimensionamento do ambiente em modo *off-line*. 40
Figura 3-15 Tela principal do aplicativo. 40
Figura 3-16 Escolha do método de localização. 41
Figura 3-17 Amostrando de valores de RSSI. 41
Figura 3-18 Exibindo a posição estimada pelo algoritmo selecionado. 42
Figura 4-1 Interface do servidor em *Matlab*® exibindo os 12 locais amostrados (escala em cm). 43
Figura 4-2 Aparente relação entre média e variância do RSSI. 44
Figura 4-3 Variação temporal do sinal (1 amostra/segundo). 45
Figura 4-4 Percentual de estimativas dentro de certa margem de erro. 46
Figura 4-5 Análise da presença de média dos pontos de acesso. Média feita sobre as amostras em todos os locais. 47
Figura 4-6 Variação da acurácia pela redução do número de pontos de acesso. 48

ÍNDICE DE TABELAS

Tabela 2-1 Acurácia média de posição em diferentes condições. 11
Tabela 3-1 Exemplo dos principais *kernels* com sua eficiência medida em termos de valor esperado do erro, em relação ao *kernel* ótimo 33
Tabela 3-2 Exemplo de extensões da função Gaussiana. 34

LISTA DE ABREVIATURAS

KDE – *Kernel Density Estimation*

Wi-Fi – Wireless Fidelity

GPS – Global Positioning System

UWB – Ultra-Wide Band

SNR – Signal Noise Ratio

LoS – Line of Sight

FM – Frequency Modulation

API – Application Programming Interface

RSSI – Received Signal Strength Indication

MAC – Media Access Control

LAN - Local Area Network

Pdf – Probability Density Function

IGES-CAD – Initial Graphics Exchange Specification (Computer-Aided Design)

RFID – Radio Frequency Identification

1 Introdução

1.1 Motivação

Esse trabalho é parte de um projeto maior com o intuito de resolver o problema de localização *indoor*, tendo em vista atuações precisas de segurança em situações de emergência.

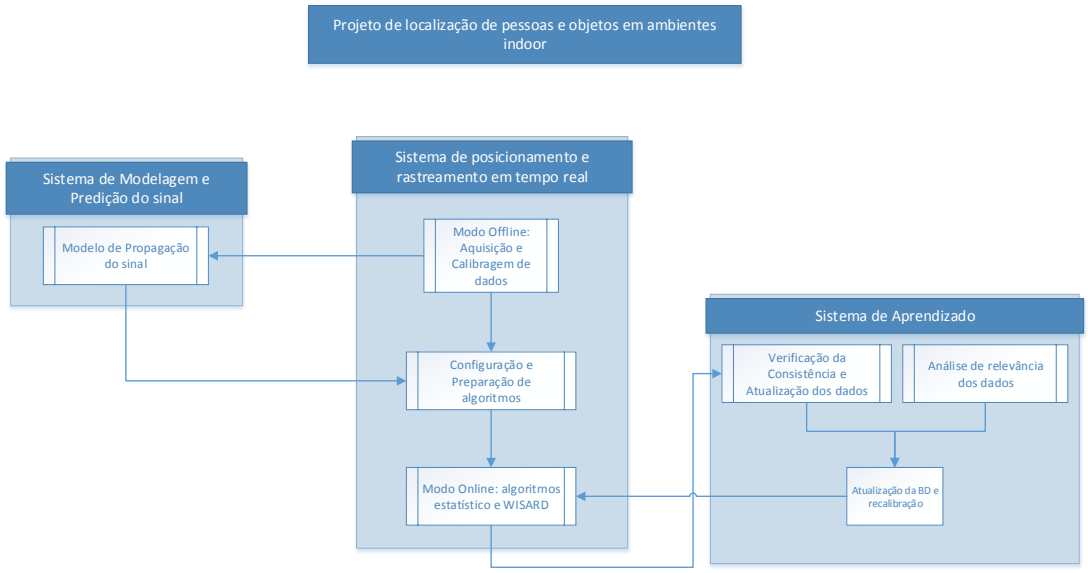


Figura 1-1 Visão geral simplificada do projeto de localização.

Existem diversas pesquisas na área, mas não há um consenso de qual seria um modelo ótimo, criando, assim, a necessidade de se determinar uma solução para os atuais pré-requisitos. Com a ideia que esse trabalho faria parte do veio principal do projeto maior (Figura 1-1), ao qual ele faz parte, foi possível traçar objetivos bem definidos.

1.2 Objetivos

Esse projeto consiste a criação de um sistema de localização capaz de funcionar utilizando apenas capacidade de um *smartphone*, tanto para coletar dados necessários para a implantação, como para a localização propriamente dita.

O sistema deve ser capaz de localizar pessoas ou objetos em um ambiente já conhecido. Para isso ele fará uso da infraestrutura de WiFi já instalada no ambiente, avaliando, pois se a qualidade para o número de pontos de acesso disponível é razoável.

Ele será um sistema de localização probabilístico, i.e. levará em consideração as variações aleatórias de potência do sinal WiFi, que ocorrem mais intensamente em ambiente *indoor*. Para determinar as probabilidades de localização, através de inferência Bayesiana, será necessário estimar determinadas distribuições de probabilidade. É nesse contexto que aparece o *KDE – Kernel Density Estiamation* – como uma forma de inferir estas distribuições.

Serão firmadas certas premissas sobre o ambiente e característica dos dispositivos e assim determinados parâmetros para o KDE. Feito isso, o sistema será implementado em um *smartphone* sempre visando uma robustez matemática e baixo custo computacional.

1.3 Estrutura do Projeto

Existem quatro grande capítulos que dividem o conteúdo desse trabalho, além do presente capítulo introdutório, estes são: Sistemas de Localização; Uma Solução de Localização *Indoor*; Resultados Experimentais; Conclusão e Trabalhos Futuros.

O primeiro trata de passar ao leitor uma visão global do que seria localização *indoor*, suas diferenças para localização *outdoor*, assim como desafios e motivações para seu uso. Ele trata também de fazer um apanhado das abordagem mais utilizadas e comentar sobre as que mais influenciaram esse trabalho.

Os detalhes da solução proposta por esse trabalho são tratadas no segundo capítulo, onde são vistas as premissas, os detalhes do algoritmo e por fim como é feita sua implementação no dispositivo móvel.

No quarto capítulo o foco é a avaliação do algoritmo no que diz respeito a precisão e robustez, para isso é descrito o método de avaliação assim como seus resultados.

Por fim, o quinto capítulo, traz um julgamento qualitativo sobre o funcionamento do algoritmo, apontando pontos fortes e possíveis formas de melhora.

2 Sistemas de Localização

2.1 Panorama

Nessa seção será passado uma visão geral sobre os sistemas de localização no mundo, tanto *indoor* como *outdoor*.

2.1.1 Outdoor vs Indoor

A razão da existência da separação entre sistemas *indoor* e sistemas *outdoor* está no centro da questão do porquê se pesquisa tanto nessa área e porque não temos nenhuma solução fechada para o dia a dia. A situação os já bem estruturados sistemas de geoposicionamento que deram a solução definitiva para os problemas de localização *oudoor*, assim com razões para o contínuo interesse em sistemas *indoor* será visto a seguir.

2.1.1.1 GPS e similares

A localização em ambientes abertos (também conhecida pelo termo em inglês *outdoor positioning system* em contraste com *indoor positioning system*) já está, atualmente, bem estabelecida. O sistema americano GPS – *Global Positioning System* – é reconhecidamente um expoente na navegação por satélite. Além do GPS outros sistemas de geoposicionamento possuem certo destaque, como o GLONASS – Globalnaya navigatsionnaya sputnikovaya sistema ou Global Navigation Satellite System – da Rússia que está operando com cobertura global desde outubro de 2011 (Russia restores its orbital GLONASS group, 2011) sendo o único além do GPS até junho de 2013.

Outros sistemas como o Galileo (Galileo navigational system enters testing stage, 2012), a ser criado pela União Europeia, e o COMPASS, pela China, também conhecido como Beidou (WEIRONG, 2010), ainda se encontram em fase de desenvolvimento em junho de 2013.

O GPS faz uso de uma rede de satélites sincronizados que enviam periodicamente sinais para o posicionamento. O sensor GPS recebe esse sinal através de frequências ortogonais, a codificação desse sinal permite ao receptor estimar a diferença

de tempo de chegada dos sinais. Além disso, a predição de posição do satélite permite que seja feita uma multilateração com três ou mais valores de diferença de tempo de chegada dos sinais e atrás disso estimar a posição do receptor GPS.

A dificuldade do uso, e consequente perda de precisão, do sistema de GPS para localização *indoor* aparece devido ao sinal enviado do satélite não chegar com força suficiente para penetrar a parede dos edifícios, o que ocorre também em certas áreas urbanas. (MIU, 2002)

2.1.1.2 Abordagem *Indoor*

Dada essa ineficiência do sistema GPS diversas soluções foram propostas, algumas dessas serão abordadas mais adiante, dentre elas destacam-se a RADAR (BAHL e PADMANABHAN, 2000) desenvolvido pela Microsoft, as tecnologias baseadas em *Zigbee* (CHEN, YANG, *et al.*, 2009) e outras que fazem uso da *UWB – Ultra Wide Band*. Diversos sistemas de localização pra celulares foram propostos nos últimos anos também com o intuito de suprir deficiências ou melhorar precisão com relação a sistemas de localização global fazendo uso das medidas de atenuação do sinal, ângulo de chegada (*AOA – Angle of Arrival*) e diferença do tempo de chegada (*TDOA – Time Difference of Arrival*). Entretanto a eficiência dessas abordagens são limitadas *indoors* pelas reflexões, obstruções e da indisponibilidade de hardwares capaz de fornecerem uma precisa sincronização temporal a baixo custo.

Grande parte do investimento nessa área veio pela necessidade de se criar aplicativos *location-aware*¸ i.e. aplicativos onde seu comportamento varia de acordo com a localização do usuário. Tais aplicativos possuem grande importância na área médica, onde podem ser usados evitar erros de protocolo, e na área de segurança, onde pode ajudar na evacuação, isolamento de áreas e na entrada de primeiros socorros. Com o barateamento dos dispositivos inteligentes portáteis, conhecidos como *smartphones*¸ o próprio mercado que visa o usuário doméstico começa a fazer pressão para soluções de localização que funcionem em ambientes fechados.

2.2 Localização Indoor

A localização *indoor* pode ser mais especificamente separada em duas abordagens, a localização propriamente dita, se referindo a um cliente estático, e o rastreio onde o cliente se move livremente. Para ambos os casos informações relevantes sobre o ambiente e sobre o sinal eletromagnético utilizado servem para a criação de sistemas de localização eficientes.

2.2.1 Características do Ambiente

As características do ambiente, no contexto desse trabalho, podem ser divididas em três: estrutural, dinâmica e eletromagnética.

2.2.1.1 Estrutural

A característica mais trivial de um ambiente é sua informação estrutural, localização das portas, paredes, janelas, móveis e equipamentos, assim como a composição de cada um desses elementos. É essa estrutura que diferencia um ambiente *indoor* de outro *outdoor* e força o desenvolvimento de outra solução de localização que seja viável mesmo sob a complexa influência dos elementos envolvidos na propagação das ondas eletromagnéticas.

Em um ambiente completamente caracterizado pela sua estrutura, dado o conhecimento detalhado dessa, seria possível determinar a distribuição de um sinal eletromagnético por toda sua extensão através de um *CEM* (*computational electromagnetics* – simulação computacional que visa resolver as equações de Maxwell fora do espaço livre, onde a geometria do mundo real muitas vezes impossibilita encontrar uma solução analítica). Entretanto para que se pudesse utilizar simulação como forma de prever a propagação de uma onda seria necessário conhecer toda a estrutura e materiais constituintes do ambiente, o que não é razoável na maioria das vezes, além da disponibilidade de grande poder computacional.

2.2.1.2 Dinâmica

O que vem a ser definido nesse trabalho como característica dinâmica de um ambiente é tudo aquilo que afeta a propagação ou características de um sinal eletromagnético e varia durante o período de uso de um sistema de localização que seja baseado em tal sinal.

Desde a reorganização dos móveis, movimentação de pessoas até o simples fato de uma porta estar aberta ou fechada muda a forma de propagação de um sinal. Alguns desses eventos de mudança são temporários como a movimentação de pessoas e o estado das portas e nos levam a uma abordagem mais inexata e probabilística, enquanto outros são na verdade o que pode ser visto como uma mudança estrutural, é o exemplo do efeito da reorganização dos móveis, o que conduz à necessidade de alguma forma de aprendizado em longo prazo.

2.2.1.3 Eletromagnética

A característica eletromagnética de um ambiente é consequência da posição, potência e propriedades da antena do emissor do sinal eletromagnético e também de outros sinais presentes no ambiente que venham a aparecer como ruído.

Contendo as importantes informações usadas por alguns sistemas de localização para determinar a posição de um objeto de interesse, como a relação sinal-ruído (SNR) e a potência do sinal.

Apesar de poder ser vista como uma característica secundária, consequência de diversos fatores inclusive das próprias características estrutural e dinâmica do ambiente, e difícil de ser inferida, ela pode ser aferida diretamente. Isso a torna uma mais apropriada para ser usada ao descrever o ambiente para um sistema de localização.

2.2.2 Características de um sinal eletromagnético

Para a construção de um sistema de localização pode-se considerar características do próprio sinal ao ser medido que, ao se distribuírem pelo ambiente, virão

a criar a estrutura eletromagnética descrita anteriormente. Dentre elas temos a relação sinal-ruído (SNR), a fase do sinal, e a potência do sinal.

A característica que será explorada nesse trabalho é a potência de sinal recebido (RSS), mais especificamente será usado o indicador de sinal recebido (RSSI) definido pelo padrão de comunicação sem fio IEEE 802.11 (CARPENTER e BARRETT, 2007), que é uma das mais utilizadas para esse fim. Duas formas de exploração dessa característica serão abordadas, a modelagem da propagação e a assinatura de potência do sinal no ambiente (*fingerprinting*).

2.2.2.1 Modelo de propagação

A modelagem da propagação do sinal leva em conta as leis físicas do eletromagnetismo para inferir como um sinal irá se comportar no ambiente e assim estimar qual será a potência de um sinal medida em um determinado ponto.

Devem ser levados em consideração diversas variáveis para se estimar o caimento de um sinal ao percorrer um ambiente, como a distância percorrida, a frequência do sinal, condições atmosféricas e o próprio tipo de terreno, montanhas, vales, lagoas, etc. (BARCLAY, 2002).

Quando se trata de ambientes indoor aparecem outros complicadores advindos da interação com os objetos do ambiente, reflexões, refrações e atenuações (MOLKDAR, 1991). Em decorrência disso aparece um fenômeno chamado *multipath* onde um sinal é recebido através de múltiplos caminhos. O sinal provindo de outro caminho chega com flutuações de fase e envelope que, ao se somar o sinal recebido pelo caminho direto com todos os demais recebidos indiretamente, contribui para produzir uma versão distorcida do sinal emitido. (BAHL e PADMANABHAN, 2000)

Segundo (BAHL e PADMANABHAN, 2000) três modelos que podem ser considerados para ambientes indoor: *Rayleigh fading model* (HASHEMI, 1993), *Rician distribution model* (RICE, 1944) e o modelo de propagação *Floor Attenuation Factor*.

O amplamente utilizado *Rayleigh fading model* (RICE, 1944) descreve as flutuações de amplitude na ausência de uma componente forte no sinal recebido. A distribuição de Rayleigh tem uma construção teórica elegante o que o faz ser amplamente

utilizado para o previsão do decaimento de um sinal submetido a múltiplos caminhos. Entretanto ele tem como premissa que todos as componentes do sinal recebido possuem a mesma potência, o que para o Por exemplo, considere o caso de uma região de relevo diversificado, em que existam montanhas muito altas, rios caso do ponto de acesso estar no mesmo ambiente que o sensor que vai aferir a potência do sinal se torna visível falso, pois existe um caminho direto (*LoS – Line-of-sight*) que possuirá uma potência maior que de suas reflexões.

Para contornar a inconveniente premissa do *Rayligh fading model*, o *Rician distribution model* (HASHEMI, 1993) pode ser usado. Ela descreve a situação de haver uma componente do sinal sensivelmente mais forte no meio de diversas outras de potência inferior. Visto dessa maneira, o Rayligh fading se torna um caso especial da distribuição de Rice. Entretanto, os parâmetros desse modelo se mostram difíceis de estimar com precisão (a potência do sinal dominante e a média local da potência das reflexões).

