UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MÉTODO DE MULTILATERAÇÃO PARA ALGORITMOS DE LOCALIZAÇÃO EM REDES DE SENSORES SEM FIO

DISSERTAÇÃO DE MESTRADO

Crístian Müller

Santa Maria, RS, Brasil

2014

MÉTODO DE MULTILATERAÇÃO PARA ALGORITMOS DE LOCALIZAÇÃO EM REDES DE SENSORES SEM FIO

Crístian Müller

Dissertação apresentada ao Curso de Mestrado Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientador: Profº. Dr. João Baptista dos Santos Martins

Co-orientador: Profº. Dr. Renato Machado

Santa Maria, RS, Brasil

2014

Müller, Crístian

Método de Multilateração para Algoritmos de Localização em Redes de Sensores Sem Fio / por Crístian Müller. – 2014.
96 f.: il.; 30 cm.

Orientador: João Baptista dos Santos Martins
Co-orientador: Renato Machado
Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2014.

I. Martins, João Baptista dos Santos. II. Machado, Renato. III. Título.

E-mail: cristianmuller.50@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

MÉTODO DE MULTILATERAÇÃO PARA ALGORITMOS DE LOCALIZAÇÃO EM REDES DE SENSORES SEM FIO

elaborada por
Crístian Müller

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

João Baptista dos Santos Martins, Dr.
(Presidente/Orientador)

Leonardo Londero de Oliveira, Dr. (UFSM)

Bartolomeu Ferreira Uchôa Filho, Ph.D. (UFSC)

Santa Maria, 18 de Março de 2014.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Nery Müller e Ile Grohs Müller, e demais familiares que sempre me apoiaram.

À minha namorada, Sueli Elisa Kullmann, pelo carinho, confiança, paciência e compreensão.

Ao meu amigo Guilherme Dietrich.

Ao meu orientador Prof. João Baptista dos Santos Martins, ao meu Co-orientador Prof. Renato machado, ao Prof. Leonardo Londero de Oliveira e ao Prof. César Augusto Prior por me orientarem e apoiarem na minha trajetória.

Aos demais colegas e professores do Gmicro pelo apoio e colaboração.

Ao Prof. Cesar Albenes Zeferino e aos integrantes do LEDS da UNIVALI, por terem me acolhido, pelos conselhos e pelos momentos de descontração.

A FAPERGS pelo suporte financeiro.

Em fim, a todos que de alguma forma tenham me ajudado.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

MÉTODO DE MULTILATERAÇÃO PARA ALGORITMOS DE LOCALIZAÇÃO EM REDES DE SENSORES SEM FIO
AUTOR: CRÍSTIAN MÜLLER
ORIENTADOR: JOÃO BAPTISTA DOS SANTOS MARTINS
CO-ORIENTADOR: RENATO MACHADO
Local da Defesa e Data: Santa Maria, 18 de Março de 2014.

As redes de sensores sem fio tiveram um grande progresso na última década e estão cada vez mais presentes em diversos campos como a segurança e monitoramento de pessoas, animais ou itens, medicina, área militar entre muitas outras que, devido à evolução tecnológica, tornaram-se viáveis. As principais características dos nós sensores, denominados nodos, constituintes destas redes são as de possuírem recursos extremamente limitados, sendo estes a capacidade de processamento e de armazenamento de dados, taxa de transmissão de dados e energia disponível para operação. Deste modo, em redes com centenas ou até milhares de nodos, seria inviável que todos estes possuíssem dispositivos de posicionamento global para se localizarem, pois estes acarretariam num considerável aumento de custo e consumo de potência. Como o conhecimento de sua localização, por parte dos nodos, é necessário em aplicações como rastreamento, monitoramento e coleta de dados ambientais, foram criados algoritmos de localização com a função de viabilizar e/ou tornar mais precisa esta tarefa. Assim, neste trabalho é apresentado o desenvolvimento de um método de multilateração iterativo de baixa complexidade para o uso em algoritmos de localização. Para provar que este novo método é eficiente, foi criado um simulador simples baseado no software Matlab com o intuito de avaliar, em termos de erro na localização, a precisão e robustez deste em cenários com disposição aleatória dos nodos de referência, modelo de propagação log-normal com sombreamento e estimação das distâncias através da potência do sinal recebido. Nestas condições, o método de multilateração desenvolvido apresentou uma perda de precisão desconsiderável em relação ao de máxima verossimilhança e com um baixo número de iterações. Desta forma foi possível aumentar a precisão dos algoritmos de localização sem que isto acarrete num aumento de complexidade e de consumo de potência.

Palavras-chave: redes de sensores sem fio, algoritmos de localização, multilateração, máxima verossimilhança.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

MULTILATERATION METHOD FOR WIRELESS SENSOR NETWORKS LOCATION ALGORITHMS

AUTHOR: CRÍSTIAN MÜLLER
ADVISOR: JOÃO BAPTISTA DOS SANTOS MARTINS
COADVISOR: RENATO MACHADO
Defense Place and Date: Santa Maria, March 18st, 2014.

The wireless sensor networks have made great progress in the last decade and are increasingly present in several fields like security and monitoring of persons, animals or items, medicine, military area and many others that, due to technological developments, have become viable. These networks are formed by sensor nodes that have extremely limited resources, such as processing and data storage capacity, data transmission rate and energy available for operation. Thus, in networks with hundreds or even thousands of nodes, it is unfeasible to locate each one of them with global positioning devices, because those will considerably increase the cost and power consumption. As localization knowledge by the nodes is required in applications such as tracking, monitoring and environmental data collection, location algorithms were created to cheapen and/or improve this task. Thus, this master thesis presents the development of a low complexity iterative multilateration method, since most location algorithms uses some kind of multilateration. To prove this new method efficiency, a simple simulator based on the Matlab software was created in order to evaluate, in terms of location error, accuracy and robustness in a scenario with random arrangement of the reference nodes, log-normal propagation model with shadowing and received signal strength distance estimation. Under these conditions, the presented multilateration method presents inconsiderable loss of accuracy in comparison to the maximum likelihood method and also a low number of iterations is required. In this way was possible to increase the location algorithms accuracy without this entailing an increase in complexity and power consumption.

Keywords: wireless sensor network, location algorithms, multilateration, maximum likelihood.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE APÊNDICES

LISTA DE ABREVIATURAS E SIGLAS

RSSF	Redes de Sensores Sem Fio
DSN	*Distributed Sensor Networks*
DARPA	*Defense Advanced Research Projects Agency*
MEMS	*Microelectromechanical systems*
RF	Radiofrequência
CI	Circuito Integrado
GPS	Sistema de Posicionamento Global
AOA	*Angle of Arrival*
TOA	*Time of Arrival*
TDOA	*Time Difference of Arrival*
RSSI	*Received Signal Strength Indicator*
IEEE	*Institute of Electric and Electronic Engineers*
MQL	Mínimos Quadrados Linearizado
MV	Máxima Verossimilhança
MRQ	Menor Residual Quadrático
EL	Erro de Localização
PR	Posição Real
SNR	*Signal-to-Noise Ratio*
WNCS	*Wireless Networked Control Systems*
PHY	*Physical Layer*
MAC	*Medium Access Control*
WPAN	*Wireless Personal Area Networks*
LR-WPANs	*Low-Rate WPANs*
LQI	*Link Quality Indication*)
CSMA-CA	*Carrier Sense Multiple Access with Collision Avoidance*
CAP	*Contention Access Period*
CFP	*Contention-Free Period*
CCA	*Clear Channel Assessment*
TDMA	*Time Division Multiple Access*
PRR	*Packet Reception Rate*
MVI	Máxima Verossimilhança Iterativa
ELM	Erro de Localização Médio

AP	Aproximação Polinomial
NIM	Número de Iterações Médio
CORDIC	*Coordinate Rotation Digital Computer*
HDL	*Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*

SUMÁRIO

1 INTRODUÇÃO

As pesquisas modernas na área de Redes de Sensores Sem Fio (RSSF) iniciaram-se por volta de 1980, com a criação do projeto *Distributed Sensor Networks* (DSN) pela Agência de Projetos de Pesquisa Avançada de Defesa dos Estados Unidos, denominada DARPA (do inglês *Defense Advanced Research Projects Agency*), conforme (CHONG; KUMAR, 2003). O objetivo do projeto DSN era criar uma rede com muitos nodos autônomos de baixo custo e distribuídos espacialmente, pela qual informações seriam roteadas até chegar ao nó que faria o melhor uso delas. Este era um projeto ambicioso levando-se em conta a tecnologia existente na época.

(CHONG; KUMAR, 2003) mostra que durante os anos 80 e 90 as aplicações para as redes de sensores concentravam-se em aplicações militares, como monitoramento de ameaças e controle de armas. Nestas décadas, a tecnologia existente não permitia alcançar o objetivo inicial proposto pela DSN, visto que os nodos eram grandes e construídos com componentes customizados.

A partir dos anos 2000, com a crescente capacidade de integração de circuitos eletrônicos em silício, conforme a lei de Moore (MACK, 2011), tornou-se possível criar sistemas embarcados com capacidades de processamento e comunicação sem fio surpreendentes. Isto, agregado a também crescente evolução dos *Microelectromechanical Systems* (MEMS), conforme (J.W. GARDNER; AWADELKARIM, 2001), permitiu a criação dos nodos que conhecemos hoje em dia. Estes geralmente consistem de sistemas embarcados compostos por sensores, microcontrolador, comunicação por radiofrequência (RF) e baterias.

Em (21 IDEAS FOR THE 21ST CENTURY, 1999), a tecnologia de RSSF foi considerada como uma das 21 mais importantes do século 21. Isto se deve à grande abrangência de aplicações desta, antes nem mesmo sonhadas, viabilizadas pela evolução tecnológica. Conforme (CHONG; KUMAR, 2003), algumas das principais aplicações, além das militares, são: infraestrutura para segurança, monitoramento domiciliar e do meio-ambiente, sensoriamento industrial, controle de tráfego, entre outras.

A miniaturização dos nodos chegou ao seu limite através da Tecnologia *Smart Sensor*, conforme (KAHN; KATZ; PISTER, 1999), tendo como proposta a criação de nodos com alguns milímetros de tamanho, através da integração de todo o sistema em uma única pastilha de silício. Estes têm capacidade de sensoriamento, comunicação óptica e são auto-alimentados por células

solares.

Por fim, (EDWARDS, 2012) mostra um dos desafios atuais das RSSF, que é a vigilância e o controle urbano. Neste contexto, desejam-se obter em tempo real informações de diversos tipos de sensores espalhados tanto em ambientes externos como internos. Os principais propósitos desta integração seriam facilitar o controle, qualidade de vida e economia de recursos, tendo como exemplos o controle do tráfego, da qualidade do ar, do consumo de água, energia elétrica, e assim por diante.

1.1 Motivação

Assim como o número de possíveis aplicações para as RSSF é enorme, os desafios também são. Conforme (CHONG; KUMAR, 2003), o desenvolvimento das RSSF depende de três áreas de pesquisas distintas: sensores, comunicações e computação (hardware, software e algoritmos), de modo que, tanto os avanços isolados como os em conjunto guiam as pesquisas nesta área. Já (AKYILDIZ et al., 2002), mostra que o projeto de uma RSSF é influenciado por uma série de fatores, muitas vezes ligados aplicação, incluindo tolerância a falhas, escalabilidade, custo, restrições de hardware, topologia, ambiente, meios de transmissão e consumo de potência.

Como exemplo de aplicação, pode-se considerar uma RSSF projetada para coletar informações ambientais de uma região de mata densa, montanha ou caverna. Nestas condições, um dos principais fatores é a tolerância a falhas, visto que por se tratar de regiões perigosas e com agentes ambientais ativos (animais e exposição às intemperes da natureza como sol e chuva), a danificação dos nodos pode ocorrer antes do esperado. Outro fator importante é o consumo de energia, pois por se tratar de regiões de difícil acesso, a substituição das baterias torna-se impossível ou inviável de ser realizada em um curto espaço de tempo, e a utilização de nodos auto-alimentados inviável economicamente e/ou por não se ter uma fonte de energia adequada (por exemplo, na mata fechada ou caverna não se consegue utilizar a energia solar).