Em uma tentativa de achar um melhor compromisso entre simplicidade e acurácia aparece o modelo de propagação *Floor Atteenuation Factor* (SEIDEL e RAPPORT, 1992). Ele apresenta a potência do sinal dependente da distância percorrida e de um fator que é referente a atenuação causada pelo chão. Em uma versão adaptada por (BAHL e PADMANABHAN, 2000) o efeito de atenuação do chão é substituído pelo efeito das paredes presentes entre o emissor e o receptor, sendo assim mais relevante para o caso de ambos estarem no mesmo andar de uma construção, e recebe o nome de *Wall Atteenuation Factor*:

$$P(d)[dBm] = P(d_0)[dBm] - 10n\log\frac{d}{d_0} - \begin{cases} nW * WAF, & nW < C \\ C * WAF, & nW \geq C \end{cases}$$

Onde n é uma constante relacionada com a taxa de decaimento, $P(d_0)$ é a potência do sinal em um ponto d_0 tido como referência e d é a distância entre o emissor e o receptor. A constante C é o número de paredes até o qual é relevante a quantidade de paredes entre os dois pontos, nW é o número de paredes levados em consideração e WAF é o fator de atenuação relacionado a travessia do sinal por uma parede.

Trabalhos anteriores provém uma vasta informação sobre a perda de potência do sinal para diversos matérias em diversas frequência (RAPPORT, 1996). Mas devido a característica dinâmica do ambiente, onde pessoas e outros objetos que causam

interferência mudam de local com frequência, (BAHL e PADMANABHAN, 2000) sugere uma mediação empírica de sua proposta constante WAF.

Nesse projeto, entretanto, será utilizado um modelo ainda mais simples de propagação do sinal, derivado do *Wall Atteenuation Factor*:

$$P(d)[dBm] = \alpha - \beta \ln d$$

Onde d é a distância do ponto medido ao ponto de acesso, α é um parâmetro relacionado a potência original do sinal, assim como a perda de potência em todos os obstáculos até o ambiente atual, e β é uma constante que representa o decaimento do sinal pela distância percorrida.

Este modelo é mais conveniente a este caso, pois o será aplicado para fazer a extrapolação do valor da potência de um sinal em um determinado quarto, à partir de amostras de coletadas nesse mesmo ambiente. Sendo assim, não haverá variação no número de paredes, e pouca variação no total de obstáculos.

2.2.2.2 Assinatura de potência

A abordagem pela assinatura de potência, também conhecido como *signal fingerprints* (POPLETEEV, 2011), sendo um método empírico, não se apoia em nenhuma informação com relação a propagação do sinal ou interação desse com o ambiente para tentar estimar a posição. A ideia por trás da técnica é utilizar a caracterização eletromagnética do ambiente, relacionando sinais aferidos locais pré-determinados com os sinais medidos no ponto cuja localização pretende-se estimar. Ela consiste em duas etapas, uma de calibração e outra de localização.

Durante etapa de calibração, são coletados, em uma malha suficientemente densa (SECO, JIMÉNEZ, *et al.*, 2009), valores da intensidade do sinal por todo o ambiente onde o sistema de localização estará atuando. Criando, assim, uma base de dados que relaciona as posições onde foram feitas as aferições da potência do sinal com os respectivos valores de potência. Para varrer o ambiente, duas soluções são mais comuns. Pode-se coletar os dados manualmente, sobretudo em fase de teste, percorrendo o ambiente com um notebook (NAVARRO, PEUKER, *et al.*), ou outro hardware capaz de aferir a potência do sinal. Ou de uma maneira mais padronizada e escalável utilizar um dispositivo

autônomo, i.e. um robô, que faça o serviço de maneira eficiente (PALANIAPPAN, MIROWSKI, *et al.*, 2011).

Quando se implementa o sistema é iniciada etapa de localização, nesta o cliente que será localizado analisa a potência dos sinais onde ele se encontra e, através de um algoritmo apropriado, faz uma correspondência entre os dados presentes na base de dados e os valores adquiridos, tendo como saída o local mais provável de a assinatura de potência dos sinais aferidos pertencem (SECO, JIMÉNEZ, *et al.*, 2009).

Sistemas baseados em reconhecimento da assinatura de potência do sinal utiliza técnicas baseadas em aprendizado de máquina que normalmente emprega uma série de abordagens: determinístico e probabilístico, classificação e regressão (POPLETEEV, 2011). Um dos métodos mais utilizados é a estimação através do *k-nearest neighbour* (*KNN*) devido à sua simplicidade de implementação e razoável acurácia (LIN e LIN, 2005). O algoritmo *KNN* classifica as amostras adquiridas de acordo com os *k* elementos mais próximos no espaço de potência dos sinais, onde *k* é o número de vizinhos utilizados para a classificação (Figura 2-1).

Entretanto outros método mais complexos já foram explorados: Redes Neurais (BATTITI, NHAT e VILLANI, 2002; LIN e LIN, 2005), inferência Bayesiana (SECO, JIMÉNEZ, *et al.*, 2009), dentre outros (SEGATA e BLANZIERI, 2010).

O KNN aparece como um caso especial de classificação utilizando o *Adaptative Kernel Density Estimation* (*AKDE*), onde a adaptação da largura de banda[1] do *kernel* é feita proporcional à densidade de amostras, conhecido como *ballon estimators*, e utilizando um *kernel* uniforme (MILLS, 2011).

Entretanto esse trabalho se propõe a uma abordagem diferente, utilizando *Kernel Density Estimation* (*KDE*), nele será usado uma largura de banda fixa, diferenciando de sua forma adaptativa, devido redução de processamento necessário ao cliente móvel que deseja ter sua posição inferida. Esse impacto ficará claro mais à frente.

[1] Sendo $K(x)$ um dado *kernel* unidimensional tido como padrão, e $K_h(x)$ o *kernel* com largura de banda *h*, esta pode ser entendida através da relação: $K_h(x) = \frac{1}{h}K(\frac{x}{h})$

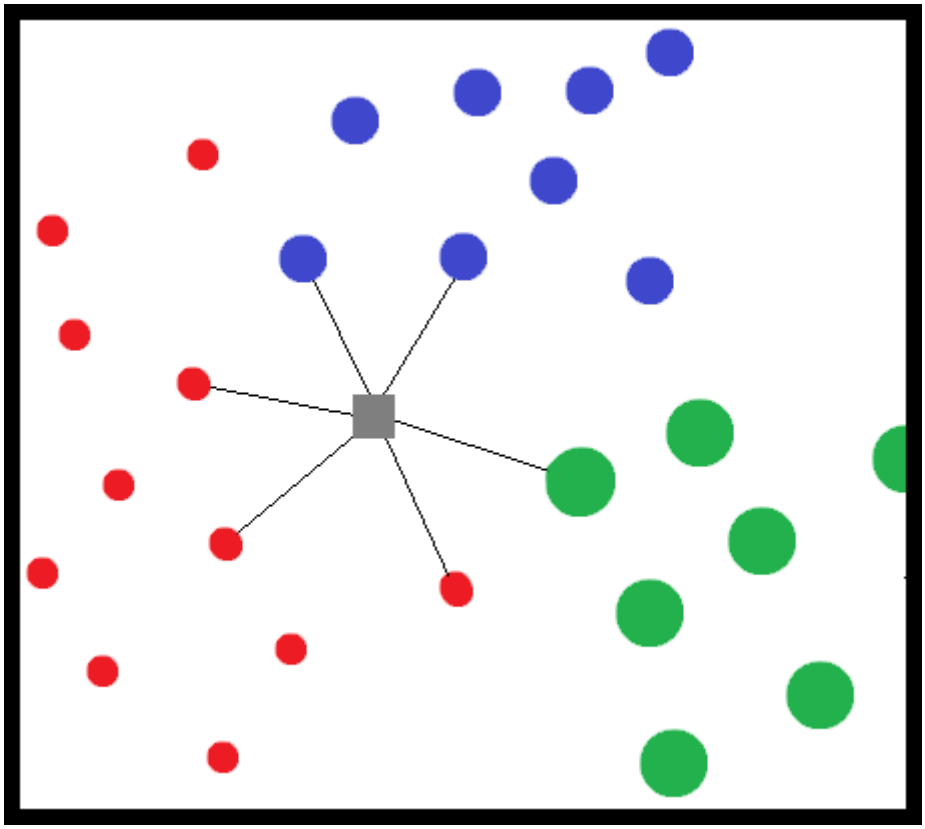


Figura 2-1 Exemplo de classificação por KNN, onde sobre a nova amostra são avaliados os cinco vizinhos mais próximos (k=5).

Um dos maiores problemas de modelos baseados no reconhecimento da assinatura de potência do sinal é o fato de estar fadado a degradação com o tempo, devido à característica dinâmica do ambiente, ou mesmo a incorrer em erros momentâneos em decorrência deste mesmo fator. Alguns fatores como a humidade, portas abertas e fechadas e a presença de pessoas já tiveram seus impactos reportados na literatura (CHEN, CHIANG, *et al.*, 2005). Outro fator identificado foi a posição relativa da antena do cliente.

Tabela 2-1 Acurácia média de posição em diferentes condições. (CHEN, CHIANG, *et al.*, 2005)

Condição padrão: sem pessoas, portas fechadas e 40% de nível de humidade relativa				
Condição do ambiente	Sem mudanças	70% de humidade	Portas abertas	Pessoas no ambiente
Efeitos	2.13 *m*	3,06 *m* (43,7%)	7,17 *m* (236,6%)	3,96 *m* (85,9%)

Para superar o problema da característica dinâmica do ambiente é necessário atualizar a base de dados de referência periodicamente. Seja utilizando-se de hardwares especiais em tempo de execução, como é o caso do emprego de *sniffers*[2] estacionários (ASTUTO, 2006) ou do uso de sensores RFID como pontos de referência, que identificam a passagem do usuário e permitem, assumindo que o usuário se mova com velocidade praticamente constate, a calibragem das posições intermediárias entre dois sensores RFID (CHEN, CHIANG, *et al.*, 2005). Ou mesmo pelo emprego de robôs que refaçam a tarefa de aferição periodicamente (PALANIAPPAN, MIROWSKI, *et al.*, 2011).

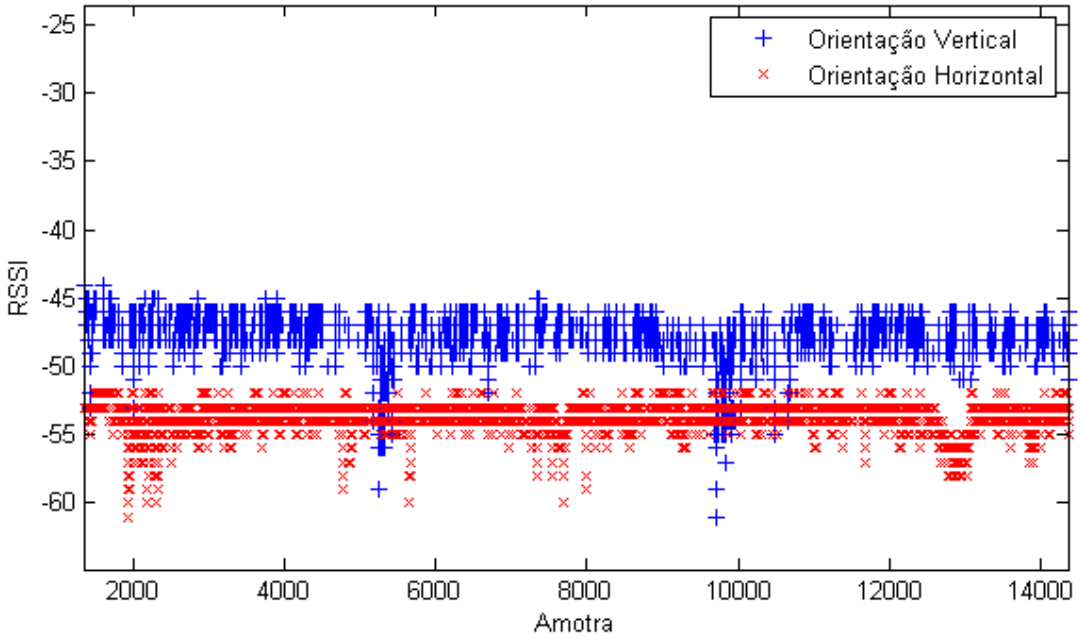


Figura 2-2 Variação dos valores de potência medidos para uma rotação de 90º do receptor ao cuso de aproximadamente 12 horas.

Recentemente está sendo desenvolvida uma proposta baseada na análise da rotina do cliente móvel, definindo pontos de interesse, e.g. próximo a cama onde se põe

[2] *Sniffers* são hardwares especializados em capturar sinais de rede.

o celular para carregar todas as noites, e os utiliza para a calibragem. Não necessitando, assim, de nenhum hardware adicional (POPLETEEV, 2011).

Outra possibilidade seria o uso dos demais sinais, não afetados por determinada alteração no ambiente, como base para calibrar os valores de potência dos sinais afetados. Entretanto, a avaliação da eficácia ou não dessa abordagem depende de pesquisa, pois não foi encontrado na literatura esse tipo tratamento.

Nesse trabalho tal problema não será endereçado, podendo ser minimizado através de quaisquer das técnicas supracitadas.

2.2.3 Localização vs Rastreio

A localização, como comentado até agora, se refere à inferência da posição de um cliente móvel em situação estática, i.e. não há variação da sua posição durante o período de estimação da posição. Já o rastreio é o intuito de se estimar a posição de um cliente móvel a todo instante.

A grande diferença entre as duas abordagens é o conhecimento das estimativas recentes das posições e/ou os valores das potências dos sinais em amostragens anteriores, cuja a informação existe para o rastreio e não na localização. Por mais que se possa memorizar as dados medidas anteriores, feitas para a localização, a falta de proximidade temporal, em geral, reduz o valor da informação como forma de melhorar a precisão da estimativa.

Por outro lado, a localização requer que o algoritmo encontre soluções rápidas, comparada a ordem de grandeza da velocidade do cliente móvel. Em situações *Indoor*, onde as velocidade são sensivelmente reduzidas, em geral restritas ao caminhar de um ser humano, esse problema não é tão considerado. Entretanto, no caso *Outdoor*, pela natureza do cliente móvel, que pode ser um veículo trafegando em alta velocidade.

A solução mais simples para o rastreio é reduzi-lo a uma série de estimativas de localização (BAHL e PADMANABHAN, 2000). Descartando assim a informação extra com relação às posições anteriores, e incorrendo nos erros devido à velocidade do cliente móvel, pois as amostras não são tomadas exatamente no mesmo local. Em testes realizados, o erro de distância mediano foi de 3,5 metros, 19% pior que o resultado para

o mesmo algoritmo trabalhando em cima de um cliente móvel em situação estacionária (BAHL e PADMANABHAN, 2000).

Com o intuito de fazer uso das estimativas anteriores, restrições de variação de posição podem ser criadas (NAVARRO, PEUKER, *et al.*). O efeito prático observado é a alteração de uma rota impossível, e.g. atravessando paredes, para uma rota viável que melhor se ajuste aos dados.

Outra abordagem é fazer uso do fato de pessoas andarem em geral em velocidade constante em uma caminhada direta entre dois pontos de interesse, podendo não só fazer uso desta informação para prever a posição de uma pessoa quando em movimento, como mesmo utilizar essa previsão para calibrar os valores de potência dos sinais nas posições intermediárias. Para isso são fixados valores máximos e mínimos de velocidade considerados normais para uma pessoa em trajeto uniforme de um ponto de interesse a outro (CHEN, CHIANG, *et al.*, 2005).

O impacto geral do rastreio em algoritmos baseados em probabilidade é o acréscimo de mais uma função de probabilidade condicional a ser definida. Ou seja, a simples regra de Bayes utilizada para estimar a posição, tendo as medidas das potências em cada posição:

$$P(A_i|S) = \frac{P(S|A_i)\,P(A_i)}{P(S)}$$

Onde A_i é o evento de estar em determinada posição *i* e S de se obter determinado conjunto de potências, por ele representado.

Acaba se tornando ligeiramente mais complexa e com probabilidades difíceis de se estimar:

$$P(A_i|L,S) = \frac{P(S|A_i)\,P(A_i|L)}{\sum_j P(S|A_j)\,P(A_j|L)}$$

Onde L é o evento do cliente móvel estar em determinada posição na última estimativa. O somatório em j ocorre sobre todas as posições possíveis. Essa fórmula decorre da premissa que S e L são condicionalmente independentes para um dado A_i qualquer.