Devido a essa diversidade de fatores que influenciam no projeto e desenvolvimento das RSSF, a utilização de simuladores tornou-se algo praticamente indispensável. Segundo (LEVIS et al., 2003), os simuladores têm um importante papel para estudar alternativas de projeto em ambientes controlados, explorar configurações difíceis de serem construídas fisicamente e observar interações que são difíceis de serem capturadas em um sistema real. Alguns dos principais simuladores utilizados na literatura são o NS-2 (MOZUMDAR et al., 2008), TOSSIM

(ZHENYU, 2009) e OMNeT++ (STEHLIK, 2011), os quais são softwares livres.

Outro aspecto importante das RSSF, que pode ser destacado é o conhecimento de sua localização por parte dos nodos, pois muitas aplicações, como o monitoramento do meio ambiente e do tráfego, dependem desta informação. Além disso, conforme (HU; EVANS, 2004) e (BACHIR et al., 2010), o conhecimento da localização também pode ser importante para a economia de energia, no roteamento de pacotes e para aumentar a segurança da rede.

1.2 Objetivos

Este trabalho possui como o objetivo principal o desenvolvimento de um método de multilateração iterativo de baixa complexidade para algoritmos de localização em RSSF. A métrica utilizada para avaliar e comparar o desempenho deste método com outros da literatura é o erro entre a posição real e a estimada pela multilateração, denominado erro de localização, em ambientes com medidas de distâncias inexatas entre nodos âncora e desconhecidos.

Paralelamente, desenvolveu-se um simulador baseado no software Matlab, específico para algoritmos de localização e que permitirá a avaliação do método de multilateração criado em diferentes algoritmos e cenários. Como principais características do simulador estão a implementação do modelo de propagação log-normal com sombreamento e o suporte a co-simulação de linguagem de descrição de hardware.

1.3 Contribuições do trabalho

Pode-se destacar neste trabalho duas contribuições principais, sendo a primeira a criação de um método de multilateração iterativo de baixa complexidade computacional e com erro de localização próximo ao ideal. A segunda foi o desenvolvimento de um simulador otimizado para a simulação de algoritmos de localização em RSSF em uma linguagem de fácil compreensão e com muitos recursos, permitindo uma rápida criação de cenários customizados e modelamento de novos algoritmos. Ainda para o simulador, destaca-se que ele possui suporte a co-simulação de linguagem de descrição de hardware, acelerando assim o processo de correção de erros de algoritmos que serão implementados em circuito reconfigurável ou integrado.

1.4 Organização do trabalho

Esta dissertação está organizada em 5 capítulos. O capítulo seguinte apresenta uma revisão dos principais conceitos necessários para a compreensão do restante trabalho. No Capítulo 3 é detalhado o desenvolvimento do método de multilateração iterativo, bem como sua avaliação e comparação em determinados cenários. Já no Capítulo 4 são descritos os principais aspectos do simulador construído especificamente para algoritmos de localização em RSSF, e a sua utilização para comparação deste método de multilateração com outros da literatura em cenários com disposição aleatórias dos nodos de referência. Por fim, no Capítulo 5 são sumarizadas as principais contribuições desta dissertação e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os principais conceitos necessários para acompanhar o desenvolvimento do método de multilateração e do simulador para algoritmos de localização em RSSF. Assim, dividiu-se este em quatro seções, nas quais são abordadas as características gerais, algoritmos de localização, métodos de multilateração e simuladores para RSSF.

2.1 Características gerais das RSSF

Uma RSSF consiste basicamente de uma rede formada por nodos organizados de tal maneira a desempenhar determinada tarefa como vigilância, coleta de dados, rastreamento, entre outras. Uma organização clássica para este tipo de rede é vista na Figura 2.1, na qual os dados são coletados e roteados pelos nodos até o *sink*, e então transmitidos por este para um centro de controle. Como exemplo de uma RSSF real, pode-se citar a GreenOrbs, conforme (GREENORBS, 2011), criada para coletar dados de uma floresta, incluindo temperatura, umidade, iluminação e concentração de dióxido de carbono.

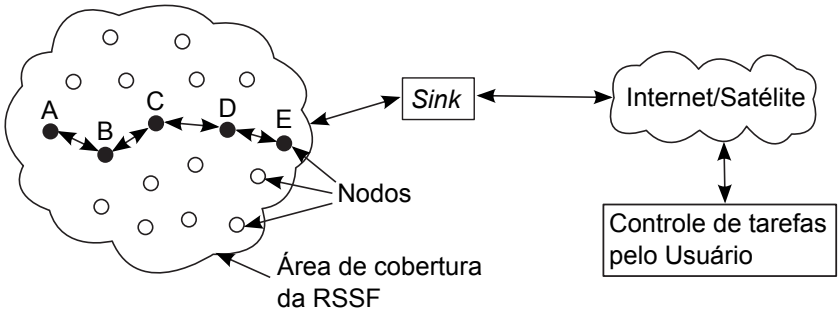


Figura 2.1: Nodos espalhados numa determinada área (adaptado de (AKYILDIZ et al., 2002)).

A arquitetura básica de um nodo é mostrada na Figura 2.2, podendo-se dividi-la em quatro unidades principais, sendo elas a de sensoriamento, processamento, comunicação e fornecimento de energia. Como os nodos são sistemas autônomos, geralmente alimentados por baterias, uma das maiores preocupações das RSSF é o controle do consumo de potência, conforme (PANTAZIS; VERGADOS, 2007).

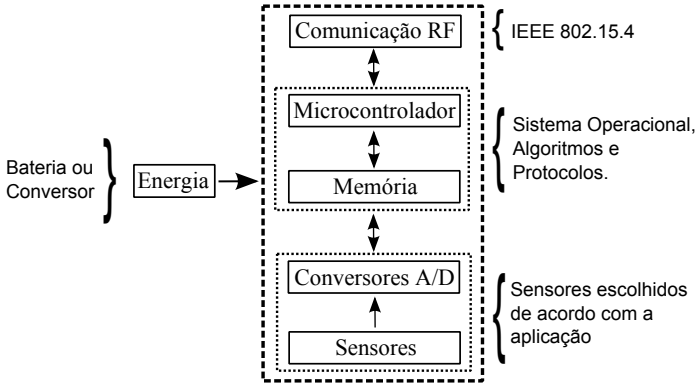


Figura 2.2: Componentes de um nodo para RSSF.

2.2 Algoritmos de localização para RSSF

O conhecimento da localização dos nodos em uma RSSF, além de ser necessário para muitas aplicações, também pode ser importante para a economia de energia, o roteamento de pacotes e o aumento da segurança da rede, conforme (HU; EVANS, 2004). O Sistema de Posicionamento Global (GPS), conforme (B. HOFMANN-WELLENHOF; COLLINS, 1997), é uma aplicação largamente utilizada para localização em escala global, porém, conforme (ZHAO et al., 2013), falha ao ser utilizada em regiões encobertas como em edificações, abaixo do solo ou em matas densas. Além disso, conforme já destacado por (BULUSU; HEIDEMANN; ESTRIN, 2000), o uso destes dispositivos acarreta num considerável aumento do preço e do consumo de energia dos nodos, o que se torna inviável em larga escala.

Os algoritmos de localização podem ser divididos em duas categorias principais, conforme (OLIVEIRA, 2009), sendo a primeira composta pelos que são baseados na medição das distâncias entre os nodos e a segunda apenas na conectividade entre eles, denominados de *range-based* e *range-free* respectivamente. Na categoria dos *range-based* podem ser utilizados os princípios medição do ângulo, tempo e intensidade de determinado sinal recebido, sendo que este pode ser RF, acústico, infravermelho ou ultrassom.

O método de medição do ângulo de chegada (AOA do inglês *Angle of Arrival*) consiste na determinação do ângulo de chegada de um sinal, cuja posição da fonte é conhecida ou informada através desta mesma transmissão, segundo (NICULESCU; NATH, 2003). Então, com no mínimo dois transmissores de referência não colineares, denominados de nodos âncora, é possível determinar a posição de um nodo desconhecido através de relações geométricas, conforme visto na Figura 2.3. Como principal desvantagem deste método tem-se a necessidade de

um hardware especializado para determinação do ângulo de chegada do sinal, sendo este um conjunto de antenas direcionais, além de ter problemas de imprecisão nas medidas devido a efeitos como reflexão, difração e dispersão do sinal propagado.

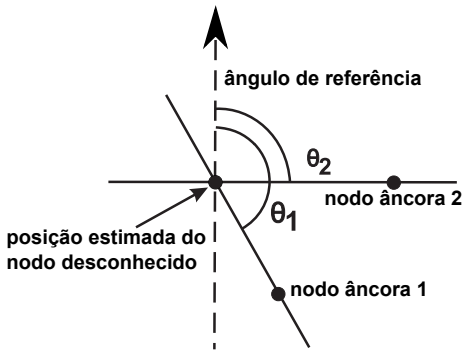


Figura 2.3: Localização de um nodo desconhecido pelo método AOA (adaptado de (OLIVEIRA, 2009)).

Os métodos baseados na medição do tempo de propagação de um sinal podem ser os de tempo de chegada (TOA do inglês *Time of Arrival*) e o tempo de diferença de chegada (TDOA do inglês *Time Difference of Arrival*) de acordo com (B. HOFMANN-WELLENHOF; COLLINS, 1997) e (GUSTAFSSON; GUNNARSSON, 2003), respectivamente. No primeiro método, estima-se a distância entre o nodo desconhecido e o âncora através da duração e velocidade de propagação do sinal. Após obterem-se as distâncias de no mínimo três nodos âncora, num espaço de duas dimensões, encontra-se a localização do nodo desconhecido através de um cálculo de trilateração, sendo este explicada em detalhes na Seção 2.3.

Já na TDOA, através da diferença dos tempos de chegada de sinais transmitidos ao mesmo tempo por dois nodos âncora perfeitamente sincronizados em um nodo desconhecido, é possível criar uma hipérbole sobre a qual estima-se que este está localizado. Esta hipérbole caracteriza-se por manter a diferença de distâncias entre os nodos âncora constante e igual à diferença dos tempos de chegada multiplicado pela velocidade (v) de propagação do sinal, conforme mostrado na Figura 2.4 e respeitando (2.1). Então, repetindo-se este processo para uma ou mais combinações diferentes de nodos âncora encontram-se novas hipérboles, cujas intersecções entre si e com a primeira resultarão em estimativas de localização do nodo desconhecido, conforme mostrado nesta mesma figura.

$$v\delta t_{12} = \sqrt{(x_1 - u_x)^2 + (y_1 - u_y)^2} - \sqrt{(x_2 - u_x)^2 + (y_2 - u_y)^2} \tag{2.1}$$

O principal problema dos métodos baseados no tempo de propagação do sinal é o sin-

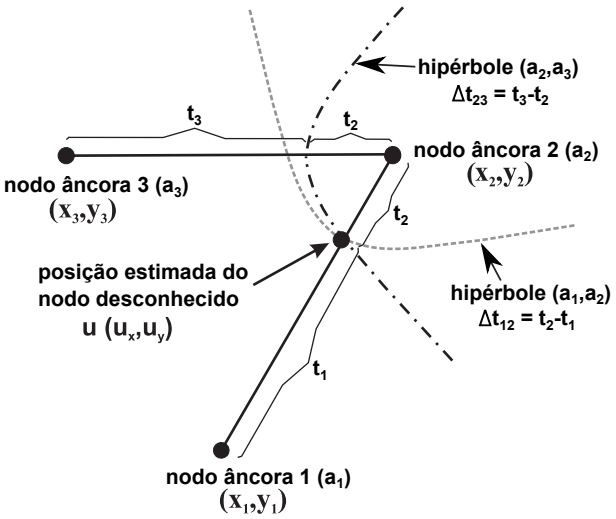


Figura 2.4: Localização de um nodo desconhecido pelo método TDOA (adaptado de (OLIVEIRA, 2009)).

cronismo necessário entre os nodos. Isto pode ser evidenciado para o caso da medição do tempo de propagação de um sinal RF, no qual com a velocidade de propagação de aproximadamente $300.000Km/s$, ou seja, em $3,3ns$ viaja-se $1m$. Logo, considerando que o relógio do sistema do nodo seja $10MHz$, para o atraso de apenas um ciclo deste obtém-se um erro de $30m$ na estimativa de distância utilizada no método TOA. Por este motivo, muitos trabalhos utilizam sinais de áudio que se propagam com velocidades muito menores, como é o caso de (PENG; SHEN; ZHANG, 2012), porém, retorna o problema da necessidade de um hardware especializado do método AOA.

O último método *range-based* citado neste trabalho é o baseado na potência do sinal recebido (RSSI do inglês *Received Signal Strength Indicator*), conforme (BAHL; PADMANABHAN, 2000). Através de modelos de propagação do sinal RF é possível definir uma distância entre o transmissor e o receptor de acordo com a amplitude do sinal recebido, conforme (RAPPAPORT, 2001). Então, ao obterem-se estimativas de distâncias de no mínimo três nodos âncora é possível localizar um nodo desconhecido num espaço de duas dimensões através da trilateração, semelhante ao visto no método TOA.

Como principal problema deste método destaca-se a imprecisão nas estimativas devido à natureza da propagação do sinal em RF, sofrendo uma série de variações devido a efeitos como reflexão, difração, interferência e falta de visada direta. Porém, como a medição do RSSI está previsa pelo padrão IEEE 802.15.4, segundo (IEEE, 2003) (apresentado na Seção 2.4.1), que é utilizado em praticamente todos os módulos de comunicação para RSSF, este é o método

range-based com o menor custo de implementação, pois não necessita de hardware extra ou sincronismo entre nodos.

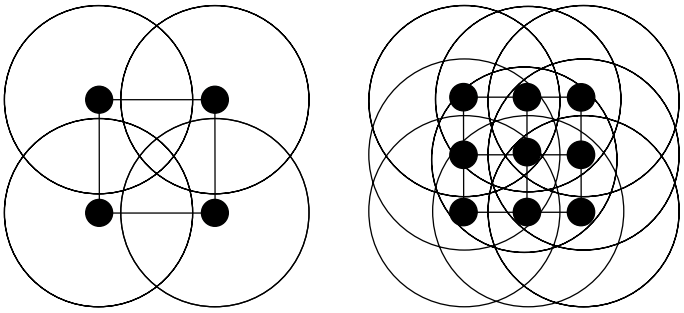
Um exemplo de módulo de comunicação com este padrão é o circuito de rádio modelo CC2420 utilizado no nodo comercial MICAz , (Texas Instruments, 2012) e (CROSSBOW, 2013). Este padrão foi desenvolvido pelo Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE do inglês *Institute of Electric and Electronic Engineers*) para regulamentar as redes de comunicação sem fio com dispositivos de baixa taxa de transmissão, baixo custo e com alcance limitado, sendo estas as principais características dos nodos.

Já os algoritmos *range-free* consistem basicamente em estimar a localização dos nodos apenas com a troca de mensagens, tendo como única noção de distância a de que estão ou não dentro do raio de alcance de outros nodos. As principais características desta categoria são a simplicidade de implementação e, tendo como consequência, erros de localização maiores que os com os métodos *range-based*. Neste trabalho são apresentados os algoritmos denominados Centroid e o DV-Hop, conforme (BULUSU; HEIDEMANN; ESTRIN, 2000) e (NICULESCU; NATH, 2001) respectivamente.

No algoritmo Centroid, inicialmente distribuem-se os nodos âncora de maneira uniforme pelo cenário, os quais realizam transmissões periódicas para todos os nodos que estão no seu raio de alcance, sendo este período conhecido por todos na rede. Este tipo de transmissão é denominada de *broadcast* e para este algoritmo o único conteúdo da mensagem é a posição dos nodos âncora que as estão transmitindo. Então, os nodos desconhecidos irão receber e contabilizar estas mensagens criando uma avaliação chamada de métrica de conexão, a qual resulta da divisão do total de mensagens recebidas pelo total que deveriam ser transmitidas. Por fim, o nodo desconhecido irá gerar a sua estimativa de localização como sendo a média geométrica das posições dos âncoras cujas métricas de conexão atingiram determinado valor, sendo este de $0,9$ ou 90% em (BULUSU; HEIDEMANN; ESTRIN, 2000).

Ainda, segundo (BULUSU; HEIDEMANN; ESTRIN, 2000), a precisão do algoritmo de localização Centroid é diretamente dependente da densidade de nodos âncora do cenário, conforme visto nas Figuras 2.5(a) e 2.5(b). Isto se deve ao fato de que o aumento desta densidade também aumenta a quantidade de possíveis localizações estimadas, de forma que a média das posições dos nodos âncora se torne cada vez mais próxima da posição real do nodo desconhecido. Assim, como principal problema do algoritmo Centroid destaca-se a dependência de uma distribuição uniforme e com alta densidade de nodos âncora para se obter uma precisão

razoável das estimativas de localização.



(a) Distribuição regular de 4 nodos âncora.

(b) Distribuição regular de 9 nodos âncora.

Figura 2.5: Aumento da densidade de nodos âncora mantendo o alcance de transmissão (adaptado de (BULUSU; HEIDEMANN; ESTRIN, 2000)).

Em (REICHENBACH; BLUMENTHAL; TIMMERMANN, 2006), definiu-se uma relação que maximiza o custo benefício entre o raio de alcance dos nodos e o erro de localização, visando uma redução considerável do consumo de energia da rede através do correto ajuste da potência de transmissão. Segundo este, considera-se como ideal para o algoritmo Centroid um raio de alcance de 86% da distância lateral entre dois nodos âncora, semelhante ao visto na Figura 2.5(a).

Já o algoritmo de localização DV-Hop foi desenvolvido para uma RSSF com densidade de nodos âncora muito baixa, de modo que uma grande parcela dos nodos desconhecidos não esteja no raio de alcance destes. Este algoritmo caracteriza-se pela contagem dos saltos que uma determinada mensagem sofre para ser transmitida através da rede, de um nó âncora a outro, podendo-se dividi-lo em 3 etapas: determinação do fator de correção, contagem de saltos e trilateração.

Na primeira etapa, cada nodo âncora inicia um contador de saltos, to inglês *hops*, que é propagado pela rede de modo que quando um outro nodo o receba, este o incrementa e retransmite, sendo que estas mensagem também possuem a posição dos âncoras que iniciaram a contagem. Como as antenas dos nodos são omnidirecionais, idealmente deveriam transmitir com mesma potência em todas as direções, estes recebem vários contadores provenientes de um mesmo âncora com diferentes valores, devendo apenas incrementar e retransmitir os que possuem uma contagem menor que a menor já recebida. Ao término destas propagações, todos os âncoras criam uma tabela contendo a posição e o número de saltos para cada um dos outros nodos de referência do cenário. Então, cada um destes nodos divide o somatório das distâncias

entre si e os outros âncoras pelo somatório dos saltos, gerando assim seus fatores de correção. Por exemplo, de acordo com a Figura 2.6, o cálculo do fator de correção do nodo âncora A1 é dado por

$$\frac{40m_{A2} + 100m_{A3}}{2hops_{A2} + 6hops_{A3}} = 17,5m/hop. \tag{2.2}$$

Assim, a segunda etapa consiste de uma nova propagação de contadores, diferenciando-se da primeira apenas por acrescentar ao conteúdo das mensagens o fator de correção calculado por cada nó âncora, permitindo que todos os nodos desconhecidos construam uma tabela semelhante à dos âncoras, porém, contendo esses fatores a mais. Esta segunda etapa é exemplificada pela Figura 2.6, na qual destacam-se os caminhos (identificados pelas linhas tracejadas) com os menores números de saltos entre o nodo desconhecido (identificado pela letra D) e os âncoras (identificados por A1, A2 e A3).

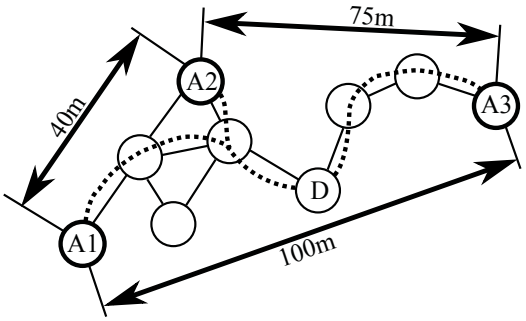


Figura 2.6: Localização de um nodo desconhecido através do método DV-Hop (adaptado de (NICULESCU; NATH, 2001)).

Por fim, na terceira etapa os nodos desconhecidos geram estimativas de distâncias multiplicando o número de saltos pelo fator de correção referente ao âncora mais próximo, ou seja, com o menor número de saltos. Como exemplo, pode-se citar que na Figura 2.6 é utilizado o fator de correção do nó âncora A2 para localizar o nó desconhecido D. Então, após estimar as distâncias em relação a cada um dos âncoras realiza-se uma trilateração para se obter a posição estimada do nodo desconhecido.

Como o algoritmo DV-Hop é utilizado em redes com disposições aleatórias dos nodos, as estimativas de distâncias através dos saltos podem possuir erros grosseiros. Tendo isso em vista, (NICULESCU; NATH, 2001) sugere que as distâncias entre nodos sejam estimadas através de algum método *range-based*, melhorando assim a precisão. Outra possibilidade é apresentada em (SAVARESE; RABAEY; LANGENDOEN, 2002), onde utiliza-se o algoritmo

DV-Hop para se criar estimativas iniciais da localização dos nodos, com as quais realiza-se uma fase de refinamento destas através da troca de mensagens, contendo estas estimativas, entre nodos vizinhos.

Finalizada a apresentação de alguns dos algoritmos de localização *range-based* e *range-free* mais importantes da literatura, destaca-se que a maioria deles foram originalmente projetados para RSSFs estáticas, ou seja, não consideraram a mobilidade tanto dos nodos âncora quanto dos desconhecidos. O estudo de algoritmos para estes tipos de redes não será abordado neste trabalho devido à complexidade que a mobilidade agrega a uma RSSF.

2.3 Multilateração aplicada a RSSF

A multilateração aplicada a redes de sensores sem fio consiste no processo de determinação da localização de um nodo desconhecido a partir de distâncias medidas entre ele e nodos de referência, denominados âncoras. Para localizar um nodo num espaço de 2 ou 3 dimensões (2D ou 3D) são necessários no mínimo 3 ou 4 nodos âncora, respectivamente. Na Figura 2.7 é apresentado um exemplo de trilateração (multilateração em que existem apenas 3 pontos de referência) em que as distâncias entre os nodos são conhecidas precisamente.

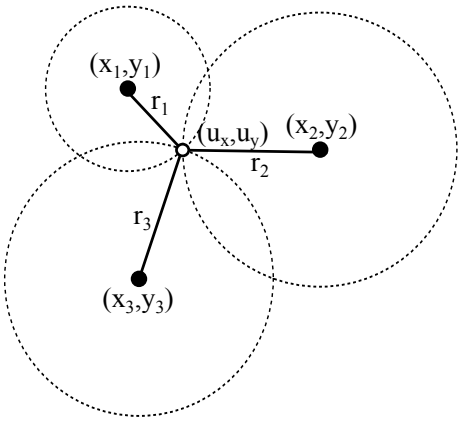


Figura 2.7: Trilateração com medição precisa das distâncias (r_n) entre o nodo desconhecido (u_x, u_y) e os âncoras (x_n, y_n).

Considerando um cenário em que se deseja localizar um nodo desconhecido em 2D, semelhante ao da Figura 2.7, com um total de n âncoras, obtêm-se conjunto de equações

$$\begin{bmatrix} (x_1 - u_x)^2 + (y_1 - u_y)^2 \\ (x_2 - u_x)^2 + (y_2 - u_y)^2 \\ \vdots \\ (x_n - u_x)^2 + (y_n - u_y)^2 \end{bmatrix} = \begin{bmatrix} {r_1}^2 \\ {r_2}^2 \\ \vdots \\ {r_n}^2 \end{bmatrix}. \tag{2.3}$$

Este sistema de equações não lineares pode ser transformado em um linear subtraindo-se a primeira equação das restantes, eliminando-se assim os termos ${u_x}^2$ e ${u_y}^2$ semelhante a (KARALAR et al., 2004), porém para um sistema em 2D ao invés de 3D. Como resultado, têm-se $n - 1$ equações, que podem ser representadas por um sistema do tipo

$$Au = b, \tag{2.4}$$

sendo

$$A = \begin{bmatrix} (x_1 - x_2)^2 & (y_1 - y_2)^2 \\ \vdots & \vdots \\ (x_1 - x_n)^2 & (y_1 - y_n)^2 \end{bmatrix}, \tag{2.5}$$

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \tag{2.6}$$

e

$$b = 0,5 \times \begin{bmatrix} {r_2}^2 - {r_1}^2 - {x_2}^2 + {x_1}^2 - {y_2}^2 + {y_1}^2 \\ \vdots \\ {r_n}^2 - {r_1}^2 - {x_n}^2 + {x_1}^2 - {y_n}^2 + {y_1}^2 \end{bmatrix}. \tag{2.7}$$

Já em (SAVVIDES; HAN; STRIVASTAVA, 2001) e (CHEN; CHIN; HUANG, 2012), a linearização ocorre subtraindo-se a última equação das restantes, mesmo assim produzindo o mesmo resultado das equações utilizadas neste trabalho.