O aparecimento da probabilidade condicional $P(A_i|L)$, sendo a probabilidade do cliente estar em determinado local dado que ele estava em outro na última amostragem, significa a necessidade de se estimar o padrão de comportamento das pessoas no ambiente. Diversos estudos foram feitos para se determinar esse tipo de padrão, em grande parte visando ajudar a atuação dos primeiros socorros (FUREY, CURRAN e MCKEVITT, 2011), mas utiliza a mesma base teórica que sua contrapartida que visa melhorar sistemas de posicionamento *indoor* (FUREY, CURRAN e MCKEVITT, 2010).

2.2.4 Dificuldades práticas

As dificuldades práticas vão além das características do ambiente, que claramente interferem na boa estimativa da posição. Seja, sua característica dinâmica, gerando grande variabilidade no valor medido de potência dos sinais, ou sua característica estrutural que, mesmo na ausência da primeira, possui uma não linearidade tal que a previsão da distribuição de potência dificilmente se daria de maneira exata.

A própria posição do sensor, no caso um *smartphone*, e consequentemente sua antena podem afetar as medições, como visto na Figura 2-2, mas levar em consideração o ganho desconhecido da antena em cada direção aumenta em muito a complexidade do algoritmo que deve rodar em um dispositivo móvel.

Outra barreira é decorrente da maneira funcionamento dos módulos *Wi-Fi* nos *smartphones*, pois esses não captam sempre os sinais advindos dos mesmos pontos de acesso. Fica a cargo do sistema decidir quando parar a busca por novos pontos de acesso disponíveis, possibilitando apenas a medida em todos os instantes da potência de uma rende à qual ele esteja conectado. Entretanto existe uma aparente relação entre o nível médio de potência medido para um dado ponto, e o número de vezes que o sinal não é

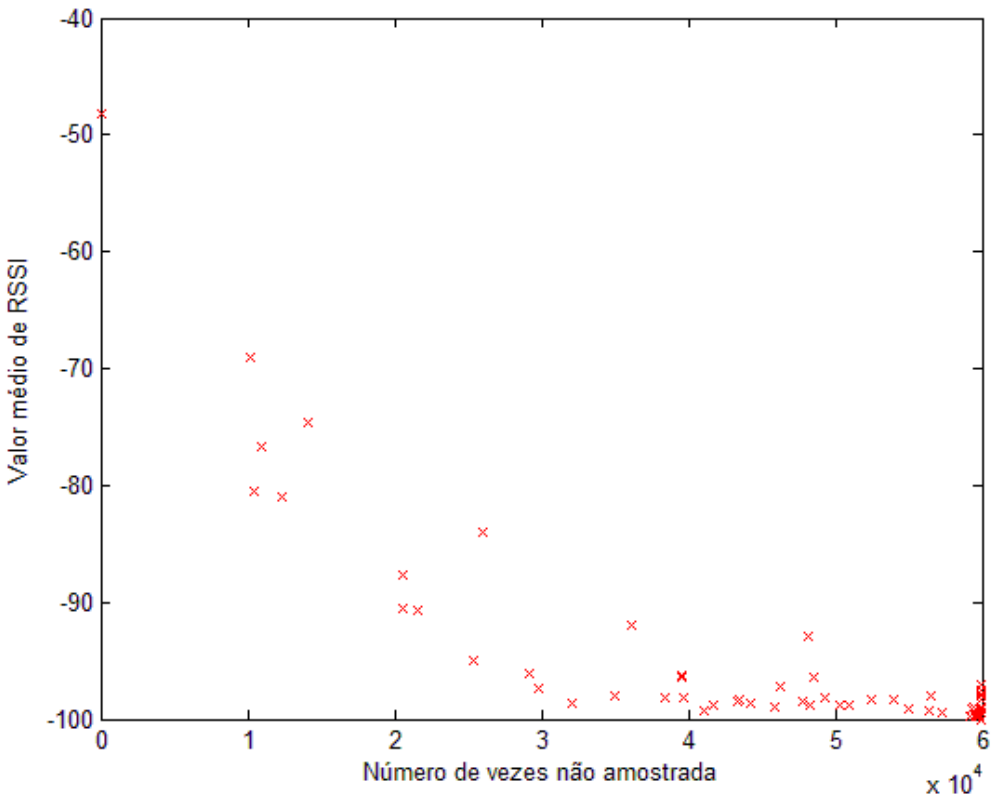


Figura 2-3 Relação entre a potência e número de vezes que o sinal não é captado, em 60 mil amostras de 60 sinais. (p-valor aprox. -0,8)

captado, ambos podendo indicar distância com relação ao cliente móvel.

Outra dificuldade das pesquisas nesse ramo é de cunho humano, hardwares dedicados para trabalhar com a emissão e recepção dos sinal, reduzindo o erro dos experimentos, tem um preço elevado, impossibilitando a compra de diversos desses equipamentos para uma pesquisa com financiamento reduzido.

2.3 Trabalhos Correlatos

No contexto da seção 2.2.2 (Características de um sinal eletromagnético), sobretudo na sua subseção 2.2.2.2 (Assinatura de potência), são descritos diversos outros trabalhos e como utilizam as características do sinal para poder prover a localização.

Nessa seção é dado destaque aos trabalhos que foram considerados de maior relevância, seja em um aspecto global, trazendo mais impacto para o desenvolvimento dos sistemas de localização *indoor* ou apresentado inovações, seja num aspecto relativo, tendo maior influência sobre o desenvolvimento desse trabalho.

2.3.1.1 RADAR

O RADAR (BAHL e PADMANABHAN, 2000) foi um dos primeiros projetos da área, dando grande estímulo às pesquisas para localização de cliente móveis *indoor*.

Antes do RADAR, as tecnologias de localização *indoor* se resumiam majoritariamente em tecnologias baseadas em sinais infravermelhos (IR) (WANT, HOPPER, *et al.*, 1992) (AZUMA, 1993), tendo problemas relacionados à escalabilidade, custos de manutenção e interferência eletromagnética do sol (BAHL e PADMANABHAN, 2000).

A técnica utilizada para localização através do algoritmo do RADAR consiste na extensão da base dados que contém as amostras do valores de potência para pontos onde essas potências não foram aferidas, através do modelo de propagação do sinal. Para tal, faz uso da *Wall Attenuation Factor* (vide 2.2.2.2) e do algoritmo *Cohen-Sutherland line-clipping* para avaliar o número de paredes entre o ponto medido e o local do ponto de acesso, tido como conhecido.

E, em fase de localização, tem-se a utilização de seu algoritmo *nearest neighbor(s) in signal space* (NNSS), que é basicamente a aplicação do KNN ao espaço das potências do sinal, junto com estimativas por triangulação.

A vantagem dessa abordagem é baixa necessidade de amostras, por outro lado possui uma precisão não muito elevada. Uma das possíveis razões é a vizinhança no espaço das potências do sinal pode não ser necessariamente equivalente a vizinhança no espaço físico.

2.3.1.2 Rádio FM

Recentemente (2011) uma nova abordagem está sendo desenvolvida, fazendo uso de redes de rádio FM, que são *broadcasted*, de ampla cobertura para localização *indoor* (POPLETEEV, 2011).

Esse método também faz uso da classificação por KNN, entretanto se baseia em características específicas das ondas de rádio FM. Mas sua contribuição mais geral vem de tentativa de calibração espontânea.

A ideia por trás da calibração espontânea é o reconhecimento de locais onde a posição do cliente móvel pode ser inferida pelo contexto, e não apenas pelo sistema de localização. A aplicação desse conceito reduziu a necessidade de calibração constante dos valores da base dados do sistema.

2.3.1.3 Estimação probabilística

A ideia de aplicar probabilidade para tentar achar o local com maior chance do cliente móvel estar teve grande influência sobre este projeto.

O conceito reside em utilizar as amostras, e consequente probabilidade estimada, do conjunto de potências dos sinais para cada local para se obter a probabilidade de um determinado local ter certo conjunto de potências dos sinais aferidos em tempo de execução (YOUSSEF, AGRAWALA e SHANKAR, 2013). Aparece matematicamente como uma aplicação direta do Teorema de Bayes[3].

Além disso o trabalho explorou a ideia de *clusterização* das posições de maneira a reduzir consideravelmente o esforço computacional necessário para a estimação, em comparação com trabalhos que não fazem uso como o RADAR (BAHL e PADMANABHAN, 2000).

[3] $P(A|B) = \frac{P(A|B)P(A)}{P(B)}$, onde A e B são eventos quaisquer.

3 Um Solução de Localização *Indoor*

Nesse capítulo trataremos da solução de localização proposta pro esse trabalho, passando pelas premissas, o desenvolvimento do algoritmo e, depois, sua implementação.

3.1 Premissas do Projeto

Devido à grande complexidade de soluções e problemas práticos, algumas premissas foram tomadas. Dentre essas premissas existem aquelas que podem ser consideradas como meros casos especiais de um sistema de localização mais geral, entretanto também foram feitas premissas que provavelmente afetam o desempenho do sistema, visando a redução da complexidade, que se fossem removidas e tratadas em toda sua riqueza de detalhes provavelmente traria melhores resultados ao algoritmo.

3.1.1 Antena

A posição da antena do dispositivo móvel (como visto nas seções 2.2.4 e 2.2.2.2) afeta o valor de potência aferido.

Isso é causado pela antena dos *smartphones* não ser uma antena isotrópica ideal, mas ser, geralmente, uma antena omnidirecional (vide Figura 3-1). Assim, o ganho proporcionado pela antena varia com a orientação do aparelho, e consequentemente sua antena, gerando assim direções preferenciais.

Soma-se a isto o fato das antenas normalmente não responderem igualmente a qualquer polarização. O sinal que era possivelmente na fonte uma onda eletromagnética circularmente polarizada, após atravessar meios diferentes, sofrendo reflexões, refrações e espalhamentos, acaba por chegar ao receptor com sua polarização alterada, tendo como consequência ganhos diferentes para caminhos percorridos diferentes.

Devido a este senário complexo, o trabalho aqui descrito terá como premissa um comportamento isotrópico para a antena do cliente móvel. O que é certamente irreal, mas dado que o aparelho capte sinais de vários ponto de acessos diferentes e tenha uma boa

orientação com relação a alguns deles, a variação no ganho poderá aparecer como um ruído no recebimento dos outros sinais.

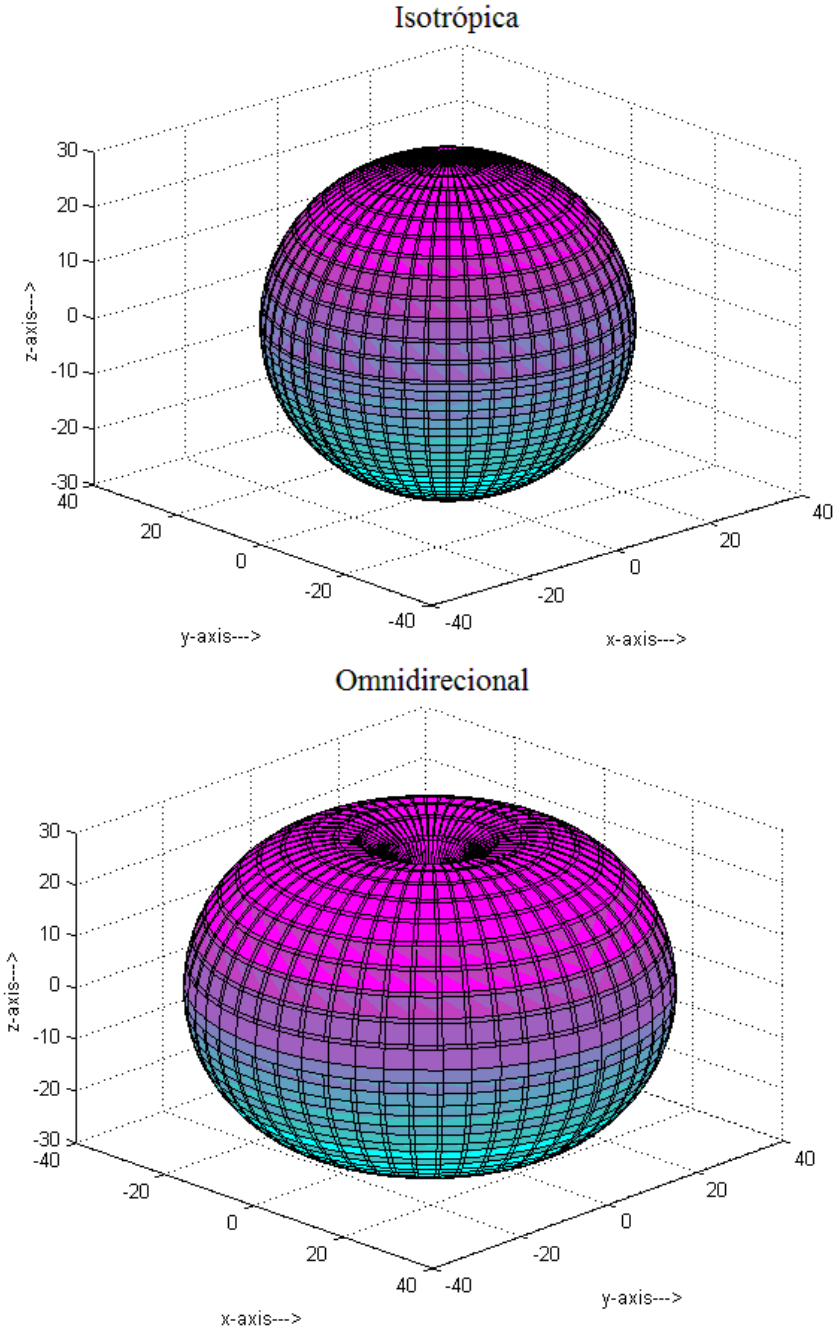


Figura 3-1 Comparação entre o os ganhos de uma antena isotrópica ideal e uma omnidirecinal, posicionadas verticalmente no eixo Z.

3.1.2 Receptor estático

Como visto na seção 2.2.3 existem diferenças importantes entre localização e rastreio. O foco desse projeto se restringe ao caso de localização, para isso tem-se como premissa que o cliente móvel é estático, assumindo-se assim que todas as amostras feitas dos valores de RSSI dos sinais são tomadas à partir de um mesmo ponto.

3.1.3 Desconhecimento de redes vizinhas

Diversos trabalhos, como o RADAR em 2.3.1.1, levam em consideração o conhecimento exato da posição dos pontos de acesso, de maneira que ficam restritos aos sinais provindos de pontos de acessos que possuem e consequentemente tem a posição controlada.

Em ambientes *indoor* comuns, e.g. casas, escritórios, shopping centers, a grande maioria dos pontos de acesso com sinal disponível não são de posse de quem implanta o sistema de localização (Figura 3-2). A independência desse fator possibilita uma maior flexibilidade com relação ao local de implantação do sistema, redução no custo de instalação e no tempo de implantação.

Sendo assim, neste trabalho não será levado em consideração o conhecimento exato da posição dos pontos de acesso, ou seja, terá como premissa o desconhecimento completo das redes captadas.

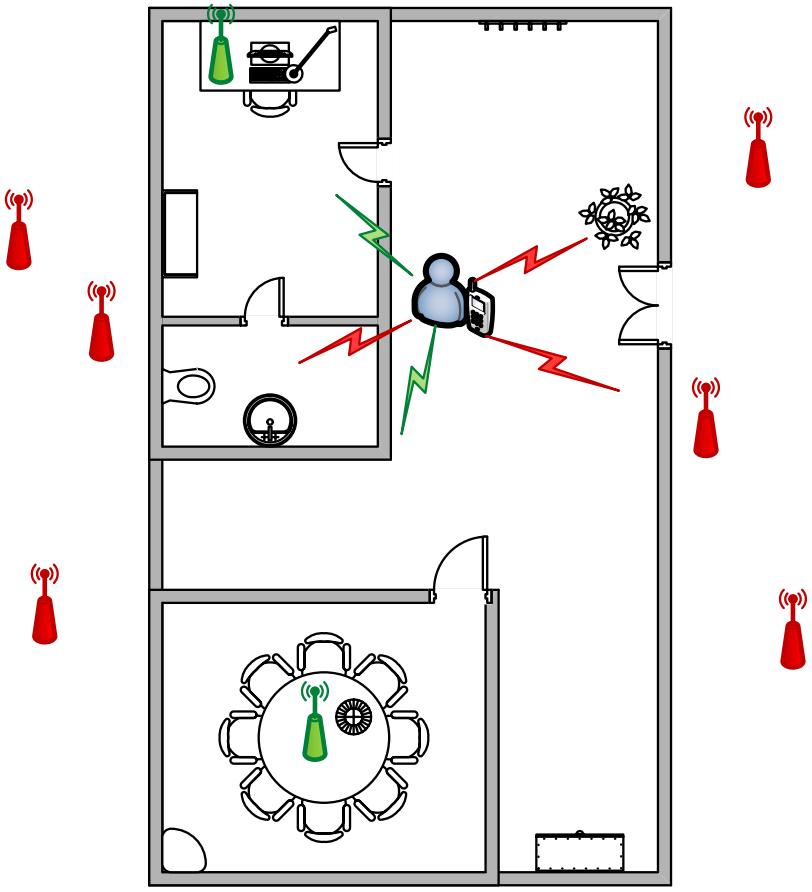


Figura 3-2 Maior parte dos sinais recebidos em um ambiente comum não pertencem ao proprietário do sistema de localização.