Este sistema de equações é sobredeterminado para $n > 3$ em um espaço 2D, sendo que a variável u pode ser determinada resolvendo-se

$$u = \left(A^T A\right)^{-1} A^T b \tag{2.8}$$

pelo método de mínimos quadrados. Porém, devido à complexidade computacional para se inverter uma matriz, geralmente utilizam-se métodos por decomposição QR para simplificar a resolução do sistema (2.4) (maiores detalhes no Apêndice A).

Por outro lado, em uma RSSF, as distâncias medidas entre os nodos desconhecidos e os âncoras raramente serão precisas. Nesses casos, a multilateração é representada pela Figura 2.8, onde não existe a interseção exata, em um único ponto, entre as distâncias estimadas em

relação aos âncoras mas sim uma região, modificando assim (2.3) para

$$\begin{bmatrix} (x_1-u_x)^2+(y_1-u_y)^2 \\ (x_2-u_x)^2+(y_2-u_y)^2 \\ \vdots \\ (x_n-u_x)^2+(y_n-u_y)^2 \end{bmatrix} = \begin{bmatrix} (r_1e_1)^2 \\ (r_2e_2)^2 \\ \vdots \\ (r_ne_n)^2 \end{bmatrix}. \tag{2.9}$$

Esta equação possui uma abordagem de erro multiplicativo (r_1e_1) ao invés de aditivo (r_1+e_1), conforme (PATWARI et al., 2005), voltada às estimativas de distâncias baseadas na potência do sinal recebido.

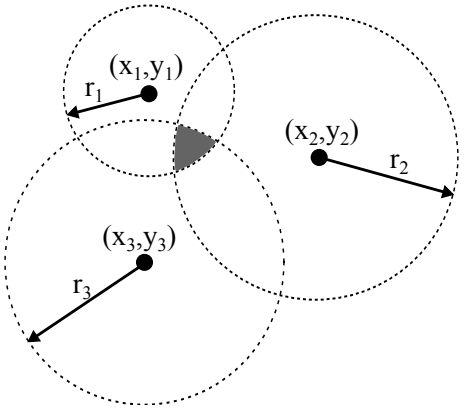


Figura 2.8: Trilateração com erro na medição das distâncias.

Ignorando-se o erro e_i de (2.9), pode-se linearizar o sistema e resolvê-lo pelo método de mínimos quadrados, sendo este denominado mínimos quadrados linearizado (MQL). Porém, a resolução por máxima verossimilhança (MV) é a mais adequada para este tipo de multilateração, que tem por objetivo encontrar a melhor posição para o nodo desconhecido $(\hat{x},\hat{y})$ de acordo com

$$(\hat{x},\hat{y}) = arg\min_{x,y}\{f(x,y)\}, \tag{2.10}$$

ou seja, minimizando determinada função custo $f(x,y)$. Em (YEDAVALLI, 2002) a função utilizada foi a do residual quadrático

$$f(x,y) = \sum_{i=1}^{n}\left(r_i - \sqrt{(x-x_i)^2+(y-y_i)^2}\right)^2, \tag{2.11}$$

que consiste no somatório dos quadrados das diferenças entre as distâncias medidas (r_i) e as reais do nodo desconhecido (x,y) em relação aos âncoras (x_i,y_i).

A forma mais segura para resolver (2.10), conforme implementado em (YEDAVALLI; KRISHNAMACHARI, 2009), consiste em criar uma grade de pontos (isto para um cenário

2D) sobre toda a área de interesse, executar (2.11) para todos eles, e então escolher o ponto que apresente o menor residual quadrático (MRQ). No entanto, além deste ser o processo mais custoso em termos computacionais, ainda insere um erro relacionado ao passo da grade de pontos utilizada. Este último representa a distância nos eixos x e y entre os consecutivos pontos da grade, por exemplo, em (YEDAVALLI; KRISHNAMACHARI, 2009) utilizou-se 1% da largura do cenário. Assim, quanto menor for este passo, maior será o número de pontos para os quais deverão ser calculados os residuais quadráticos, e, consequentemente, menor será a diferença entre o MRQ calculado e o real, sendo que este último considera um passo tendendo a zero.

Existem métodos iterativos que, através da propriedade do gradiente, convergem para o ponto de MRQ do cenário com um custo computacional menor que o da grade, sendo um exemplo apresentado em (YEDAVALLI, 2002). Porém, estes métodos necessitam de um ponto inicial, cuja escolha equivocada pode fazer o sistema convergir para um ponto de mínimo local e não o mínimo global esperado. Segundo (LI et al., 2005), o resultado do método MQL é um bom ponto de partida para estes métodos iterativos.

Como métrica para avaliar o desempenho da multilateração, será utilizado o erro de localização (EL). Este consiste em calcular a distância euclidiana, dada por

$$EL = \sqrt{(\hat{x} - x_r)^2 + (\hat{y} - y_r)^2}, \tag{2.12}$$

entre a posição real (x_r, y_r) e a calculada $(\hat{x}, \hat{y})$ ou (u_x, u_y) do nodo desconhecido. Esta última também pode ser chamada de posição estimada, pois o objetivo deste trabalho é otimizar métodos matemáticos para utilizá-los no processo de estimação da localização de nodos desconhecidos.

Conforme (LI et al., 2005), os erros de localização dos métodos MQL e MV aumentam proporcionalmente em relação ao desvio padrão do erro e_i das medidas utilizadas na multilateração, de modo que quanto maior for este, mais vantajosa será a utilização do método por MV em relação ao por MQL. Por outro lado, o método por MV é mais complexo, resultando em um maior consumo de recursos computacionais e de energia dos nodos.

Para compreender melhor o comportamento dos métodos descritos acima em ambientes com erros nas medições, decidiu-se realizar uma simulação através do software Matlab (MATHWORKS, 2012). Criou-se o cenário apresentado na Figura 2.9, onde em uma área com dimensões de $100m \times 100m$ distribui-se um total de 4 nodos âncora numerados e posicionados nos vértices deste quadrado. O restante dos pontos representam 2597 nodos desconhecidos

posicionados em forma de grade com espaçamento vertical e horizontal igual a 2m.

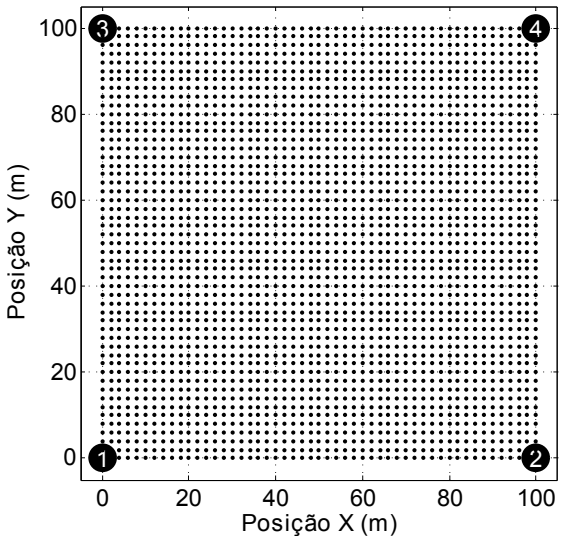


Figura 2.9: Cenário simulado para comparar o erro de localização entre os métodos MQL e MV na presença de erros nas medidas.

Os erros inseridos nas distâncias entre nodos desconhecidos e âncoras foram gerados aleatoriamente, a partir da função *randn* do Matlab, seguindo uma distribuição normal. Essa inserção de erros ocorre em modo percentual, ou seja, ao dizer-se que há um erro de 10% positivo em determinada distância, entende-se que esta é 10% maior do que deveria ser. Esta abordagem é semelhante à vista em (PATWARI et al., 2005), voltada a estimativas de distâncias baseadas na potência do sinal recebido.

Nesta simulação, cada nodo desconhecido deve calcular a sua posição através dos métodos MQL e MV, sendo que este último é executado através da função *fminsearch* do Matlab e utiliza como ponto inicial o resultado do primeiro. Considerou-se que os nodos desconhecidos recebem as informações de distância e posicionamento de todos os âncoras, sendo o resultado da simulação apresentado na Figura 2.10. Nesta simulação, o eixo horizontal representa o aumento do desvio padrão percentual dos erros aleatórios inseridos nas distâncias utilizadas pelos dois métodos de multilateração. O eixo vertical representa a média do erro de localização em metros de todos os nodos desconhecidos do cenário. Percebe-se que o método por MV possui melhor desempenho neste aspecto.

Já a Figura 2.11 mostra o resultado da simulação de apenas um nodo desconhecido posicionado exatamente no centro, posição real (PR) igual a $[50, 50]$, do cenário visto na Figura 2.9. Nesta simulação, configurou-se o desvio padrão do erro percentual adicionado nas medidas como 10%, que é o pior caso de imprecisão visto na Figura 2.10. Como resultado desta

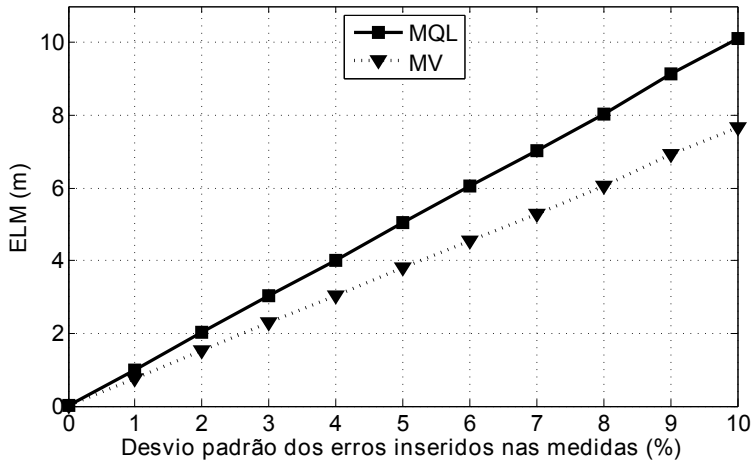


Figura 2.10: Comparação do erro de localização médio entre os métodos MQL e MV na presença de medidas imprecisas.

imprecisão, percebe-se que os semi-círculos tracejados, que indicam o alcance das distâncias em relação a cada nodo âncora, não possuem um ponto em comum. Conforme o esperado, a multilateração executada pelo método MV possui um erro de localização menor que o MQL, $4,3m$ contra $6,6m$.

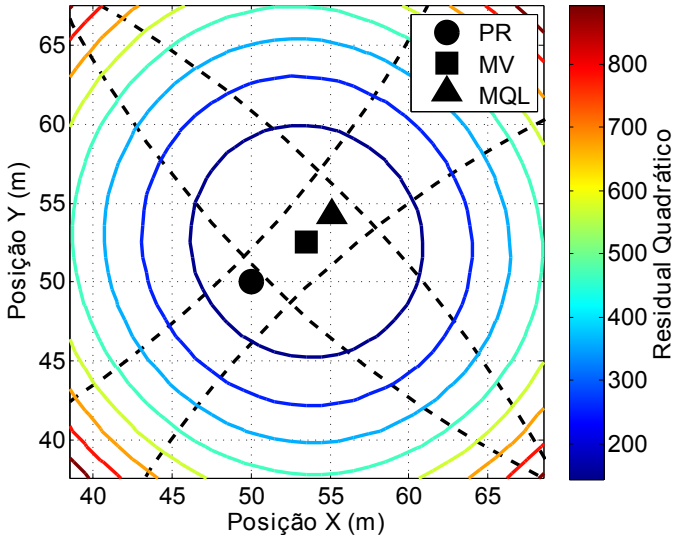


Figura 2.11: Multilateração executada pelos métodos MV e MQL considerando um desvio padrão do erro percentual adicionado nas medidas de 10%.

2.4 Simuladores para RSSF

Os simuladores são fundamentais para o estudo de RSSF, conforme (EGEA-LOPEZ et al., 2006), pois a criação de redes reais pode ser muito cara, complexa e muitas vezes inviável

de ser implementada para fins de estudo. Outro aspecto importante na utilização dos simuladores, segundo (LEVIS et al., 2003), é a possibilidade do estudo de como características isoladas das redes e do ambiente influenciam no desempenho da RSSF como um todo, o que também é difícil de ser realizado na prática.

Neste trabalho não se deseja realizar um comparativo detalhado entre os principais simuladores existentes na literatura por dois motivos principais. O primeiro deles é que já existem trabalhos relacionados a revisões dos principais simuladores, metodologias para a escolha de um deles e como se avaliar a coerência de seus resultados, sendo alguns destes (ORFANUS et al., 2008), (KORKALAINEN et al., 2009), (IMRAN; SAID; HASBULLAH, 2010) e (KHAN et al., 2011), (STEHLIK, 2011). Destaca-se este último por ser uma dissertação de mestrado especifica sobre comparação de simuladores entre si e com resultados de experimentos práticos, além de realizar um revisão bibliográfica que permite compreender a complexidade do aprendizado destes simuladores, bem como suas principais características.

Já o segundo motivo é que o objetivo principal deste trabalho é a avaliação do erro de localização do método de multilateração desenvolvido no Capítulo 3 numa RSSF distribuída aleatoriamente, tarefa que será realizada com um menor esforço através da criação de um simulador simples baseado no software Matlab. Desta maneira, esta seção tem por objetivo principal a compreensão dos principais conceitos adotados na criação deste simulador, bem como os motivos que nos levaram a esta escolha.

Alguns dos principais simuladores genéricos, não desenvolvidos para um determinado modelo de nodo fabricado, para RSSF utilizados na literatura, seguidos de suas respectivas referências, são o NS-2 (NS-2, 2012), OMNet++ (OMNET++, 2012), Worldsens (FRABOULET; CHELIUS; FLEURY, 2007) e OPNET (OPNET, 2012), sendo que este último não é software livre. Como principais características destes destacam-se que são codificados em C ou C++, possuem interface gráfica e uma série de modelos, padrões e algoritmos disponíveis. Outra característica importante destes simuladores é que são baseados em eventos, ou seja, a simulação não ocorre em passos temporais constantes, mas sim de acordo com o agendamento de atividades como, por exemplo, transmissão e recepção de mensagens ou realização de determinado sensoreamento ou processamento. Esta é uma das principais características que viabilizam a simulação de redes com até alguns milhares de nodos e grande precisão em tempos de execução aceitáveis.

Os emuladores, por sua vez, diferenciam-se dos simuladores por imitarem a um hard-

ware e/ou sistema operacional específico, conforme (KIESS; MAUVE, 2007), geralmente utilizados para avaliar o comportamento de determinado algoritmo ou protocolo num produto específico. Como exemplo, pode-se citar o PowerTOSSIM visto em (PERLA et al., 2008), que visa executar o sistema operacional TinyOS, conforme (TINYOS, 2012), sobre uma representação fiel do hardware do nodo comercial MICAz, conforme (CROSSBOW, 2013). Assim, uma das principais vantagens dos emuladores em relação aos simuladores é a precisão na estimativa de potência consumida pelo nodo, porém ao custo de um aumento considerável do tempo de execução.

Conforme (HASAN et al., 2009), ao contrário de C ou C++ a linguagem Matlab não é otimizada para codificação de algoritmos baseados no agendamento de eventos, como é o caso dos simuladores e emuladores para RSSF. Assim, simuladores como o Prowler, apresentado em (SIMON et al., 2003), possuem um tempo de execução elevado se comparado aos outros simuladores codificados em C ou C++.

Porém, conforme (ALI; ABDULMAOWJOD; MOHAMMED, 2010), o ambiente Matlab/Simulink fornece um rico ambiente gráfico com uma série de bibliotecas e uma linguagem simples que facilitam o projeto e simulação de sistemas variantes no tempo, os quais incluem comunicações, controle, processamento de sinais, entre outros. Por isso, (ALI; ABDULMAOWJOD; MOHAMMED, 2010) utilizou este ambiente para a simulação de uma RSSF focando-se na avaliação dos efeitos causados na rede pela variação de diferentes parâmetros do canal de comunicação, como SNR (do inglês *Signal-to-Noise Ratio*), atenuação e interferência. Já (MOZUMDAR et al., 2008), escolheu utilizar os softwares Matlab/Simulink para a criação de um simulador para RSSF pela capacidade de geração automática de códigos compatíveis com nodos comerciais a partir de fluxogramas.

De modo a tentar unir as características positivas dos simuladores de RSSF baseados em eventos com a capacidade de modelagem de sistemas reais baseados no tempo através do Matlab/Simulink, surgem os trabalhos de co-simulação. Entende-se por co-simulação a realização de uma simulação de um sistema em mais de um software, de modo que cada um deles seja utilizado para o que é otimizado. Como primeiros exemplos podem ser citados (KOHTAMAKI et al., 2009) e (HASAN et al., 2009) projetados para o trabalho com as chamadas WNCS (do inglês *Wireless Networked Control Systems*), nas quais deseja-se realizar determinado controle industrial tendo os sensores e atuadores conectados através de uma rede de comunicação sem fio. Em ambos utilizou-se o ambiente Matlab/Simulink para o desenvolvimento do sistema de

controle, conectado ao simulador da RSSF através de uma conexão de rede, ou seja, são executados em computadores diferentes e sincronizados através da troca mensagens pela interface de rede no formado de soquetes UDP (do inglês *User Datagram Protocol*). Como principal diferença entre estes trabalhos destaca-se que em (KOHTAMAKI et al., 2009) utiliza-se o NS-2 e em (HASAN et al., 2009) o OPNET, para a simulação da RSSF.

De modo semelhante, (ZHANG et al., 2010) e (DIDIOUI et al., 2013) realizam co-simulação entre o Matlab/Simulink com os simuladores de RSSF OMNet++ e WSNet (parte do simulador Worldsens), respectivamente. No primeiro utiliza-se o ambiente Matlab/Simulink para modelar o canal de comunicação sem fio em três dimensões, enquanto que no segundo um sistema para aumentar a duração da bateria através da captação de energia solar. A principal diferença destas duas co-simulações em relação as vistas no parágrafo anterior é a de que ambos os softwares são executados paralelamente num mesmo computador, apesar do método de sincronismo ser o mesmo, facilitando a sua utilização.

Percebendo-se as vantagens e facilidades que o ambiente Matlab/Simulink fornece, atrelado ao fato de não se desejar a simulação detalhada em termos de camadas de rede como a encontrada na maioria dos simuladores da literatura, decidiu-se pela utilização apenas deste software para a avaliação do método de multilateração desenvolvido no Capítulo 3. Esta decisão fundamenta-se em dois motivos principais, sendo o primeiro deles o fato de que não basta apenas escolher um simulador da literatura, pois podem haver grandes diferenças entre os resultados gerados por estes, conforme ressaltado por (ANDEL; YASINSAC, 2006), sendo necessária uma escolha criteriosa, baseada no estudo das características e utilização dos simuladores, o que consome um tempo considerável.

O segundo motivo é que se deseja realizar uma prova de conceito de que o método de multilateração desenvolvido neste trabalho possui um erro de localização menor que os outros encontrados na literatura para uma determinada RSSF. Conforme destacado por (STOJMENOVIC, 2008), esta prova deve ser baseada em uma simulação simples, na qual desconsideram-se quaisquer fatores indesejados que venham a influenciar no resultado, ou seja, deve-se estudar uma variável de cada vez, permitindo que fiquem claros os resultados. Como exemplo desta questão pode-se citar o trabalho desenvolvido em (YEDAVALLI; KRISHNAMACHARI, 2009), no qual avaliou-se o desempenho de determinado algoritmo de localização desconsiderando a colisão de mensagens transmitidas entre os nodos, o que também será considerado neste trabalho.

Então, para criar o simulador utilizando o ambiente Matalb/Simulink, apresentado no Capítulo 4, para avaliação do erro de localização em algoritmos de localização em RSSF serão levados em conta características do padrão IEEE802.15.4 e o modelo de propagação log-normal. Ambos são implementados na maioria dos simuladores para RSSF encontrados na literatura, sendo estes explicados no restante desta seção.

2.4.1 Padão IEEE 802.15.4

O padão IEEE 802.15.4, conforme (IEEE, 2003), foi criado com a finalidade de definir as camadas físicas (PHY do inglês *Physical Layer*) e de acesso (MAC do inglês *Medium Access Control*) em redes de comunicação sem fio, denominadas WPAN (*Wireless Personal Area Networks*), com baixa taxa de transmissão (LR-WPANs do ingês *Low-Rate WPANs*). Este padrão é direcionado a dispositivos que possuem baixa capacidades de fornecimento de energia, processamento e alcance muito limitados, sendo estas as principais características dos nodos constituintes das RSSF. Algumas das principais características deste padrão são:

- A alocação de 16, 10 e 1 canais nas bandas de $2450MHz$, $915MHz$ e $868MHz$, respectivamente, com as respectivas taxas de transmissão de $250kb/s$, $40kb/s$ e $20kb/s$;
- Operação em modo estrela ou ponto-a-ponto, conforme a Figura 2.12, nos quais os nodos só se comunicam apenas com um nó central coordenador ou com quem quiserem, respectivamente;
- Evitar conflitos para múltiplos acessos ao mesmo canal (CSMA-CA do inglês *Carrier Sense Multiple Access with Collision Avoidance*);
- Medidor da potência que está sendo recebida no canal, independente se está ou não recebendo mensagens, permitindo detectar se o canal está sendo ocupado por outra transmissão (utilizado pelo algoritmo CSMA-CA) e determinar o RSSI de uma mensagem recebida;
- Tempo garantido para determinadas transmissões (GTSs do inglês *Guaranteed Time Slots*);
- Baixo consumo de energia, permitindo que todos os nodos entrem em modo de economia de energia (*sleep*) ao mesmo tempo;
- Endereçamento com 16 bits ou 64 bits.

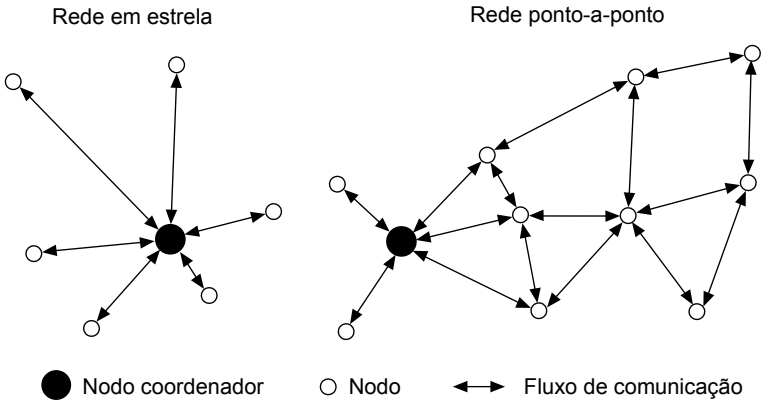


Figura 2.12: Redes em estrela e ponto-a-ponto (adaptado de (CALLAWAY et al., 2002)).

Apesar de ter sido desenvolvido um padrão para o MAC, ainda é realizada uma série de pesquisas nesta área para otimizar questões como redução do consumo, redução de conflitos e aumento da taxa de transmissão, conforme visto em (BACHIR et al., 2010). Otimizar estas características é uma tarefa árdua pois não se consegue isolá-las, sendo o tráfego irregular de dados e a mobilidade os maiores agravantes.

De um modo geral, o objetivo principal da camada MAC é manter o rádio no modo *sleep* a maior parcela de tempo possível sem comprometer o fluxo de dados, visto que neste modo ele é incapaz de atuar como roteador de mensagens. Isto é essencial para aumentar a vida útil da rede por dois motivos, sendo o primeiro que no modo de espera de mensagens (rádio em *idle*) o consumo de potência é alto se comparado ao *sleep*, conforme visto na Tabela 2.1. Já o segundo é que o rádio consome a maior parte da energia dos nodos, conforme (BACHIR et al., 2010).

Tabela 2.1: Consumo de corrente do módulo de rádio CC2500, conforme (Texas Instruments, 2013).