3.1.4 Não variabilidade do ambiente

Como discutido na seção 2.2.1.2, uma das características importantes do ambiente é a dinâmica. Importante, pois ela influência não só ruídos momentâneos como alterações na própria estrutura, ou seja, que perduraram por um longo período.

A abordagem feita neste projeto não levará em consideração qualquer dinâmica, seria necessário algum tipo de heurística (como visto em 2.3.1.2) ou aprendizado por contexto. Entretanto essa premissa é feita por outros projetos na área, onde o impacto é a necessidade de uma calibração frequente do sistema para atualizar a base de dados de acordo com as alteração ocorridas no período.

3.1.5 Portabilidade

O projeto objetiva ter como resultado um sistema completamente autônomo, onde o cliente possa instalar o aplicativo e utiliza-lo livremente sem a necessidade de qualquer estrutura subjacente.

Para prover essa portabilidade foi escolhido como plataforma qualquer *smartphone* com sistema operacional *Android*® (versão 2.3 ou superior). É uma plataforma bem difundida com uma extensa literatura sobre suas *API*'s, permitindo um futuro reuso dos códigos em outros projetos, ou complementação do mesmo, sem maiores esforços.

Entretanto numa fase de testes e análise de resultados será utilizado um servidor em *Matlab*® para comunicação com o *smartphone*, no caso um *XT316 Motorola*®.

3.2 Algoritmo de Localização

O algoritmo se divide em duas grande etapas, um treinamento *off-line* e a posterior computação *online* das probabilidades de se estar em cada posição utilizando os valores de RSSI medidos. Entre elas existe uma etapa menor de preparação dos dados para uma computação mais eficiente das probabilidades quando rodando em modo *stand-alone*[4] no *smartphone* (Figura 3-3).

[4] Onde o aplicativo computa todas as probabilidade sem se comunicar com o computador.

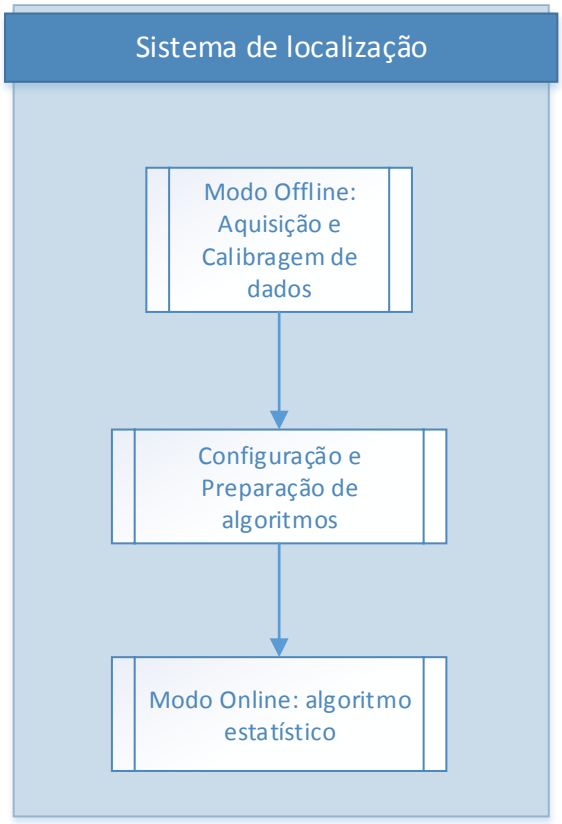


Figura 3-3 Fluxograma do Sistema de Localização

3.2.1 Treinamento

Durante a fase de treinamento coleta-se manualmente com o *smarthphone* valores de RSSI dos sinais disponíveis em posições pré-determinadas. São feitas 20 amostras por posição e todas são em sequência armazenadas numa base de dados em SQLite[5] que possui suporte nativo do *Android*®. A base de dados é composta de 4 tabelas: Local, Access_Point, KSD, e Sample. A declaração de todas as tabelas se encontra no anexo.

[5] Um tipo menos robusto de sistema de gerenciamento de banco de dados que em sua origem não garante integridade do domínio, para mais informações favor consultar: http://www.sqlite.org/.

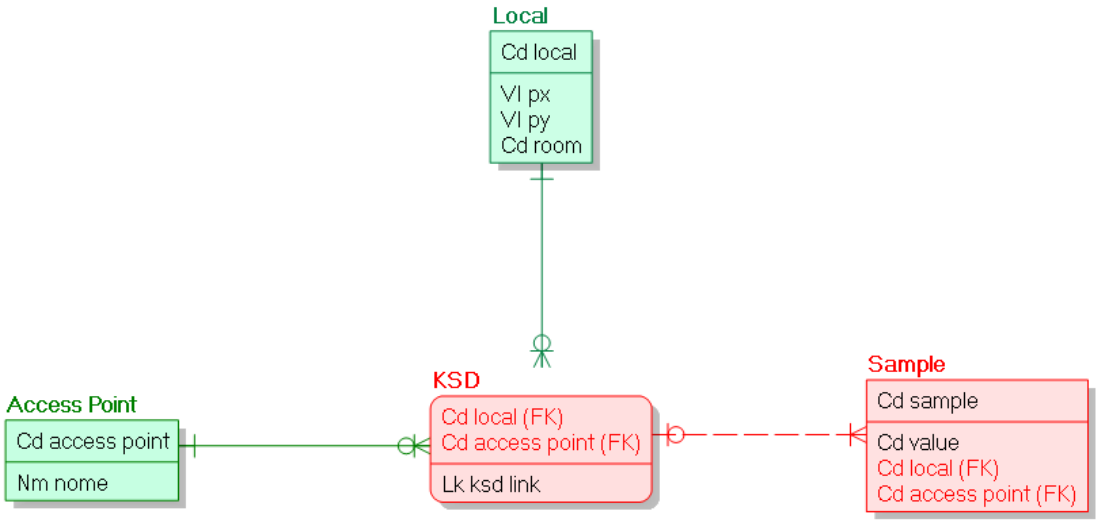


Figura 3-4 Modelo lógico da base de dados, que irá conter as informações sobre os valores de RSSI amostrados.

3.2.1.1 Local

Composta de 4 campos, uma chave primária e uma restrição de unicidade, essa tabela tem o objetivo de armazenar os locais, i.e. posições X e Y na planta 2D e o quarto, identificado por um número natural, onde essa posição foi amostrada.

Sua chave primária (Cd_local) é um identificador único autoincrementável utilizado apenas como referência interna do banco de dados. Os outros três campos são valores inteiros referentes às posições X e Y (Vi_px e Vi_py, respectivamente) e o número do quarto (Cd_room).

Sua restrição é o do tipo “UNIQUE” e aplicada ao par posição X e Y. Essa restrição impede, retornando um erro, a inserção de outra entrada que possua os mesmos

Cd_local	Vi_px	Vi_py	Cd_room
1	50	63	1
2	287	94	2
3	315	292	3
4	168	212	4

Figura 3-5 Tabela local após quatro posições amostradas em cômodos diferentes.

valores de posição X e Y. Assim evita entradas múltiplas na base dados para um mesmo local.

3.2.1.2 Access_Point

A tabela Access_Point é utilizada para armazenas os valores de BSSID[6], i.e. os identificadores únicos de cada ponto de acesso.

Composta por apenas dois campos, um identificador único (Cd_access_point) autoincrementável como chave primária e outro de texto para armazenar o os BSSID propriamente ditos (Nm_name), este último é submetido a uma restrição do tipo "UNIQUE" para evitar que sejam armazenadas múltiplas entradas do mesmo ponto de acesso.

Cd_access_point	Nm_name
21	80:3f:5d:98:14:b0
22	20:10:7a:ea:87:6d
23	90:f6:52:d9:2e:16
24	00:1d:d5:09:29:b0
25	00:1d:d5:cf:e4:10
26	00:25:9c:35:8d:c7
27	20:10:7a:df:e5:c3
28	14:5b:d1:48:3a:29
29	b0:48:7a:c3:d6:b2
30	6c:2e:85:ec:01:3b

Figura 3-6 Tabela Access_Point exibindo alguns dos BSSID amostrados.

[6] basic service set identification (BSSID) numa infraestrutura BSS (basic service set), como é o caso de uma LAN em funcionamento não ad hoc, é o endereço MAC (media access control address) desse ponto de acesso.

3.2.1.3 KSD

A tabela KSD serve para associar cada ponto de acesso e cada local a uma função de distribuição de probabilidade dos valores de RSSI medidos que será utilizado para calcular a probabilidade de estar em cada local quando em modo *online*.

Composta por três campos, o par identificador do ponto de acesso (Cd_access_point) e identificador do local (Cd_local), que serve como chave primária para a tabela e é ao mesmo tempo chave estrangeira vinda das tabelas Local e Access_Point, e o campo de texto que irá conter o nome do arquivo interno do aparelho onde estará armazenada a função de distribuição de probabilidade dos valores de RSSI. Este último só é preenchido em uma etapa intermediária, sendo deixado em branco durante a etapa de coleta de dados.

Cd_access_point	Cd_local	Nm_ksd_link
1	1	ks1
2	1	ks2
3	1	ks3
4	1	ks4
5	1	ks5
6	1	ks6
7	1	ks7
8	1	ks8
9	1	ks9
10	1	ks10
1	2	ks11
4	2	ks12
2	2	ks13
8	2	ks14
11	2	ks15
7	2	ks16
9	2	ks17
12	2	ks18

Figura 3-7 Tabela KSD após a etapa intermediária de preparadação dos dados, já contendo o nome dos arquivos com as distribuições de probabilidade.

3.2.1.4 Sample

A tabela Sample pode-se dizer que é a mais importante, armazenado todos os valores de RSSI medidos e relacionando-os com o local e o ponto de acesso aos quais pertence.

São 4 campos, uma chave primária autoincrementável (Cd_sample), um campo para os valores medidos de RSSI (Vi_value), e outros dois campos com as identificações de local e ponto de acesso (Cd_local e Cd_access_point, respectivamente) que aparecem com chaves estrangeiras da tabela KSD.

Cd_sample	Vi_value	Cd_local	Cd_access_point
112	-88	1	5
113	-90	1	6
114	-83	1	4
115	-91	1	8
116	-89	1	3
117	-37	1	1
118	-36	1	1
119	-81	1	2
120	-86	1	5
121	-92	1	6
122	-83	1	4
123	-37	1	1
124	-56	2	1
125	-86	2	4
126	-80	2	2
127	-87	2	8
128	-88	2	11
129	-89	2	7
130	-90	2	9

Figura 3-8 Tabela Sample contendo dados de diversos locais e pontos de acesso.

3.2.2 KDE – *Kernel Density Estimation*

3.2.2.1 Probabilidade condicionada à posição

O projeto se baseia na estimação das probabilidades de se estar em um determinado local dado os valores de potência aferidos nele ($P(A_i|S)$). Recordado o teorema de Bayes (visto em 2.2.3):

$$P(A_i|S) = \frac{P(S|A_i)\,P(A_i)}{P(S)}$$

Onde A_i é o evento de estar em determinada posição i e S o evento de se obter determinado conjunto de potências.

Assim dividimos o trabalho de computar $P(A_i|S)$ em três, sendo necessário computar separadamente os valores de $P(A_i)$, $P(S)$ e $P(S|A_i)$.

Partimos do pressuposto que todos os lugares são igualmente prováveis de se estar, uma suposição diferente iria requerer um estudo do comportamento das pessoas no ambiente, e considerando uma probabilidade não condicional $P(A_i)$ é tido como sendo uniformemente distribuído sobre todas as posições A_i.

O objetivo final não é se obter o valor exato da probabilidade $P(S|A_i)$ para cada local, e sim conseguir definir para qual posição seu valor é máximo. Sendo assim, podemos ignorar e valor de $P(S)$, dado que ele não varia em função de A_i. Obtendo-se no final um valor para comparação, que poderia ser visto como um valor de probabilidade não normalizado visto que não terá um valor de soma total unitário.

Sendo assim o cálculo de $P(A_i|S)$ recai substancialmente no cálculo de $P(S|A_i)$. Para o cálculo desse último leva-se em consideração a independência dos valores de RSSI referentes a pontos de acesso diferentes. Ou seja, temos:

$$P(S|A_i) = P(S_1, S_2, \dots, S_m|A_i)$$

Onde m é o número de pontos de acesso medidos e $S_1, S_2, \dots, S_m$ os valores de RSSI medido para cada um. Dada a independência condicional para todos os lugares, podemos escrever:

$$P(S_1, S_2, \dots, S_m|A_i) = P(S_1|A_i)P(S_2|A_i) \dots P(S_m|A_i)$$

Em resumo, o que será necessário estimar é o valor de $P(S_i|A_j)$, i.e. o valor de se medir determinado valor de RSSI vindo de um *j*-ésimo ponto de acesso em uma *i*-ésima posição.

3.2.2.2 Kernels

Como visto na seção anterior, temos a necessidade de se estimar o valor de $P(S_i|A_j)$, para isso será feito uso do *KDE*.

O *KDE* é um método não-paramétrico para se estimar a *pdf* (função de densidade de probabilidade) de uma variável aleatória, i.e. sem que se saiba de antemão o formato desta, um exemplo conhecido de método não-paramétrico são histogramas.

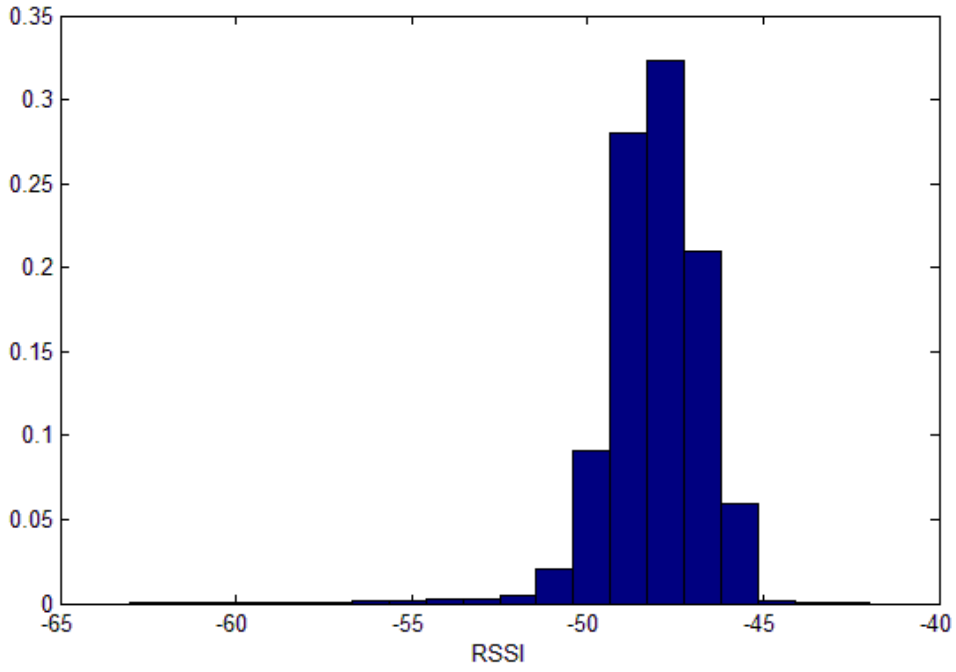


Figura 3-9 Exemplo de histograma dos valores de RSSI em um local, h=1

Introduzido por Rosenblatt (1956) e Parzen (1962), pode ser visto como uma engenhosa extensão da estimação por histograma, tentando suprir deficiências presentes neste, e.g. a falta de continuidade.

Uma maneira de pensar sobre histogramas é pela seguinte definição:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} w(x - x_i, h)$$

Onde $x_1, x_2, \ldots, x_n$ são os valores amostrados e:

$$w(x,h) = \begin{cases} \frac{1}{2h}, & |x| < h \\ 0, & c.c. \end{cases}$$

Pode se pensar, então, que para cada amostra é somado um retângulo de largura $2h$ e altura $1/2h$. E a estimativa da *pdf* num determinado ponto é o valor dessa soma normalizado por 1/n.