Modo de operação	Consumo de corrente
Sleep	900 nA
Idle	1,5 mA
Recebendo	14 mA
Transmitindo	22 mA

O padrão IEEE 802.15.4 prevê dois modos de operação denominados de *beacon-enabled* e *nonbeacon-enabled* para WPAN do tipo ponto-a-ponto, sendo que em ambos existe um nodo coordenador, cuja função será descrita no próximo parágrafo. O primeiro método permite funções mais interessantes que o segundo, conforme (KOUBAA; ALVES; TOVAR, 2006) e (RAMACHANDRAN; DAS; ROY, 2007), sendo este o motivo para a sua explicação mais de

detalhada neste trabalho.

No modo *beacon-enabled*, o coordenador envia mensagens periodicamente em modo *broadcast*, ou seja, endereçado a todos os nodos da rede. Estas são denominadas de *beacons* e carregam informações a cerca dos tempos que formarão o *superframe*, cujo período total é o próprio intervalo entre dois *beacons*. O *superframe* é composto pelo período de acesso (CAP do inglês *Contention Access Period*), período livre (CFP do inglês *Contention-Free Period*) e período inativo, conforme visto na Figura 2.13.

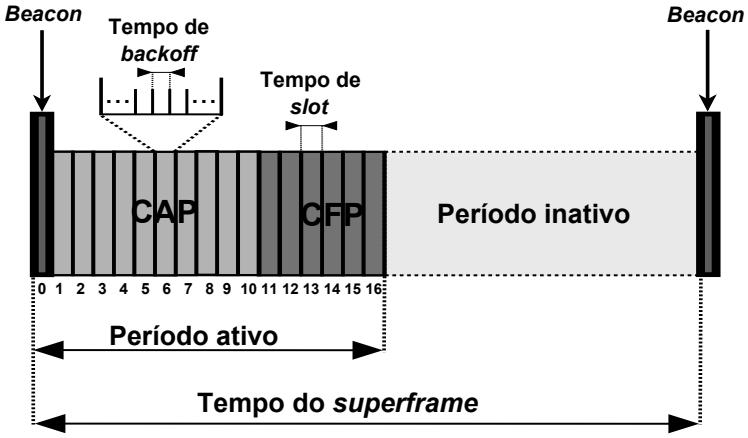


Figura 2.13: Temporização no modo de operação *beacon-enabled* (adaptado de (IEEE, 2003)).

O período ativo é dividido em 16 partes denominadas *slots*, sendo estas divididas entre o CAP e o CFP, caso este último seja utilizado pelo coordenador. O CAP é destinado ao uso normal dos nodos, no qual todos tentarão transmitir suas mensagem, enquanto que o CFP é utilizado para os GTSs (*Guaranteed Time Slots*), os quais são criados pelo coordenador da rede para suprir determinadas aplicações que requerem uma baixa latência ou taxa de transmissão específica. Os períodos ativo e o total do *superframe*, considerando a banda de 2450 MHz, são definidos de maneira semelhante através da equação $15,36ms \times 2^x$, sendo x uma variável inteira que varia de 0 a 14, de modo que o tempo ativo seja menor ou igual ao total. Logo, os períodos mínimo e máximo do *superframe* são $15,36ms$ e $251,7s$, respectivamente.

Para não ocorrerem conflitos entre as transmissões de dois nodos durante o CAP, o padrão IEEE 802.15.4 definiu o método chamado de *slotted* CSMA-CA, o qual opera com períodos de tempo denominados *backoff slots*. Estes são sincronizados com o início do *superframe* e possuem duração de $320us$ para a banda de $2450MHz$, e equivale à transmissão de 80 bits.

Neste método, para poder transmitir, um nodo precisa esperar pela próxima borda de *backoff slot* e então checar a potência do canal, denominada *Clear Channel Assessment* (CCA),

permitindo saber se outro nodo não o está ocupando. Caso o canal esteja livre, deve-se esperar um número aleatório de *backoff slots* e então realizar uma segunda checagem, sendo que se este estiver livre novamente, a transmissão pode iniciar no próximo *backoff slot*. Toda vez que o canal for encontrado ocupado, o nodo espera um número aleatório de *backoff slots* e reinicia o processo, tendo sempre que encontrar o canal livre duas vezes consecutivas para poder iniciar sua transmissão.

Já no modo de operação *nonbeacon-enabled* não existe a estrutura do *superframe*, sendo que para evitar conflitos nas transmissões é utilizado o método *unslotted* CSMA-CA. A principal diferença deste para o *slotted* CSMA-CA é que não é necessário esperar pelas bordas de *backoff slots* para realizar a CCA ou iniciar a transmissão, visto que os nodos não recebem os *beacons* para sincronizarem suas bordas entre si.

Conforme (BACHIR et al., 2010), os diferentes protocolos para a camada MAC encontrados na literatura podem ser divididos nos baseados em contenção, como é o caso do CSMA-CA, e em reserva, o qual não possui definição pelo padrão IEEE 802.15.4. Ainda segundo este autor, o principal exemplo deste segundo caso é o TDMA (do inglês *Time Division Multiple Access*), no qual é atribuído um período par cada nodo transmitir, evitando conflitos, aumentando o fluxo de dados e reduzindo o consumo de energia e latência da rede. Porém, para isto são necessários o sincronismo e conhecimento desta, o que é relativamente complexo e custoso, ainda mais para redes móveis. Por fim, está fora do escopo deste trabalho estudar mais a fundo as características de diferentes protocolos MAC, bem como algoritmos de roteamento, sincronismo e descoberta de rede, visto que cada um destes representa uma área de pesquisa vasta.

2.4.2 Modelo de propagação log-normal

Conforme (STEHLIK, 2011), os três modelos de propagação mais utilizados nos simuladores para RSSF são o de espaço livre, reflexão no solo (dois raios) e log-normal com sombreamento, sendo todos detalhados em (RAPPAPORT, 2001). O primeiro e mais simples, consiste em calcular o decaimento da potência do sinal RF de acordo com o aumento da distância em relação ao transmissor, conforme

$$P_r(d) = P_t \left(\frac{\lambda}{4\pi d}\right)^2 \frac{G_t G_r}{L}, \tag{2.13}$$

sendo $P_r(d)$ a potência recebida a uma distância d, P_t a potência transmitida, λ o comprimento de onda da portadora, $L(L \geq 1)$ o fator de perdas do sistema e G_t e G_r os ganhos das antenas transmissora e receptora, respectivamente.

Um aprimoramento deste modelo de propagação é o de reflexão do solo, pois considera a chegada no sinal refletido pelo solo no receptor. Este modelo é mais preciso que o de espaço livre para distância maiores, e é representado por

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{(d)^4 L}, \tag{2.14}$$

sendo h_t^2 e h_r^2 as alturas das antenas transmissora e receptora.

Porém, percebe-se que ao utilizar estes dois modelos os raios de alcance dos nodos são circunferências perfeitas, as quais determinam se uma transmissão é bem sucedida ou não. Como já destacado pelo experimento realizado por (KOTZ et al., 2004), o melhor modelo para representar o comportamento da propagação do sinal RF em redes sem fio é o decaimento logarítmico somado a uma variável aleatória com distribuição normal para representar efeitos como sombreamento, reflexão e interferência. Além disso, este modelo, definido por

$$P_r(d) = P_r(d_0) + 10\eta \log\left(\frac{d}{d_0}\right) + X_\sigma, \tag{2.15}$$

visa adaptar os fatores de larga escala (η) com os de pequena escala ($X_\sigma(dBm)$) ao ambiente em questão, sendo os utilizados pelo simulador NS-2 apresentado nas Tabelas 2.2 e 2.3. Ainda para (2.15), d_0 representa a distância mínima a partir da qual a equação pode ser utilizada e $P_r(d_0)$ geralmente obtida a partir de medições.

Tabela 2.2: Valores típicos para η conforme (Documentação NS-2, 2012).

Ambiente		η
Externo	Espaço livre	2
	Área urbana encoberta	2,7 até 5
Interno	Linha de visada direta	1,6 até 1,8
	Sem linha de visada direta	4 até 6

Desta forma, uma maneira para determinar se uma mensagem será ou não recebida é definir um patamar de ruído do sistema (do inglês *noise floor*), calcular o SNR do sinal recebido e aplicar sobre este uma função probabilística que determina a taxa de recepção de mensagens (PPR do inglês *packet reception rate*), conforme visto na Figura 2.14. Conforme (ZUNIGA; KRISHNAMACHARI, 2004), esta função é dependente da modulação, da taxa de transmissão

Tabela 2.3: Valores típicos para $X_\sigma(dB)$ conforme (Documentação NS-2, 2012).

Ambiente	$X_\sigma(dB)$
Externo	4 até 12
Escritório, repartição rígida	7
Escritório, repartição macia	9,6
Fábrica, linha de visada	3 até 6
Fábrica, obstruído	6,8

e do número de bits transmitido em cada pacote. Porém, neste trabalho é utilizado apenas o modo com uma potência limite, denominada sensibilidade, a qual definirá se uma mensagem será ou não recebida com sucesso, conforme (KOTZ et al., 2004).

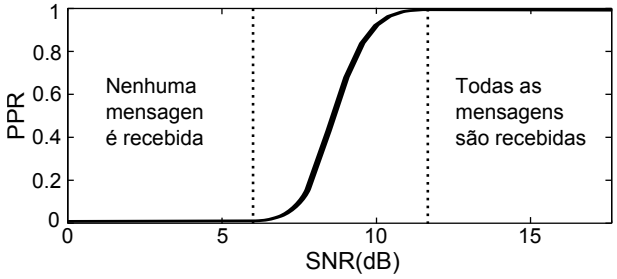


Figura 2.14: Exemplo de função que relaciona a PPR com o SNR (adaptado de (ZUNIGA; KRISHNAMACHARI, 2004)).

Conforme visto na Seção 2.2, existem algoritmos de localização para RSSF que utilizam a medida de potência do sinal recebido, nestre trabalho chamada de RSSI, para estimar distâncias. No caso do modelo de propagação log-normal, a equação para transformar uma medida de RSSI numa distância é obtida isolando-se d de (2.15), resultado em

$$d(P_r) = 10^{\frac{P_t - Pr(d_0) - P_r}{10\eta}}. \tag{2.16}$$

Umas das maiores dificuldades na utilização desta equação é a necessidade do conhecimento de η, pois este é dependente do ambiente e, geralmente, obtido a partir de experimentos, conforme realizado em (YEDAVALLI; KRISHNAMACHARI, 2009).

Para compreender melhor o comportamento do modelo de propagação log-normal, realizaram-se algumas simulações visando avaliar a potência do sinal recebido, a PRR e a estimação de distância. Considerando dados característicos do nodo MICAz dotado do chip de rádio CC2420, conforme (STEHLIK, 2011), gerou-se o gráfico na Figura 2.15 para um sinal com $P_t = 0dBm$,

$d_0 = 1m$, $P_r(d_0) = -55dBm$, $\eta = 3$, $X_\sigma(dB) = 4dBm$. Nesta figura realizou-se uma varredura de 1 a 50 metros com a geração de 50 pontos a cada intervalo de $1m$.

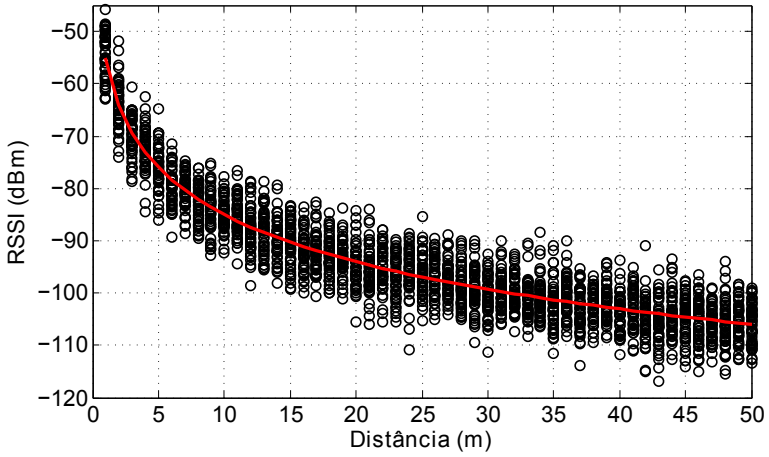


Figura 2.15: Modelo de propagação log-normal com sombreamento.

Conforme (ZUNIGA; KRISHNAMACHARI, 2004), um dos principais efeitos deste modelo de propagação sobre os resultados de simulações em RSSF, comparado com os modelos de espaço livre e reflexão do solo, é a criação de uma região de transição na qual as mensagens podem ou não ser entregues. Para compreender esta região criou-se o gráfico visto na Figura 2.16, com a mesma configuração do visto na Figura 2.15, para mostrar o comportamento da PRR com a variação de $X_\sigma(dBm)$. Destaca-se que cada ponto das curvas desta figura representa a média de 10000 pontos gerados e para uma sensibilidade de $-95dBm$, conforme (Texas Instruments, 2012) e (STEHLIK, 2011).

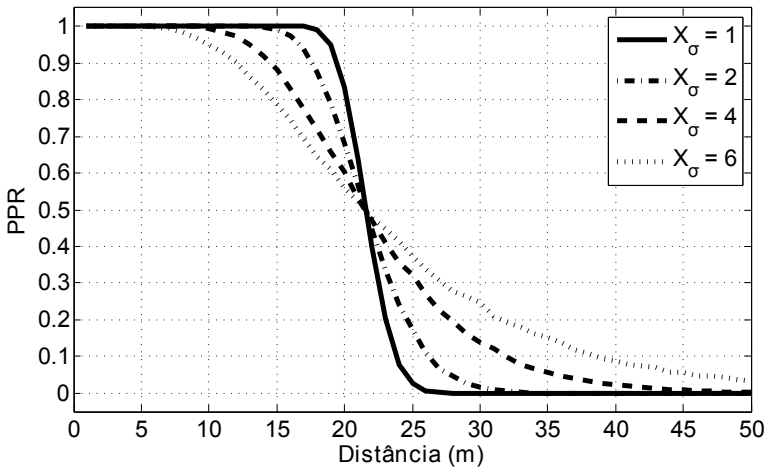


Figura 2.16: Diferentes regiões de transição de acordo com $X_\sigma(dBm)$.

Conforme (YEDAVALLI; KRISHNAMACHARI, 2009), para se minimizar o efeito do

sombreamento ($X_\sigma(dBm)$) sobre as estimativas de distância geradas pelo RSSI, realizam-se um determinado número de transmissões em sequência para então utilizar a média das potências recebidas por estas ($\overline{P_r}$) em (2.16). Porém, à medida que a distância aumenta e inicia-se a região de transição, as mensagens com potências menores que a sensibilidade do receptor são ignoradas, reduzindo o número de medidas para calcular a média, conforme a Figura 2.17. Esta figura diferencia-se da 2.15 apenas pela eliminação dos pontos abaixo da linha de sensibilidade de $-95dBm$.

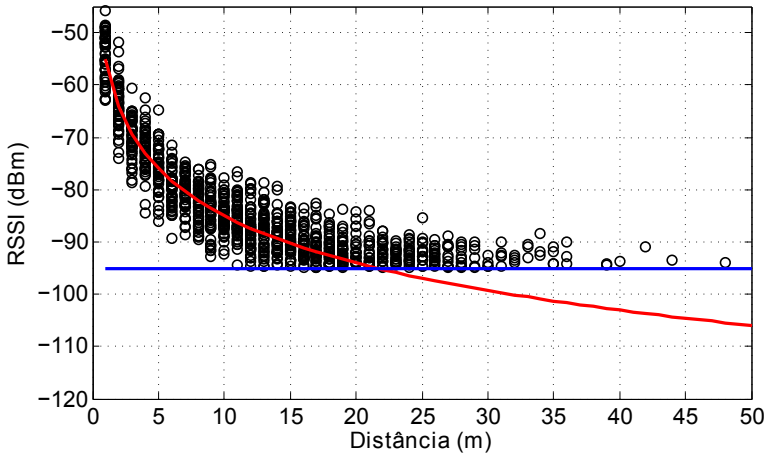
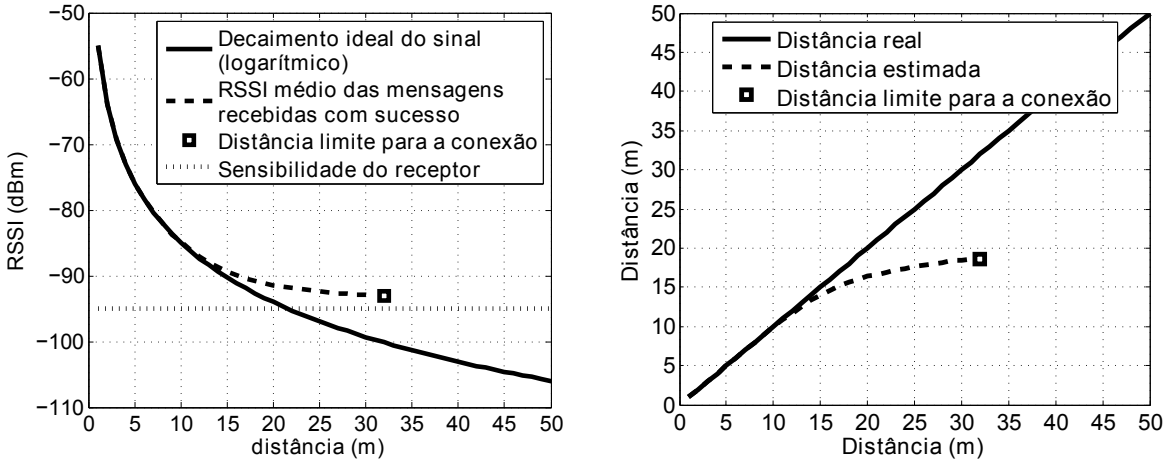


Figura 2.17: Recepção com sensibilidade de $-95dBm$ do sinal visto na Figura 2.15.

Além disso, percebe-se através da Figura 2.18(a) que a média das potências recebidas sempre será maior que a sensibilidade, fazendo com que as distâncias calculadas por (2.16) sejam cada vez mais distantes da real, conforme visto na Figura 2.18(b). Este comportamento termina quando a distância é grande o suficiente para que a PRR seja menor que $0,1$, neste caso a partir de $31,9m$, permitindo considerar-se que não há mais conectividade entre transmissor e receptor, conforme (SRINIVASAN et al., 2010).

Algumas maneiras para se minimizarem estes problemas podem ser, primeiramente, definir uma PRR mínima, resultante de uma sequência de transmissões, para se gerar as estimativas de distância, por exemplo $0,8$. Outro modo seria compensar a diferença entre a distância real e a estimada por (2.16), na região de transição, através de alguma função, a qual teria como variável de entrada a própria PRR. Por fim, conforme (ZUNIGA; KRISHNAMACHARI, 2004)), deve-se ter cuidado na determinação do η através da coleta dados experimentais, pois devem ser utilizadas para tal apenas as medições que não foram afetadas consideravelmente pela região de transição.



(a) Média das potências recebidas acima da sensibilidade de $-95dBm$ do receptor.

(b) Distâncias estimadas a partir da média das potências recebidas.

Figura 2.18: Resolução de (2.16) para a geração das estimativas de distância entre transmissor e receptor a partir de um total 10000 transmissões para cada passo de $1m$ na distância.

2.4.3 Colisões e falhas de transmissão

Nesta última parte desta seção deseja-se fazer algumas considerações acerca de colisões e falhas de transmissão, mostrando que estas tarefas são complexas de serem modeladas e fortemente ligadas ao ambiente em questão, sendo estes alguns dos principais motivos para não se considerarem tais efeitos no desenvolvimento do simulador apresentado no Capítulo 4. Conforme (BACHIR et al., 2010), uma colisão ocorre quando as camadas MAC de mais de um nodo não conseguem evitar que estes transmitam no mesmo canal, durante um mesmo intervalo de tempo e conectado ao mesmo receptor, causando a perda de mensagens. Já uma falha na transmissão ocorre quando se descarta uma mensagem por causa da interferência com sinais provenientes de outras fontes. Destaca-se que problemas nas transmissões possuem um maior impacto sobre os algoritmos de roteamento, visto que é função destes garantir a entrega dos dados de maneira segura e com eficiência energética, ou seja, passando pelo menor número de nodos possível.

Mesmo utilizando-se o modelo de propagação log-normal com sombreamento e uma camada MAC padronizada, para se obter um modelo realístico de colisão é necessário considerar uma série de outros fatores. O primeiro deles baseia-se no fato de que as antenas não são perfeitamente omnidirecionais, causando variações nos ganhos de transmissão e recepção em diferentes orientações, conforme (ZHOU et al., 2006). Ainda segundo este autor, outros aspec-

tos importantes são que os raios de interferência dos nodos são maiores que os de conexão, e que diferentes tensões das baterias podem causar variações nas potências de transmissão e recepção. Como um dos principais efeitos destas características realistas, destaca-se a assimetria durante a troca de mensagens entre dois nodos, fazendo com que as PRRs sejam diferentes em ambas as direções, sendo isto comprovado por experimentos em (SRINIVASAN et al., 2010).

(ZHAO et al., 2013), por sua vez, ressaltou que ocorrem variações nas medidas do RSSI com o passar do tempo e com mudanças de temperatura e umidade, o que prejudica a determinação do fator de decaimento de determinado ambiente ao considerar-se o modelo de propagação log-normal. Já (KOHTAMAKI et al., 2009) considerou a atenuação das paredes no decaimento do sinal em ambientes internos, o que pode ser expandido para o modelamento de obstáculos num ambiente externo. Por fim, no quesito colisões entre transmissões, (ZHANG et al., 2010) utilizou um modelo que considera a propagação do sinal em 3 dimensões para ambientes internos, visto que o raio de alcance geralmente é menor nestes casos.

Quanto a falhas na transmissão, conforme (SRINIVASAN et al., 2010) destaca-se como um dos principais problemas o compartilhamento da banda de $2450MHz$ pelos padrões IEEE 802.15.4, 802.11b (*Wi-Fi*) e 802.15.1 (*BlueTooth*), cujas alocações de banda são vistas na Figura 2.19. Ainda conforme este autor, são dois os efeitos deste compartilhamento que prejudicam as RSSF, sendo o primeiro as falhas devido a um nodo estar transmitindo e perder a conexão devido ao aumento do nível de interferência do canal proveniente da utilização das mesmas frequências em algum dos outros padrões. O segundo é o aumento da latência, pois quando um nodo tenta transmitir utilizando o método de acesso ao canal CSMA-CA, conforme visto na Seção 2.4.1, ele acaba avaliando o canal como ocupado devido à presença de potência, a qual é proveniente de transmissões nos outros padrões.

Percebe-se que muitas das características que tornam as colisões e falhas de transmissão mais reais também seriam importantes ao se considerar a estimação das distâncias através da potência do sinal recebido. Porém, como deseja-se avaliar um método de multilateração que pode ser utilizado em algoritmos de localização com diversos métodos para a estimação das distâncias entre nodos, conforme visto na Seção 2.2, considera-se a utilização do modelo de propagação log-normal com sombreamento suficiente.

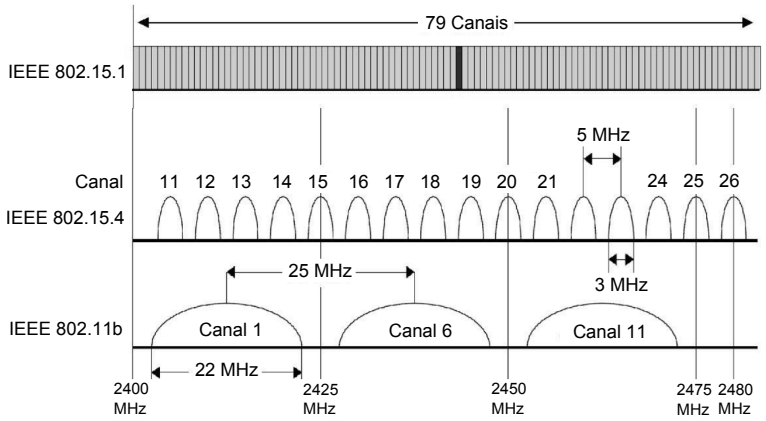


Figura 2.19: Compartilhamento de banda pelos padrões IEEE 802.15.4, 802.11b e 802.15.1 (adaptado de (SRINIVASAN et al., 2010)).