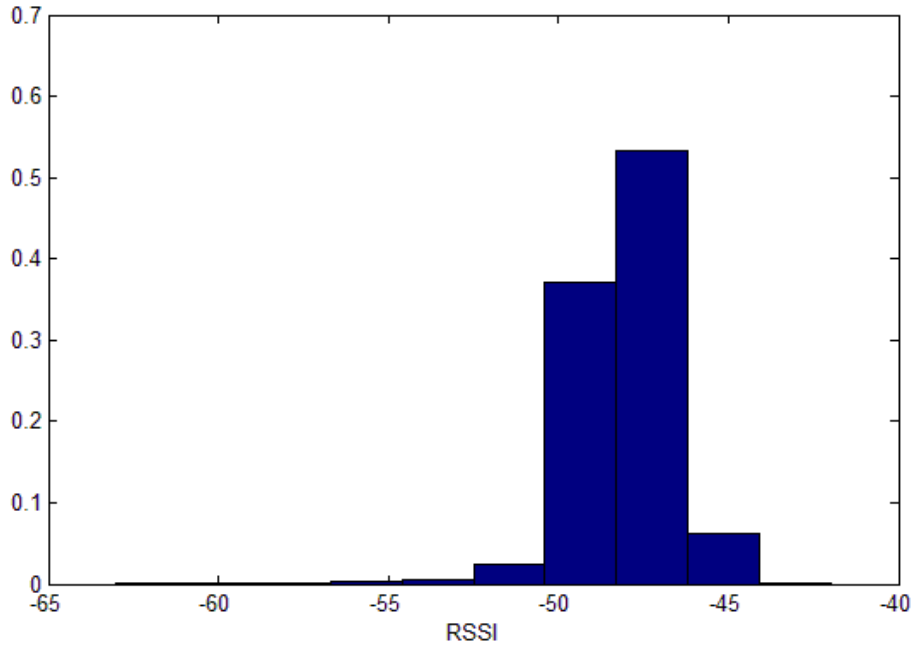


Figura 3-10 Exemplo de histograma dos valores de RSSI em um local, h=1

Pelos exemplos da Figura 3-9 e Figura 3-10, fica evidente o impacto do valor de *h*. Para valores de *h* maiores os retângulos ficam mais largos, suavizando a distribuição.

No geral, no lugar de $w(x,h)$ pode-se ter uma função de ponderação $K_h(x)$ do tipo:

$$K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right)$$

Sendo assim:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x}{h}\right)$$

A função $K(x)$ é chamado de *kernel* (núcleo). O *kernel* é uma função de ponderação padronizada, ou seja, é a função de ponderação para *h*=1. O parâmetro *h* é chamado *bandwidth*, ou largura de banda, ela determina a quantidade de suavização utilizada para estimar a *pdf*. (ZUCCHINI, 2003)

Três são as condições para uma função poder ser considerada um *kernel*:

(1)

$$\int K(x)dx = 1$$

(2)

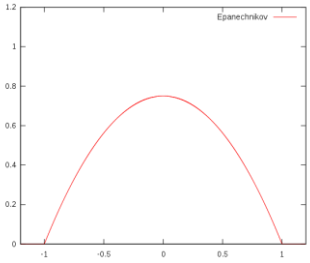
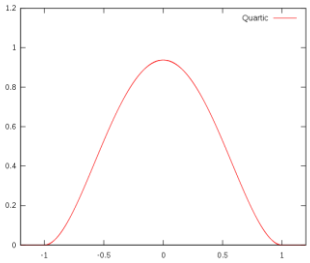
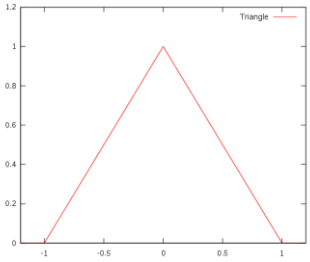
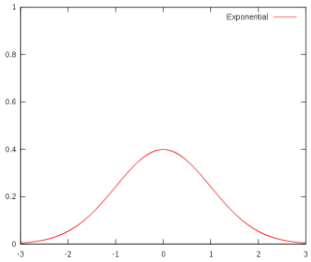
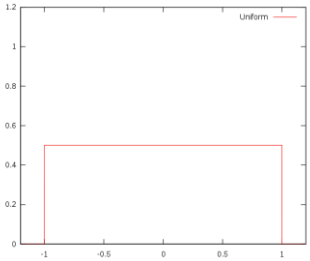
$$\int xK(x)dx = 0$$

(3)

$$\int x^2K(x)dx = k < \infty$$

Temos assim que qualquer *pdf* simétrica pode ser usada como um *kernel*.

Tabela 3-1 Exemplo dos principais *kernels* com sua eficiência medida em termos de valor esperado do erro, em relação ao *kernel* ótimo. $1_{\{\dots\}}$ é a função indicadora.

Kernel	*K(u)*	Formato	Eficiência
Epanechnikov	$\frac{3}{4}(1-u^2)1_{\{\lvert u\rvert\leq 1\}}$		**1**
Biweight	$\frac{15}{16}(1-u^2)^2 1_{\{\lvert u\rvert\leq 1\}}$		**0,9939**
Triangular	$(1-\lvert u\rvert)1_{\{\lvert u\rvert\leq 1\}}$		**0,9859**
Gaussiana	$\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}u^2}$		**0,9512**
Uniforme	$\frac{1}{2}1_{\{\lvert u\rvert\leq 1\}}$		**0,9295**

O *kernel* ótimo *Epanechnikov* (vide Tabela 3-1) é ótimo no sentido de erro médio quadrático (MÜLLER, 1984). Entretanto ele tem uma suavidade[7] μ igual a 2 em comparação com a Gaussiana que possui suavidade infinita. Uma maior suavidade reduz

[7] O grau de suavidade é medido pelo número de derivadas que a função possui.

o impacto do ruído, intuitivamente uma função mais suave reduz variações bruscas na *pdf* (WAND e SCHUCANY, 1989). É possível encontrar outros *kernels* ótimo com maior graus de suavidade, entretanto todos esses são polinómios no intervalo [-1,1] (MÜLLER, 1984).

Nesse trabalho será utilizado o *kernel* Gaussiano, além de ser altamente suave, esta possui inúmeras propriedade computacionais não presente em polinómios como facilidade na computação de convolução ou transformada de Fourier. Fatos que levam a degradação na performance de computar bons valores de *bandwidth* através de técnicas avançadas. Por outro lado, como exposto na Tabela 3-1, a escolha do *kernel* tem um impacto minoritário na eficiência da estimativa. (WAND e SCHUCANY, 1989)

A taxa de convergência do erro quadrático é uma função da ordem, número de momentos[8] não-nulos, do *kernel*, i.e. *kernels* de ordem mais alta possuem taxas de convergência mais alta. É possível estender o *kernel* Gaussiano para ordens maiores (WAND e SCHUCANY, 1989), entretanto como o número de amostras na fase *off-line* não é um restritivo o aumento do custo computacional pode não ser justificado.

Tabela 3-2 Exemplo de extensões da função Gaussiana.

Ordem	Função	Eficiência
2	$\phi(x)$	0,9512
4	$\frac{1}{2}(3-x^2)\phi(x)$	0,9320
6	$\frac{1}{8}(15-10x^2+x^4)\phi(x)$	0,9200

A largura de banda, por outro lado, possui uma maior influência em um bom ajuste da *pdf* estimada. Existem diversos métodos para se tentar achar um valor para *h*, e.g. *Cross-Validation*, *"Plug-in" Estimator*. A desvantagem desses métodos é que em geral dependem da otimização de alguma função custo ou mesmo da estimação de derivadas da *pdf* de ordem maior (ZUCCHINI, 2003). Em um compromisso entre performance e otimalidade, pode-se utilizar um estimador mais simples, qual que minimizaria o erro médio quadrático para um *kernel* Gaussiano dado que a *pdf* subjacente seja uma distribuição normal (TERRELL e SCOTT, 1992). Isso pode gerar erros por um sobre-suavizamento da estimativa em casos onde a *pdf* subjacente seja multimodal, como

[8] O *n*-ésimo momento de uma variável aleatória com *pdf* $K(x)$ é $E[X^n] = \int_{-\infty}^{\infty} x^n K(x)dx$

pode ocorrer no caso de um *multipath* que ocasione um segundo caminho preferencial. Para isso a *bandwith* deve ser dada por:

$$h_{opt} = \left(\frac{4}{3n}\right)^{1/5} \sigma$$

3.2.2.3 Casos especiais

Um complicador aparece quando a variável aleatória que se deseja estimar tem indubitavelmente suporte compacto, i.e. está restrita a determinado intervalo, que é exatamente o caso de valores de RSSI. O uso do método de estimação por *kernels* não garante isso, de fato ele tende a extrapolar esses limites, sobretudo com o uso de um *kernel* com suporte ilimitado como é o caso do Gaussiano.

Uma possível solução é aplicar uma transformação sobre as amostras de maneira a tornar o suporte ilimitado, fazer a estimação usando um *kernel* e depois computar a *pdf* original à partir da transformada (ZUCCHINI, 2003). Ou seja, dada uma variável aleatória *S*, com *pdf* $f(x)$, cria-se outra variável aleatória $T = t(S)$, sendo $t(\cdot)$ uma função monótona crescente, com *pdf* dada por $g(x)$. Então, tem-se que a relação entre f e g é dada por:

$$f(x) = g\big(t(x)\big)t'(x)$$

Um exemplo de função $t(\cdot)$ utilizada para estimar uma *pdf* com suporte compacto entre a e b é $t(x) = \ln\left(\frac{x-a}{b-x}\right)$, assim $t'(x) = \frac{1}{x-a} + \frac{1}{b-x}$, logo:

$$\hat{f}(x) = \hat{g}\left(\ln\left(\frac{x-a}{b-x}\right)\right)\left(\frac{1}{x-a} + \frac{1}{b-x}\right)$$

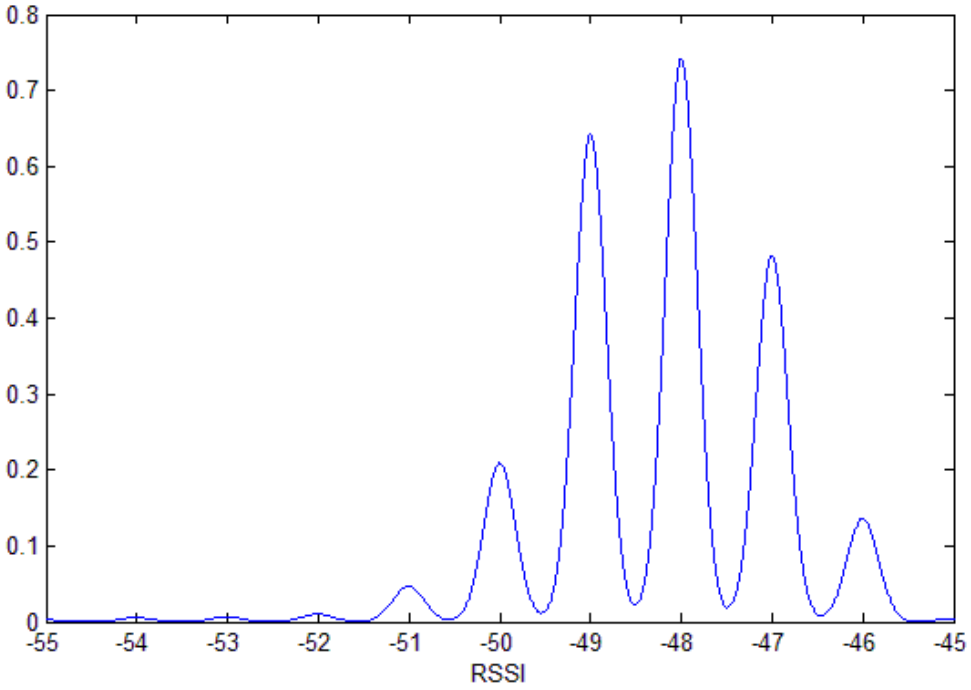


Figura 3-11 Estimação por *kernel* sem levar em consideração o suporte da distribuição subjacente.

Ao se utilizar a estimação da função transformada o que acontece, na realidade, é a aplicação de um *kernel* diferente a cada ponto da *pdf* original, sendo sua variação maior quanto mais próximo se estiver da fronteira do suporte.

Entretanto esse método é computacionalmente custoso e existe outro método mais simples e aplicável. É possível truncar o valor da estimação gerada sem levar em consideração a extensão do suporte, e renormaliza-lo (JANN, 2007).

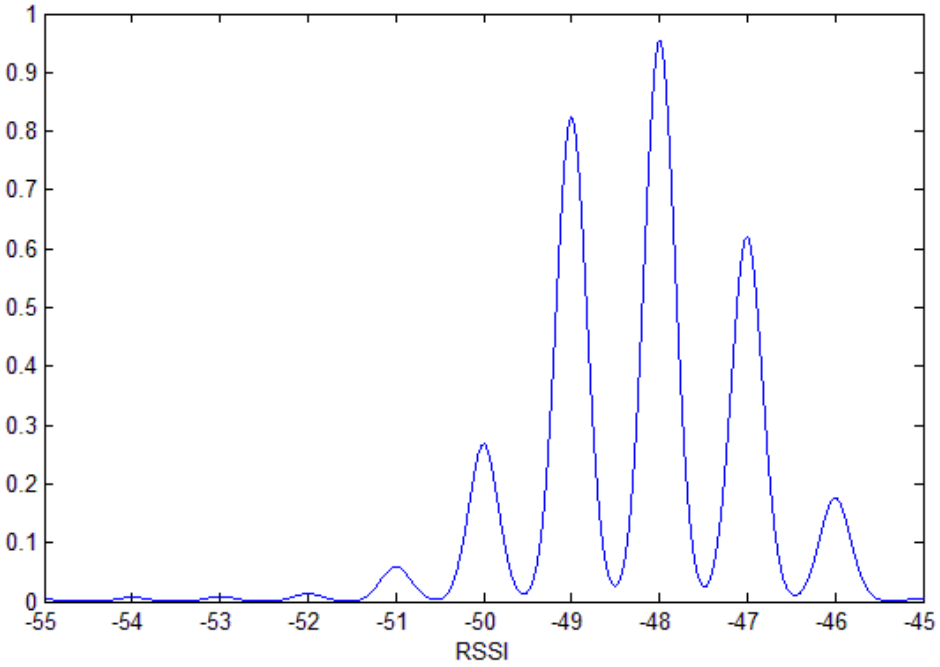


Figura 3-13 Estimação utilizando transformação para um suporte igual à (-35,-100).

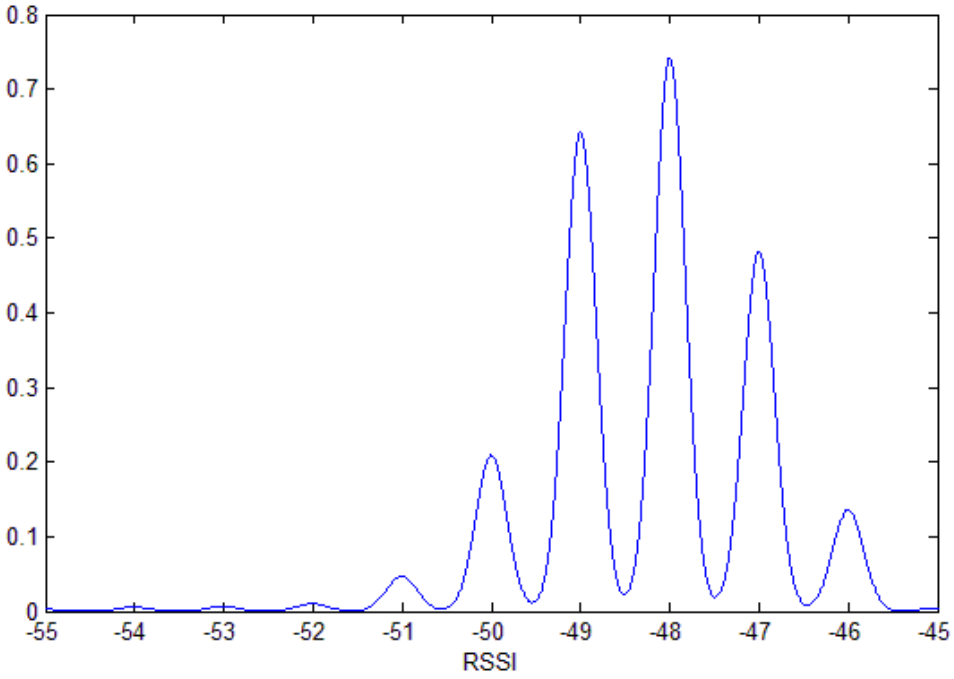


Figura 3-12 Estimação após truncamento e renormalização sobre o suporte (-35,-100).

Existe outra diferença entre o método de estimação por *kernel* descrito até agora e o que é necessário à aplicação. Os valores de RSSI são discretos e a estimação como feita até o memomento levava em consideração um domínio contínuo. Fato pelo qual os gráficos de estimativas parecem ser uma série de "picos" e "vales".

Mas de fato essa diferença entre contínuo e discreto não acaretará em maiores consequências, visto que, por mais que os valores ótimos sejam diferentes, *kernels* contínuos amostrados e dicretamente normalizados serbem como *kernels* discretos (RAJAGOPALAN e LALL, 1995). Ou seja, dado um *kernel* contínuo $K(x)$, pode-se utilizar como *kernel* discreto:

$$K_d(x) = \frac{K(x)}{\sum_{-\infty}^{\infty} K(x)}$$

Na prática, em vez de se normalizar o *kernel*, faz-se o somatório dos diversos *kernels* e posteriormente uma única normalização (não se divide pelo número de amostras).

3.2.3 Comparação de probabilidades

Como visto em 3.2.2.1, não é necessário encontrar um valor exato de $P(A_i, S)$, podendo utilizar apenas um fator de comparação $H_S(A_i)$ para um determinado conjunto de potências *S*, proporcional a $P(A_i, S)$ para um mesmo *S*. À princípio pode-se pensar então em:

Equação 3-1

$$H_S(A_i) = P(S_1|A_i)P(S_2|A_i) \dots P(S_m|A_i)$$

Onde *m* é o número de pontos de acesso medidos e $S_1, S_2, \dots, S_m$ os valores de RSSI medido em cada um destes, e:

Equação 3-2

$$P(S_j, A_i) = c \sum_{k=1}^{n} \phi_h(S_j - {}_i^k B_j)$$

Sendo ϕ_h um *kernel* Gaussiano com *bandwidth h*, ${}_i^k B_j$ o *k*-ésimo valor de RSSI amostrado no *i*-ésimo local para o *j*-ésimo ponto de acesso, e *c* um fator normalizante e *n* o número de amostras feitas em cada local.

3.3 Programação *Android*®

Como comentado em 3.1.5, o sistema foi desenvolvido para a plataforma *Android*®. Esta possui vasto suporte, contendo exemplos, documentação e API's. Em http://developer.android.com/ é possível encontrar todo o material que é preciso para se começar a programar.

A *IDE* (Ambiente Integrado de Desenvolvimento) utilizada foi o *eclipse*® para *Windows*® e escrito em Java com a ajuda do *ADT*(*Android Development Tools*) que é um *plug-in* para o eclipse, que junto com o pacote *Android SDK Tools*, forma o *kit* essencial para o desenvolvimento de aplicativos *Android*®.

O intuito dessa seção é comentar alguns detalhes técnicos da implantação computacional, na plataforma alvo, algumas otimizações necessárias para evitar alto custo computacional e erros numéricos.

3.3.1 Java e uso de API's

A forma mais comum de se fazer programas para *Android*® é em JAVA. O projeto foi idealizado visando a versão *Gingerbread* (*Android 2.3.3*) e, dessa maneira, todas as API's utilizadas, incluindo WiFi, SQLite e utilizadas para o armazenamento, estão contidas na *API level 10*.

3.3.2 Interface e funcionamento

As três etapas como descrito em 3.2, são divididas em basicamente duas telas, uma referente à parte *on-line* e outra à *off-line*. A etapa intermediária acontece quando muda-se de uma tela para outra (Figura 3-15).

A tela principal, além das opções de se fazer o treinamento *off-line* ou tentar se localizar, possui também uma opção para exportar a base de dados para o cartão de

Figura 3-15 Tela principal do aplicativo.

memória. Assim, este pode ser analisado ou mesmo utilizado em outro sistema evitando que ele precise ser treinado também.

A tela de modo *off-line* foi criada de maneira que pudesse ser utilizada em testes em qualquer ambiente, sendo assim ela possui uma planta vazia com dimensões reguláveis (Figura 3-14). Ela espera que o usuário defina, com um toque simples sobre a tela, o local onde ele está, em seguida ele pergunta em qual quarto está presente e, caso confirmado, prossegue à aquisição de dados (Figura 3-17). Durante a aquisição de dados, momento em que a tela exibe "*Loading*", roda-se 20 vezes a rotina que verifica as redes disponíveis, suas potências em RSSI, e então armazena-se todas as informações na base de dados.

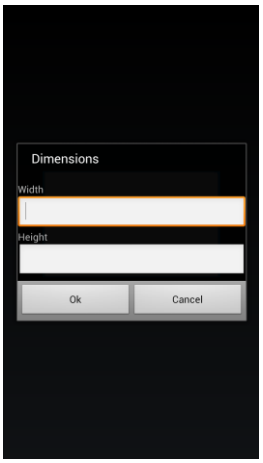


Figura 3-14 Dimensionamento do ambiente em modo *off-line*.

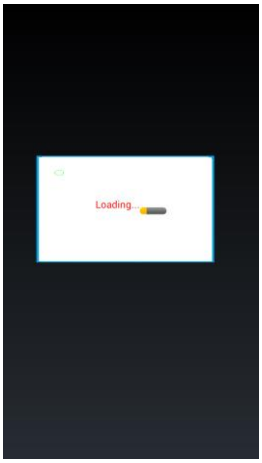


Figura 3-17 Amostrando de valores de RSSI.

Depois de coletados dados em diversos locais, retorna-se à tela principal e seleciona-se o modo *on-line*. Momento que o algoritmo de preparação dos dados transforma os valores amostrados em suas respectivas estimativas de distribuições de probabilidade atrás do KDE. Para uma implementação menos custosa computacionalmente é criada um *kernel* gaussiano, cujos valores são armazenados em um vetor na memoria do aparelho. Sendo assim, o calculo do valor do *kernel* em determinado ponto (Equação 3-2) torna-se apenas a busca do valor em uma vetor de consulta que já está armazenado.

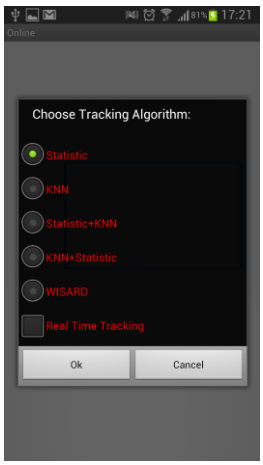


Figura 3-16 Escolha do método de localização.

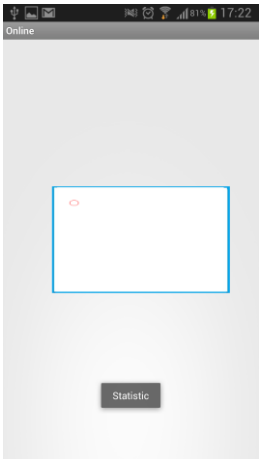


Figura 3-18 Exibindo a posição estimada pelo algoritmo selecionado.

Em modo *on-line*, diversas opções de método de estimação são exibidas na tela, elas aparecem para facilitar as futuras implementações, mas apenas o método estatístico, por KDE, está efetivamente implementado. Ao rodar o algoritmo, ele busca rede por três vezes, e utiliza o valor a média dos valores amostrados para aplicar às distribuições de probabilidade estimadas. Entretanto, devido ao grande número de pontos de acesso, o fator de comparação (Equação 3-1) tende a possuir um valor extremamente próximo a zero, e para evitar erros numéricos, utiliza-se seu valor em Bell. Ou seja, a equação se torna:

$$H_S(A_i)_B = \sum_{j=1}^{m} \log\left(P(S_j|A_i)\right)$$

Em seguida, é exibido em vermelho na tela a posição com maior valor fator de comparação $H_S(A_i)_B$. (Figura 3-18)

4 Resultados Experimentais

Os resultados apresentados nesse capítulo se baseiam em amostras feitas em um ambiente de 24m² com três cômodos. As amostras na fase de treinamento foram amostradas em 12 locais diferentes. No total houve 76 pontos de acesso diferentes reconhecidos, com um máximo de 43 e mínimo de 25 em locais diferentes. Foram feitas 50 amostragens por local.

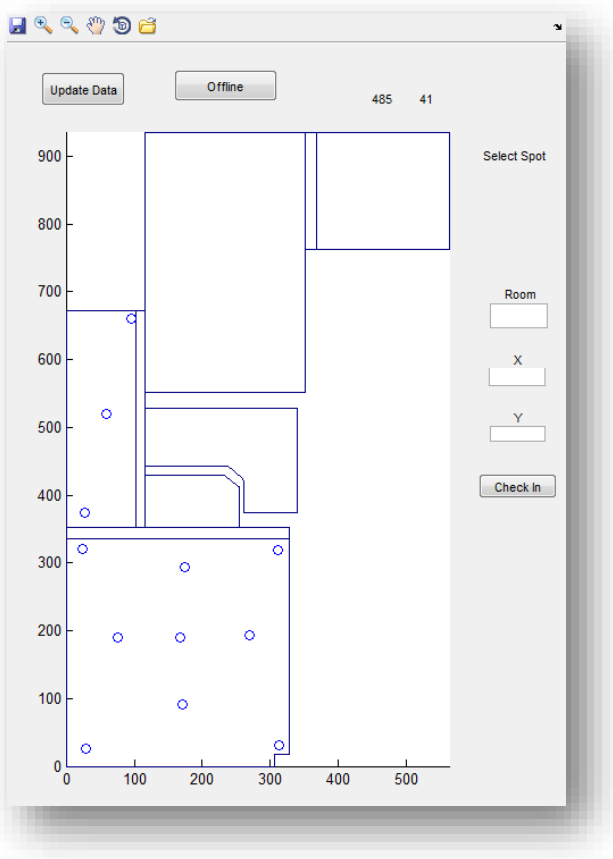


Figura 4-1 Interface do servidor em *Matlab*® exibindo os 12 locais amostrados (escala em cm).

4.1 Metodologia

Durante a fase de estudo os dados foram amostrados pelo aparelhos e diretamente transmitidos ao servidor estabelecido em um PC local, através de uma conexão WiFi, conexão essa cujos valores de potência estão sempre disponíveis exatamente pelo fato do aparelho à ela estar conectado.

4.1.1 Servidor *Matlab*®

O servidor foi programado em *Matlab*®, utilizando JAVA, o código está disponível no anexo, pois serve como contribuição para futuras pesquisas na área.

A interface criada para o servidor fez uso do *IGES toolbox*, *toolbox* desenvolvido para trabalhar com objetos IGES-CAD, i.e. formato de arquivo útil para exportar conteúdo desenvolvido em um programa CAD qualquer, para exibir a planta baixa do ambiente onde as amostras foram coletadas (Figura 4-1).

4.1.2 Comportamento do canal

Existe uma variação constante na potência medida em RSSI. Essa variação que pode ser interpretada como ruído possui uma esperada relação com a potência média, que pode ser relacionada com a distância. Há também sinais que estão no limite da recepção, em geral com um RSSI entre -95 e -100, que são medidos um número muito pequeno de vezes, não possibilitando uma real estimação do seu comportamento. Por ser um comportamento inerente ao canal o algoritmo deve ser robusto à essas variações.

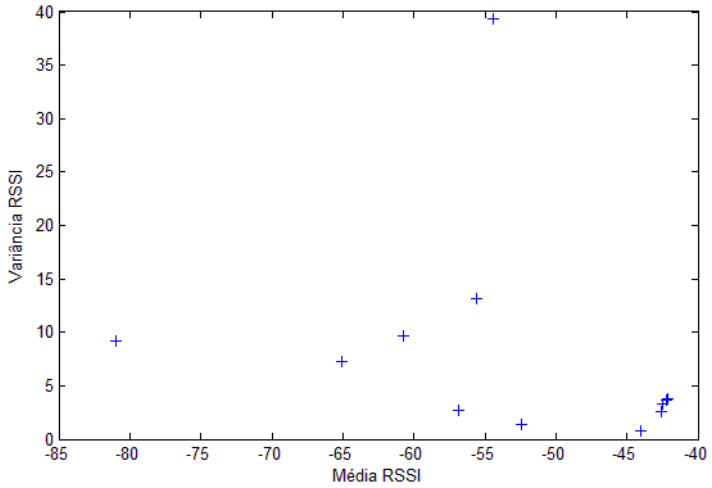


Figura 4-2 Aparente relação entre média e variância do RSSI.

4.1.3 Variação temporal

Além de sua variação instantânea, as potências apresentam uma variação menos sensível em curta escala de tempo (Figura 4-3). Essa variação que acaba por gerar a necessidade de uma atualização da base de dados. Para evitar isso, os testes foram feitos logo em seguida das medidas.

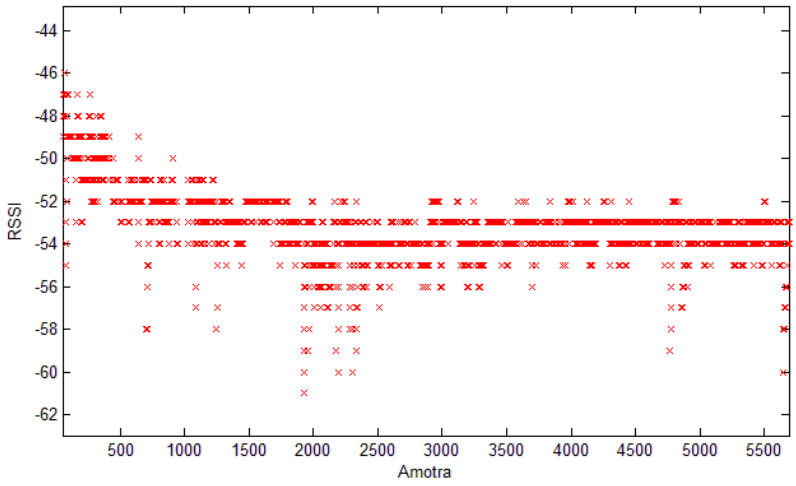


Figura 4-3 Variação temporal do sinal (1 amostra/segundo).

4.1.4 Posição da Antena

Como comentado em 3.1.1 e visto em Figura 2-2, a variação da posição da antena acarreta na variação da recepção do sinal, mas não será levado em consideração o ângulo da antena. Dessa maneira, com o intuito de melhorar a predição, durante a coleta de dados foram feitas alterações na posição da antena, isto é, o aparelho não foi mantido imóvel.

4.2 Qualidade de Localização

Nessa seção serão apresentados os resultados obtidos ao aplicar o métodos proposta nesse trabalho para a localização, no ambiente descrito em 4.1. Os valores utilizados como valores medidos *on-line* foram medidos logo após o treinamento, para evitar a influência da variação temporal do canal.

4.2.1 Acurácia e variância

A acurácia aqui descrita é entendida como a razão entre o número de vezes que o local foi corretamente inferido e o número total de tentativas.

A acurácia média encontrada depois de rodar o algoritmo 576, sobre os 12 locais, utilizando combinações de três medidas de RSSI sobre o conjunto dos valores medidos, foi de 90,3%, considerando apenas o local mais próximo como correto. Com acurácia por local mínima de 64% e máxima de 96%.

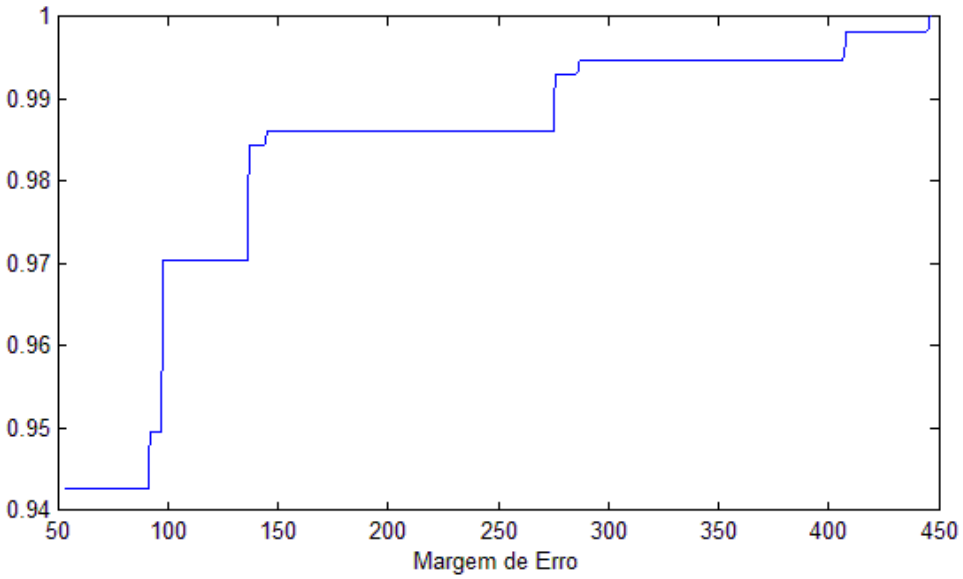


Figura 4-4 Percentual de estimativas dentro de certa margem de erro.

A acurácia dentro de um certo raio de precisão se mostrou excelente, com mais de 94% das estimativas errando por até 50 cm, vide (Figura 4-4) os patamares do gráfico são ausência de locais medidos com distâncias intermediárias.

4.2.2 Sensibilidade à variação do número de modems

Um fator importante de se estimar é a sensibilidade do sistema à presença ou não de uma grande quantidade de pontos de acesso. Para isso, foram retirados certos pontos de acesso da base de dados e, então, computado o algoritmo sobre essa nova base.

Para essa análise foram retirados primeiramente os pontos de acesso que apareceram em menos de 10% das amostras, ou seja, das 50 amostras por local foram retirados os que apareceram em média menos de 5 vezes. Dessa maneira foram retirados 40 dos 76 pontos de acesso presentes na base de dados (Figura 4-5).

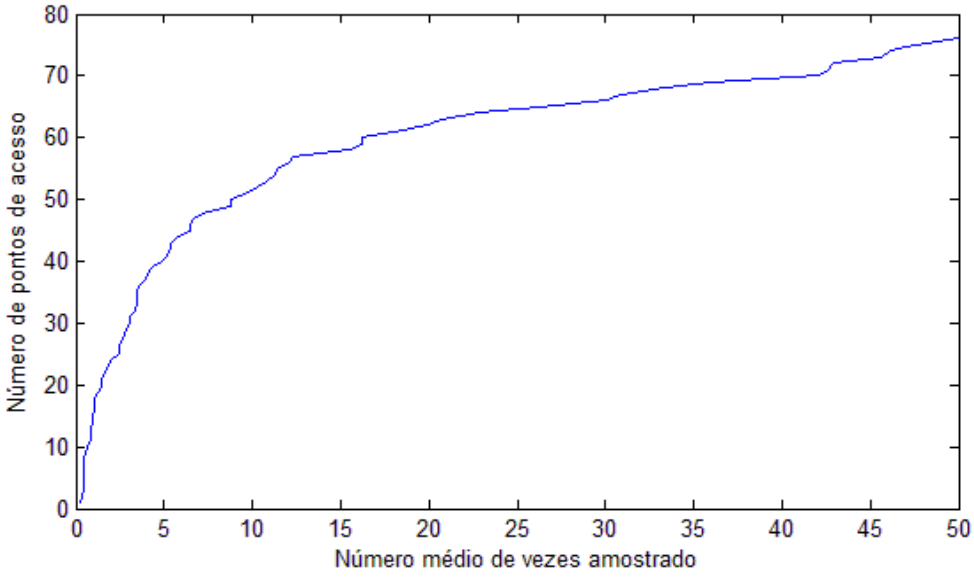


Figura 4-5 Análise da presença de média dos pontos de acesso. Média feita sobre as amostras em todos os locais.

Após a retirada dos 40 pontos de acesso da base de dados, o algoritmo rodado sobre a base com os 36 pontos de acesso restantes alcançou uma acurácia ainda maior. A acurácia média passou de 90,3% para 91,7%.

Os 36 pontos de acesso restantes foram organizados em função de um parâmetro definido para se ter uma estimativa de qualidade. Visto que é interessante que acha um ruído pequeno, i.e. uma pequena variação do RSSI em cada local, e ao mesmo tempo que o sinal tenha uma diferença de potência grande entre dois pontos, afim de que a potência deixe bem caracterizado qual o local, o parâmetro proposto foi:

$$Q_j = \frac{G_j}{\overline{L_J}}$$

Onde Q_j é parâmetro do j-ésimo ponto de acesso, G_i a variância global das médias dada por:

$$G_j = \mathrm{var}_i\left(\mathrm{E}_k\left({}_i^k B_j\right)\right)$$

Sendo ${}_i^k B_j$ o k-ésimo valor de RSSI amostrado no i-ésimo local para o j-ésimo ponto de acesso , var_i o operador de variância avaliada sobre os locais i e E_k o operador de valor esperado avaliado sobre as amostras k de RSSI. Além disso, a média das variância locais $\overline{L_J}$, é dada por:

$$\overline{L_J} = \mathrm{E}_i\left(\mathrm{var}_k\left({}_i^k B_j\right)\right)$$

Sendo o operador de variância agora sobre as amostras k de RSSI e o valor esperados sobre os locais i.

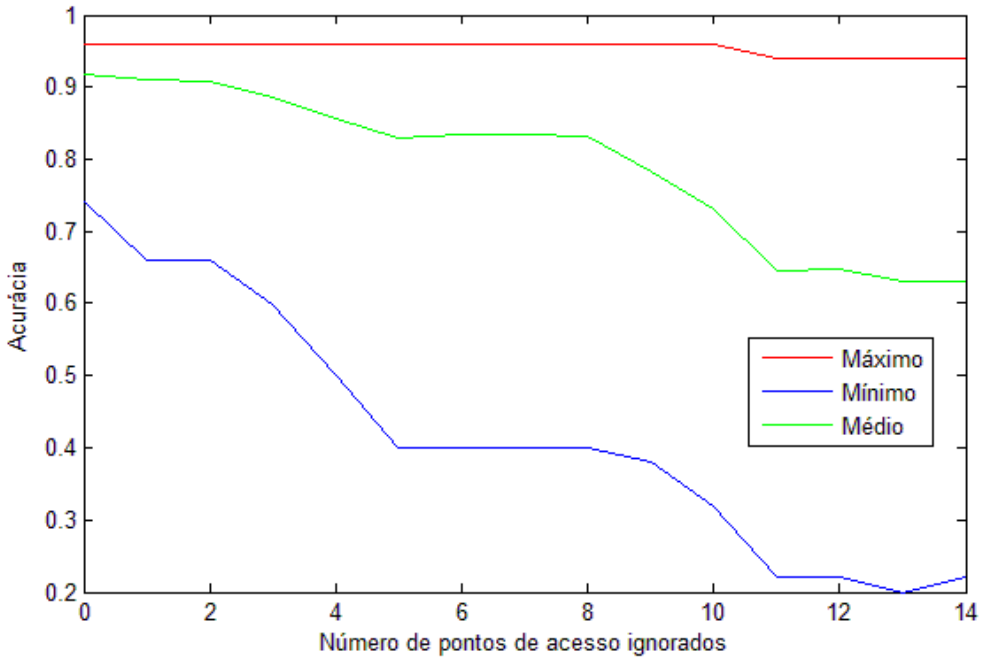


Figura 4-6 Variação da acurácia pela redução do número de pontos de acesso.

Feita a classificação segundo o parâmetro de qualidade proposto, foram retirados em ordem decrescente, começando pelo ponto de acesso com melhor parâmetro de qualidade, analisado o impacto sobre a acurácia das estimativas (Figura 4-6).

Interessante ressaltar que o valor máximo pertence a um local onde a malha de amostragem foi menos densa, consequentemente reduzindo as interseções dos valores de RSSI medidos e logo, tornando mais imune à erros. Por outro lado o valor mínimo de acurácia pertence a locais onde a malha de locais amostrados é densa.

5 Conclusão e Trabalhos Futuros

5.1 Discussão

O resultado foi muito além do esperado, superando diversos métodos já pesquisados. Esse resultado pode ser consequência de se evitar grande variação temporal, somado ao fato de estarem presentes um grande número de pontos de acesso, entretanto não deixa de ser um resultado promissor.

O sistema pode ter a precisão melhorada pela perda de pontos de acesso degenerados, i.e. onde a quantidade de informação disponível sobre ele descreve mal seu comportamento, visto que ao se retirar mais da metade dos pontos de acesso a acurácia foi aumentada.

Entretanto o algoritmo se mostra sensível à variação de pontos de acesso que provém informação relevante à localização, podendo ocorrer redução de mais de 10% de acurácia pela perda de apenas um ponto de acesso (Figura 4-6).

5.2 Possíveis Melhorias do Sistema Atual

Por ser uma área de pesquisa relativamente recente, com a maioria dos trabalhos datando do século XXI, há muitas melhorias que podem ser introduzidas tanto neste quanto na maioria dos sistemas propostos até o momento.

5.2.1 Filtro da base de dados

Como visto em 4.2.2, existe uma melhora do algoritmo por se retirar da base de dados informação de pontos de acessos mal estimados. Além disso, isso gera uma redução do tamanho da base de dados que pode se tornar útil em ambientem grandes.

O problema reside em se fazer um classificador eficiente, que retire da base apenas os pontos de acesso irrelevantes, sem afetar aqueles que são importantes a uma boa estimativa de localização.

5.2.2 Aprendizado

A ideia do aprendizado reside em não ser necessário refazer a fase de treinamento, onde são aferidos os valores de RSSI dos sinais em locais pré-determinados, de tempos em tempos para contra-atacar o efeito da variação temporal da assinatura de potência dos sinais.

A dificuldade está em conseguir um algoritmo de aprendizado que consiga manter sua precisão a longo prazo, utilizando de outros artifícios como reconhecimento de locais à partir do contexto, ou mesmo à partir dos demais pontos de acesso.

5.2.3 Uso da predição de movimento

Em um contexto de rastreamento, a informação sobre as últimas predições pode levar à precisões ainda maiores, para isso é necessário que se estime a probabilidade de uma pessoa ir de um local a outro.

Essa predição pode ser contextualizada, no sentido temporal, ou seja, estimar a rotina do ambiente segundo dia da semana, mês, ser ou não feriado, dentre outros.

5.2.4 Variação da Largura de Banda

A largura de banda utilizada nesse trabalho foi fixa, e baseada numa estimativa de função subjacente gaussiana, o que pode ser uma estimativa bem pobre para funções multimodais (3.2.2.2). Existe literatura sobre estimação por *kernel* com banda variável, a dificuldade reside em encontrar uma maneira computacionalmente eficiente de se fazer o cálculo da largura de banda ótima dentro de um aparelho móvel.

5.2.5 Solução discreta

Kernel discretos como feito nesse trabalho podem ser gerados à partir de *kernels* contínuos. Entretanto um estudo mais profundo sobre *Kernels* discretos pode apontar soluções melhores tanto para a escolha do *Kernel* e da largura de banda, quanto para a solução do caso de suporte compacto da distribuição de probabilidade subjacente.

5.2.6 Transformação de espaços

Um fato que este trabalho e diversos outros que trabalham com classificação admitem, e que não necessariamente é verdade, é que há uma relação direta entre o espaço dos sinal, o espaço com dimensão igual ao número de pontos de acesso onde ocorre a classificação, e o espaço físico onde se deseja localizar a pessoa.

Seria interessante fazer um estudo sobre a transformação que relaciona a métrica nos dois espaços, de maneira a aplicar a classificação sobre um espaço que tivesse uma relação direta com o espaço físico.

5.2.7 *Clusterização*

Há trabalhos (2.3.1.3) que fazem uso de *clusterização*, i.e. divisão em grupos, para tratar de grandes ambientes. O escopo de testes desse projeto não englobava grandes ambientes, dessa maneira é um ponto de melhoria caso seja expandido à ambientes maiores.

5.2.8 Integração com outros sensores

Aparelhos móveis modernos possuem uma série de sensores que podem ser utilizados para melhorar a precisão.

Acelerômetros podem servir para estimar a posição entre duas medidas dos valores de RSSI, ou mesmo servir como estimativa de posição para reduzir a margem de erro de localização.

Enquanto magnetômetros e giroscópios podem ajudar a incorporar o ângulo da antena às informações disponíveis para estimação.

6 Referências Bibliográficas

ASTUTO, B. **Um Sistema de Localização para Redes Wi-Fi Baseados em Níveis de Sinal e Modelo Referenciado de Propagação**. [S.l.]. 2006.

AZUMA, R. Tracking Requirements for Augmented. **Communications of the ACM**, v. 36, n. 7, p. 50-51, 1993.

BAHL, P.; PADMANABHAN, V. N. **RADAR:** An In-Building RF-based User Location and Tracking System. Proceedings of the IEEE Infocom. [S.l.]: [s.n.]. 2000. p. 775–784.

BARCLAY, L. W. **Propagation of Radiowaves**. 2ª. ed. [S.l.]: The Institution of Engineering and Technology, 2002.

BATTITI, R.; NHAT, T. L.; VILLANI, A. **Location-aware computing: a neural network model for determining location in wireless LANs**. University of Trento. [S.l.]. 2002.

CARPENTER, T.; BARRETT, J. **CWNA Certified Wireless Network Administrator Official Study Guide**. 4ª. ed. [S.l.]: [s.n.], 2007. 37-81 p.

CHEN, Y.-C. et al. **Sensor-Assisted Wi-Fi Indoor Location System for**. International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems 2005. [S.l.]: [s.n.]. 2005. p. 118–125.

CHEN, Y.-T. et al. **A RSSI-based Algorithm for Indoor Localization Using ZigBee in Wireless Sensor Network**. International Conference on Distributed Multimedia Systems (DMS 2009). [S.l.]: [s.n.]. 2009. p. 70-75.

FUREY, E.; CURRAN, K.; MCKEVITT, P. **Incorporating Past Human Movement into Indoor Locat**. the 10th International Conference on Information Technology & Telecommunication (IT&T 2010). [S.l.]: [s.n.]. 2010. p. 85-92.

FUREY, E.; CURRAN, K.; MCKEVITT, P. **A Bayesian Filter Approach to Modelling Human Movement Patterns for First Responders within Indoor Locations**. Intelligent Networking and Collaborative Systems (INCoS). [S.l.]: [s.n.]. 2011. p. 729 - 734.

GALILEO navigational system enters testing stage. **Deutsche Welle**, 2012. Disponivel em: <http://dw.de/p/16PRI >. Acesso em: 5 fev. 2013.

HASHEMI, H. The Indoor Radio Propagation Channel. **Proceedings of the IEEE**, v. 81, n. 7, p. 943-968, Julho 1993.

JANN, B. **Univariate kernel density estimation**. [S.l.]. 2007.

LIN, T.-N.; LIN, P.-C. **Performance Comparison of Indoor Positioning Techniques**. 2005 International Conference on Wireless Networks, Communications and Mobile Computing. [S.l.]: [s.n.]. 2005. p. 1569-1574.

MILLS, P. Efficient statistical classification of satellite measurements. **International Journal of Remote Sensing**, v. 32, n. 21, p. 6109-6132, 2011.

MIU, A. K. L. **Design and Implementation of an Indoor Mobile Navigation**. [S.l.]. 2002.

MOLKDAR, D. Review on radio propagation into and within buildings. **IEE Proceedings-H (Microwaves, Antennas and Propagation)**, v. 138, p. 61-73, 1991.

MÜLLER, H.-G. Smooth Optimum Kernel Estimators of Desnsities Regression Curves and Modes. **The Annals of Statistics**, v. 12, n. 2, p. 766-774, 1984.

NAVARRO, E. et al. **Wi-Fi Localization Using RSSI**. California Polytechnic State University. [S.l.].

PALANIAPPAN, R. et al. **Autonomous RF Surveying Robot for Indoor**. 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN). [S.l.]: [s.n.]. 2011. p. 21-23.

POPLETEEV, A. **Indoor positioning using FM radio signals**. University of Trento. [S.l.]. 2011.

RAJAGOPALAN, B.; LALL, U. A Kernel Estimator for Discrete Distributions. **Nonparametric Statistics**, v. 4, p. 409-426, 1995.

RAPPORT, T. S. **Wireless Communications – Principles and Practice**. IEEE Press. [S.l.]: [s.n.]. 1996.

RICE, S. O. Mathematical analysis of Random Noise. **Bell Systems Technical Journal**, v. 23, 1944.

RUSSIA restores its orbital GLONASS group. **The Voice of Russia**, 2011. Disponivel em: <http://voiceofrussia.com/2011/10/03/58065478/>. Acesso em: 20 fev. 2013.

SECO, F. et al. **A survey of mathematical methods for indoor**. IEEE Int. Symp. Intelligent Signal Processing WISP 2009. [S.l.]: [s.n.]. 2009. p. 9-14.

SEGATA, N.; BLANZIERI, E. Fast and Scalable Local Kernel Machines. **Journal of Machine Learning Research**, n. 11, 2010.

SEIDEL, S. Y.; RAPPORT, T. S. **914 MHz path loss prediction Model for Indoor Wireless Communications in Multi-floored buildings**. IEEE Trans. on Antennas & Propagation. [S.l.]: [s.n.]. 1992.

TERRELL, G. R.; SCOTT, D. W. Variable Kernel Density Estimation. **The Annals of Statistics**, v. 20, n. 3, p. 1236-1265, 1992.

WAND, M. P.; SCHUCANY, W. R. **Gaussian-Based Kernels for Curve Estimation and Window Width Selection**. [S.l.]. 1989.

WANT, R. et al. The Active Badge Location System. **ACM Transactions on Information**, v. 40, n. 1, p. 91-102, 1992.

WEIRONG, Z. Beidou satellite navigation system to cover whole world in 2020. **China Military Online**, 2010. Disponivel em: <http://eng.chinamil.com.cn/news-

channels/china-military-news/2010-05/20/content_4222569.htm>. Acesso em: 8 fev. 2013.

YOUSSEF, M. A.; AGRAWALA, A.; SHANKAR, A. U. **WLAN Location Determination via Clustering and Probability Distributions**. PerCom2013. [S.l.]: [s.n.]. 2013.

ZUCCHINI, W. **Applied Smoothing Techniques - Part 1: Kernel Density Estimation**. [S.l.]. 2003.

7 Anexo

Código para o estabelecimento de um servidor em *Matlab*®

MatServerBar.m

```
% SERVER for data gathering
%
%   Usage   -   server(message,   output_port,   number_of_retries,   number_of_samples,
use_or_not_progress_bar)
% KeyList = BSSID access points values
% ValuesList = RSSI values
% time = time of each recived sample, counting from the start.
%
function   [KeysList,ValuesList,time]   =   MatServerBar(output_port,   number_of_retries,
samples,wbar)

import java.net.ServerSocket
import java.net.InetAddress
import java.net.Socket
import java.nio.channels.Channels
import java.io.*

if (nargin < 2)
    number_of_retries = 20; % set to -1 for infinite
end
if (nargin < 3)
    samples = 20;
end
if (nargin < 4)
    wbar = 1;
end
retry             = 0;
server_socket  = [];
input_socket = [];
values      = [];
keys        = [];
KeysList = {};
Cancelption = MException('Tenkel:ProgressBar:Cancel', 'ProgressBar has been cancelled.');
LoopEnd = MException('Tenkel:DataGathering:End', 'All samples have been collected.');
time = zeros(samples,1);
ValuesList = zeros(100,samples);steps=10000;
if wbar
    h = waitbar(0,'1','Name','RSSI Mesurement',...
        'CreateCancelBtn',...
        'setappdata(gcbf,''canceling'',1)');
    setappdata(h,'canceling',0);
end

while true


    retry = retry + 1;

    try
        if wbar
            if getappdata(h,'canceling')
                throw(Cancelption);
            end
        end

        if retry > number_of_retries
            break;
        end

        if wbar
            waitbar(retry/number_of_retries,h,sprintf(['Try %d of %d ' ...
                'on port : %d'], retry,number_of_retries, output_port));
        end
        % wait for 1 second for client to connect server socket
        server_socket = ServerSocket(output_port);
```

```
        server_socket.setSoTimeout(1000);

        input_socket = server_socket.accept;

        if wbar
            waitbar(0,h,'Waiting new packets...');
        end

        % get a buffered data input stream from the socket
        input_stream   = input_socket.getInputStream;
        in = DataInputStream(input_stream);

        % read data from the socket - wait a short time first
        % exec = true;
        cnt=1;
        tic;

        while true
            pause(0.01);
            try
                %message = in.readUTF;
                while true
                    if in.available && in.readByte=='S'
                        if wbar
                            waitbar(cnt/samples,h,'Gathering Data...');
                        end
                        time(cnt)=toc;
                        break;
                    end

                    if time(1)>0
                        dtime = toc-time(cnt-1);
                        if dtime > 3 && wbar
                            waitbar(cnt/samples,h,sprintf('Waiting new packets for %3.1f
s...',dtime));
                        end
                    end

                    if wbar
                        if getappdata(h,'canceling')
                            throw(Cancelption);
                        end
                    end
                end

                size = in.readByte;

                keys = cell(size,1);
                values = zeros(size,1);

                for i = 1:size
                    keys{i} = char(in.readUTF);
                    values(i) = in.readByte;
                end

                %fprintf(1,'cnt - %d \n',cnt);

                if in.readByte ~= 'E'
                    continue;
                end


                for i = 1:size
                    Index=find(strcmp(keys{i},KeysList));
                    if isempty(Index)
                        KeysList{end+1}=keys{i};
                        Index = length(KeysList);
                    end
                    ValuesList(Index,cnt)=values(i);
                end

%                 if mod(cnt,1000)==0
%                     save KeysList.mat
%                     save ValuesList.mat
%                     save time.mat
%                 end
```

```
                cnt=cnt+1;

                if cnt>samples
                    throw(LoopEnd);
                end


            catch ME01
                if ~isempty(in)
                    in.close
                end
                rethrow(ME01);
            end
        end

    catch ME
        % pause before retrying
        if ~isempty(input_socket)
            input_socket.close;
        end
        if ~isempty(server_socket)
            server_socket.close;
        end
        if strcmp(ME.identifier,Cancelption.identifier)
            if wbar
                delete(h);
            end
            rethrow(ME);
        end
        if strcmp(ME.identifier,LoopEnd.identifier)
            break
        end
        pause(0.1);
    end
end

if wbar
    delete(h);
end

if retry > number_of_retries
    warndlg('Waiting too long...','Time Out');
    throw(Cancelption);
end
ValuesList=ValuesList(1:length(KeysList),:);
```

Android® Database table declarations

```
package com.example.autotracking;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OptionalDataException;
import java.io.StreamCorruptedException;
import java.nio.channels.FileChannel;
import java.util.ArrayList;

import android.content.ContentValues;
import android.content.Context;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import android.widget.Toast;

public class DataManager {
	public static class Local {
		public static final String ID = "Cd_local";
		public static final String PX = "Vi_px";
		public static final String PY = "Vi_py";
		public static final String ROOM = "Cd_room";
	}

	private static final String LocalTable = "Local";

	public static class Access_Point {
		public static final String ID = "Cd_access_point";
		public static final String NAME = "Nm_name";
	}

	private static final String Access_PointTable = "Access_Point";

	public static class KSD {
		public static final String ID_LOCAL = "Cd_local";
		public static final String ID_AP = "Cd_access_point";
		public static final String LINK = "Nm_ksd_link";
	}

	private static final String KSDTable = "KSD";

	public static class Sample {
		public static final String ID = "Cd_sample";
		public static final String VALUE = "Vi_value";
		public static final String LOCAL = "Cd_local";
		public static final String AP = "Cd_access_point";
	}

	private static final String SampleTable = "Sample";

	private static final String DB_NAME = "FindDB";
	private static final int DB_VERSION = 1;

	private DBMng DBManager;
	private final Context theContext;
	private SQLiteDatabase DBFind;
	private ArrayList<Cursor> cursors = new ArrayList<Cursor>();

	private static class DBMng extends SQLiteOpenHelper {

		public DBMng(Context context) {
			super(context, DB_NAME, null, DB_VERSION);
			// TODO Auto-generated constructor stub
		}

		@Override
		public void onCreate(SQLiteDatabase db) {
			// Foreign Keys = ON
			db.execSQL("PRAGMA foreign_keys = ON;");
```

```
                        // Local
                        db.execSQL("CREATE TABLE " + LocalTable + "("
                                        + Local.ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                                        + Local.PX + " INTEGER, "
                                        + Local.PY + " INTEGER, "
                                        + Local.ROOM +" INTEGER, "
                                        + "UNIQUE(" + Local.PX + ", " + Local.PY
+ ") );");


                        // Access_Point
                        db.execSQL("CREATE TABLE " + Access_PointTable + "("
                                        + Access_Point.ID + " integer primary
key autoincrement, "
                                        +   Access_Point.NAME   +   "   TEXT
UNIQUE);");

                        // KSD
                        db.execSQL("CREATE TABLE " + KSDTable + "("
                                        + KSD.ID_AP + " INTEGER,"
                                        + KSD.ID_LOCAL + " INTEGER, "
                                        + KSD.LINK + " TEXT,"
                                        + "FOREIGN KEY(" + KSD.ID_LOCAL + ")
REFERENCES " + LocalTable + "(" + Local.ID + "),"
                                        + "FOREIGN KEY(" + KSD.ID_AP + ")
REFERENCES " + Access_PointTable + "(" + Access_Point.ID + "),"
                                        + "PRIMARY KEY(" + KSD.ID_AP + ", " +
KSD.ID_LOCAL + ") );");

                        // Sample
                        db.execSQL("CREATE TABLE " + SampleTable + "("
                                        + Sample.ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, "
                                        + Sample.VALUE + " INTEGER,"
                                        + Sample.LOCAL + " INTEGER,"
                                        + Sample.AP + " INTEGER,"
                                        + "FOREIGN KEY(" + Sample.LOCAL + ")
REFERENCES " + KSDTable + "(" + KSD.ID_LOCAL + "),"
                                        + "FOREIGN KEY(" + Sample.AP + ")
REFERENCES " + KSDTable + "(" + KSD.ID_AP + ") );");


                }

                @Override
                public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
                        // TODO Auto-generated method stub

                }

        }

        public DataManager(Context c) {
                theContext = c;
        }
```

```
	public DataManager open() {
		DBManager = new DBMng(theContext);
		DBFind = DBManager.getWritableDatabase();
		return this;
	}

	public void close() {
		for(Cursor cursor : cursors)
			cursor.close();
		DBFind.close();
		DBManager.close();
	}

	public void insert(int power, String APcode, int localX, int localY,
			int room) {
		long localid, apid;

		// Local
		ContentValues entry = new ContentValues();
		entry.put(Local.PX, localX);
		entry.put(Local.PY, localY);
		entry.put(Local.ROOM, room);
		localid = DBFind.insert(LocalTable, null, entry);
		if(localid == -1){
			Cursor c = DBFind.query(LocalTable, new
String[]{"rowid"}, Local.PX + " == " + localX + " AND " + Local.PY + " == " +
localY, null, null, null, null);
			c.moveToFirst();
			localid = c.getLong(0);
			c.close();
			}

		// Access Point
		entry.clear();
		entry.put(Access_Point.NAME, APcode);
		apid = DBFind.insert(Access_PointTable, null, entry);
		if(apid == -1){
			Cursor c = DBFind.query(Access_PointTable, new
String[]{"rowid"}, Access_Point.NAME + " == '" + APcode + "'", null, null,
null, null);
			c.moveToFirst();
			apid = c.getLong(0);
			c.close();
		}

		// KSD
		entry.clear();
		entry.put(KSD.ID_AP, apid);
		entry.put(KSD.ID_LOCAL, localid);
		entry.put(KSD.LINK, "");
		DBFind.insert(KSDTable, null, entry);

		// Sample
		entry.clear();
		entry.put(Sample.VALUE, power);
		entry.put(Sample.LOCAL, localid);
		entry.put(Sample.AP, apid);
		DBFind.insert(SampleTable, null, entry);
```

```
        }

        public Cursor Local(String[] infos, String Where) {
                Cursor c = DBFind.query(LocalTable, infos, Where, null, null,
null, null);
                cursors.add(c);
                return c;
        }

        public Cursor AP(String[] infos, String Where) {
                Cursor c = DBFind.query(Access_PointTable, infos, Where, null,
null, null,
                                null);
                cursors.add(c);
                return c;
        }

        public KDE KSDFunction(long localid, long apid) {
                InputStream is = null;
                ObjectInputStream ois = null;
                KDE function = null;
                String where = KSD.ID_AP + " == " + apid + " AND " + KSD.ID_LOCAL
                                + " == " + localid;
                Cursor list = DBFind.query(KSDTable, new String[] { KSD.LINK,
                                KSD.ID_AP, KSD.ID_LOCAL }, where, null, null,
null, null);
                if (list != null) {
                        list.moveToFirst();
                        String filename;
                        if (list.getCount() == 0){
                                list.close();
                                return KDE.kde;
                        }

                        filename = list.getString(0);
                        try {
                                Log.d("DTM", "opening...");
                                is                      =                       new
FileInputStream(theContext.getFilesDir()
                                                + File.separator + filename);
                                Log.d("DTM",        "Opened        "        +
theContext.getFilesDir()
                                                + File.separator + filename);
                        } catch (FileNotFoundException e) {
                                e.printStackTrace();
                                list.close();
                                return null;
                        }

                        try {
                                Log.d("DTM", "null");
                                ois = new ObjectInputStream(is);
                                Log.d("DTM", "Opened2");
                        } catch (StreamCorruptedException e) {
                                Toast.makeText(theContext, "File " + filename +
" corrupted.",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        } catch (IOException e) {
```

```
                                Toast.makeText(theContext,
                                                "File " + filename + " cannot be
read.",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        }

                        try {
                                function = (KDE) ois.readObject();
                                Log.d("DTM", "Opened3");
                        } catch (OptionalDataException e) {
                                e.printStackTrace();
                        } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }


                        try {
                                ois.close();
                        } catch (IOException e) {
                                Toast.makeText(theContext,
                                                "Error Closing File " + filename
+ ".",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        }

                        list.close();

                        return function;
                }

                return null;
        }

        public void KSDWarming() {
                ContentValues update = new ContentValues();
                FileOutputStream fos = null;
                ObjectOutputStream oos = null;
                Cursor KSDlist = DBFind.query(KSDTable, new String[] { "rowid",
        KSD.ID_AP, KSD.ID_LOCAL }, null, null, null, null, null);
                Cursor Samplelist = null;
                KDE ksfunction = null;
                String filename = "";
                String where = Sample.LOCAL + " = ? AND " + Sample.AP + " = ?";
                String[] columns = new String[] { Sample.VALUE };


                for      (KSDlist.moveToFirst();      !KSDlist.isAfterLast();
KSDlist.moveToNext()) {
                        Samplelist = DBFind
                                        .query(SampleTable,
                                                        columns,
                                                        where,
                                                        new      String[]      {
String.valueOf(KSDlist.getLong(2)) ,
```

```
        String.valueOf(KSDlist.getLong(1)) }, null, null, null);

                        ksfunction = new KDE(Samplelist, 0);
                        filename = "ks" + KSDlist.getLong(0);
                        try {
                                fos                     =                       new
FileOutputStream(theContext.getFilesDir() + File.separator + filename);
                        } catch (FileNotFoundException e) {

                                Toast.makeText(theContext,
                                                "File " + filename + " cannot be
created.",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        }
                        try {
                                oos = new ObjectOutputStream(fos);
                        } catch (IOException e) {
                                Toast.makeText(theContext,
                                                "File " + filename + " cannot be
accessed.",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        }
                        try {
                                oos.writeObject(ksfunction);
                        } catch (IOException e) {
                                Toast.makeText(theContext,
                                                "File " + filename + " cannot be
written.",
                                                Toast.LENGTH_LONG).show();
                                e.printStackTrace();
                        }

                        update.clear();
                        update.put(KSD.LINK, filename);
                        DBFind.update(KSDTable,   update,   "rowid   ==   "   +
KSDlist.getLong(0),
                                        null);

                }
                try {
                        oos.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                KSDlist.close();
                Samplelist.close();

        }

        public Cursor Samples(long localid, long apid) {
                String where = Sample.AP + " == " + apid + " AND " + Sample.LOCAL
                                + " == " + localid;
                Cursor   list   =   DBFind.query(SampleTable,   new   String[]   {
Sample.VALUE },
                                where, null, null, null, null);
```

```
                cursors.add(list);
                return list;
        }

        public void DBexport(String toPath) throws IOException{
                String dbPath = DBFind.getPath();
                Log.d("DBexport", dbPath);
                FileInputStream newDb = new FileInputStream(new File(dbPath));
                FileOutputStream    toFile    =    new    FileOutputStream(new
File(toPath,"BDbackup.db"));
            FileChannel fromChannel = null;
        FileChannel toChannel = null;
        try {
            fromChannel = newDb.getChannel();
            toChannel = toFile.getChannel();
            newDb.close();
            toFile.close();
            fromChannel.transferTo(0, fromChannel.size(), toChannel);
        } finally {
            try {
                if (fromChannel != null) {
                    fromChannel.close();
                }
            } finally {
                if (toChannel != null) {
                    toChannel.close();
                }
            }
        }
        }
}
```