3 ALGORITMO DE MULTILATERAÇÃO PARA REDES DE SENSORES SEM FIO BASEADO NO MÉTODO DE MÁXIMA VEROSSIMILHANÇA

3.1 Método de multilateração iterativo

Os métodos de multilateração por MQL e MV, apresentados na Seção 2.3, possuem uma carga computacional relativamente elevada para nodos de uma RSSF. No primeiro, é necessário o trabalho com matrizes que aumentam de maneira quadrática em relação ao número de nodos âncora considerados, o que, além do alto consumo de potência, exige grande quantidade de memória durante sua execução. Já o principal problema do segundo é o elevado número operações aritméticas, dentre elas a divisão e a raiz quadrada, impactando também num elevado consumo de potência. Além disso, o método MV baseado no gradiente é fortemente dependente de um ponto inicial, podendo convergir para mínimos locais e não globais.

Uma solução para minimizar o custo computacional do método MQL é apresentado em (BORN; BILL, 2010), onde os cálculos são divididos em uma primeira etapa complexa e outra simples. A inicial é realizada em nodos com maior capacidade de processamento e de energia, sendo executada apenas uma vez para redes fixas, tendo como resultado uma matriz que é enviada para os nodos desconhecidos finalizarem os cálculos de suas localizações. Uma das desvantagens desta técnica é o aumento da transmissão de pacotes, pois, conforme (BACHIR et al., 2010), a maior parcela do consumo de potência dos nodos está relacionado ao módulo de comunicação sem fio, resultando num aumento do consumo de energia considerável por parte dos nodos. Outra desvantagem é o aumento da complexidade do método, devido a sua execução de maneira distribuída.

Dessa maneira, decidiu-se explorar neste trabalho métodos simplificados de multilateração baseados na MV com o intuito de reduzir sua complexidade computacional ao custo de um pequeno aumento no erro de localização. Porém, mesmo com este aumento espera-se obter um EL menor que o do método MQL considerando ambientes com medidas inexatas. Para explicar o funcionamento do algoritmo de multilateração desenvolvido neste trabalho, utilizou-se um cenário com a mesma disposição dos nodos âncora vista na Figura 2.9, com um nodo desconhecido posicionado nas coordenadas $(60, 90)$ e com medidas de distâncias entre nodos obtidas com exatidão. Ressalta-se que a execução dos cálculos ocorre apenas nestes nós desconhecidos que desejam se localizar.

Este algoritmo de multilateração possui duas etapas, tendo como informações iniciais as posições dos n nodos âncora (x_i, y_i) e medidas de distâncias (r_i) entre estes e o nodo desconhecido, o qual esta executando a multilateração, geradas a partir de algum dos métodos de estimação de distância apresentados na Seção 2.2. Na primeira etapa define-se as coordenadas do ponto de partida (x_t, y_t), neste caso utilizou-se o centro centro geométrico dos nodos âncora, e traçam-se retas (marcadas com tracejado nas Figuras 3.1 e 3.2) em direção a este ponto inicial partindo dos nodos âncora, sendo estas de comprimento r_i. A Figura 3.1 exemplifica esta primeira etapa considerando apenas o primeiro âncora, tendo como resultado a geração do ponto marcado com um triângulo de mesma numeração do respectivo nodo de referência.

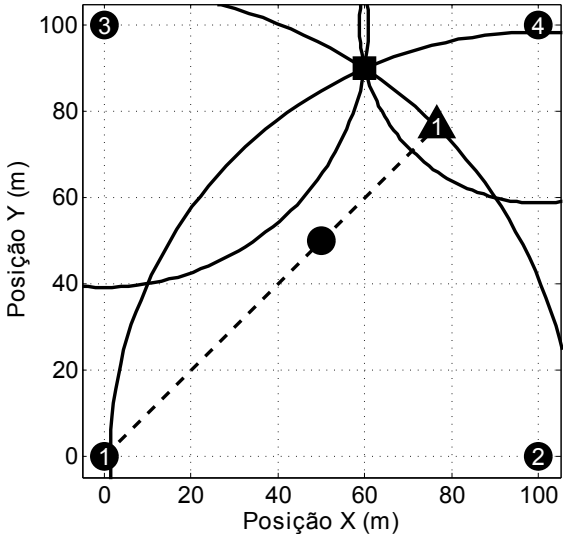


Figura 3.1: Execução de parte de uma iteração.

A segunda etapa consiste apenas em calcular o centro geométrico dos n pontos marcados com triângulos apresentados na Figura 3.2, resultando no ponto (x_{t+1}, y_{t+1}) marcado com um quadrado, o qual é o resultado de uma iteração. Analisando a Figura 3.2 percebe-se que este último ponto está mais próximo do MRQ do cenário se comparado ao ponto inicial. A execução deste algoritmo pode ser calculado por

$$x_{t+1} = x_t + \frac{w}{n}\sum_{i=1}^{n}(x_t - x_i)(r_i/d_i - 1) \tag{3.1}$$

$$y_{t+1} = y_t + \frac{w}{n}\sum_{i=1}^{n}(y_t - y_i)(r_i/d_i - 1), \tag{3.2}$$

sendo w um ganho arbitrário definido inicialmente como sendo 1, cuja utilização é explicada na Seção 3.2, e d_i dada por

$$d_i = \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2}. \tag{3.3}$$

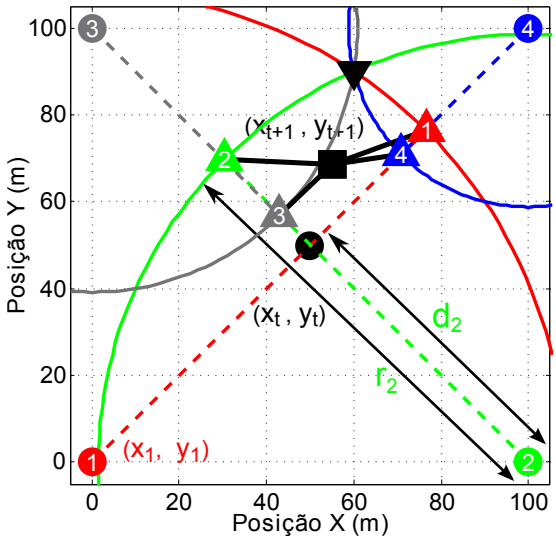


Figura 3.2: Execução de uma iteração do método de multilateração iterativo desenvolvido, tendo como resultado o ponto marcado com um quadrado.

Executando-se este processo repetitivamente para o mesmo cenário da Figura 3.2, utilizando como ponto inicial (x_t, y_t) de cada iteração o resultado da anterior (x_{t+1}, y_{t+1}), percebe-se que converge para o ponto de MRQ do cenário, conforme a Figura 3.3. Nesta figura, as curvas de nível representam regiões de mesmo residual quadrático e os pontos marcados com "x"os resultados de cada iteração. Assim, denominou-se o algoritmo que implementa (3.1) e (3.2) de máxima verossimilhança iterativa (MVI) para fins de futuras comparações.

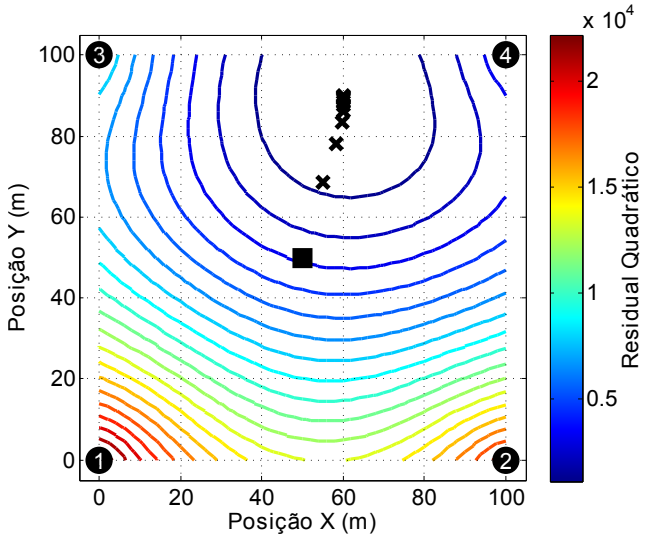


Figura 3.3: Sequência de iterações para o método MVI convergir para o ponto de MRQ do cenário.

Definiram-se duas variáveis de controle para interromper o processo iterativo deste al-

goritmo de multilateração. A primeira consiste em uma distância mínima entre dois pontos gerados consecutivamente, enquanto a segunda limita o processo a um número máximo de iterações. A simulação vista na Figura 3.3 foi interrompida pela primeira variável, definida como $0,1m$, na décima primeira iteração.

Analisando a sequência de iterações da Figura 3.3 e de outras simulações, percebe-se que os pontos gerados são uma série de vetores que apontam para locais de maior redução do residual quadrático. Conforme (THOMAS, 2002), este comportamento é característico do vetor gradiente, consistindo no cálculo de derivadas direcionais que apontam para maior variação de determinada função, neste caso a maior redução do residual quadrático, auxiliando na minimização ou maximização da mesma. (YEDAVALLI, 2002) utilizou deste princípio para a realização de multilaterações com medidas de distâncias baseadas na potência do sinal recebido, percebendo-se certa semelhança com as equações utilizadas neste trabalho.

3.2 Algoritmo de multilateração por aproximação polinomial

Analisando a Figura 3.2, percebe-se que a orientação de uma reta traçada a partir do ponto central até o resultado da primeira iteração é muito próxima da direção do MRQ. Logo, decidiu-se avaliar este comportamento em todo o cenário visto na Figura 2.9, executando uma iteração para cada nodo desconhecido utilizando o ponto central como inicial, sendo o resultado visto na Figura 3.4.

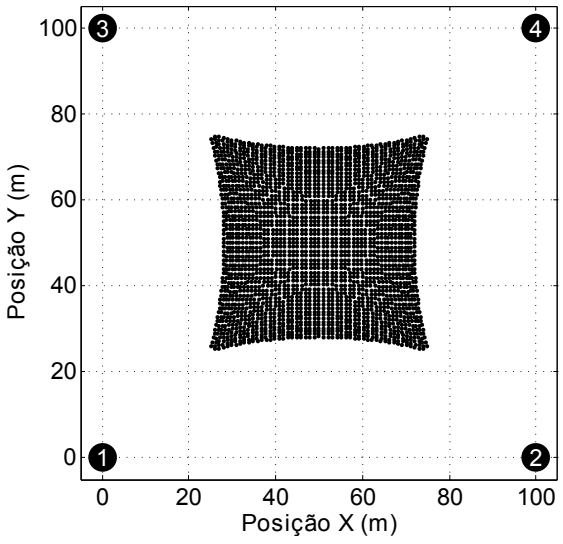
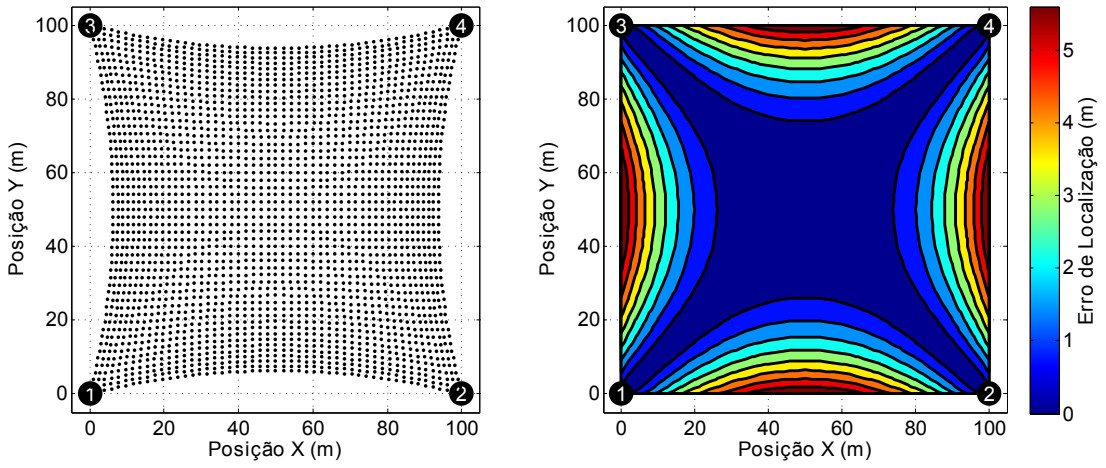


Figura 3.4: Posições calculadas pela execução de uma iteração para cada nodo desconhecido pelo método MVI, considerando o ponto central como inicial.

Considerando as posições calculadas vistas na Figura 3.4 como as estimadas para a localização dos nodos desconhecidos, decidiu-se aumentar o ganho w das Equações (3.1) e (3.2) para 2 com o intuito de aproximar estas das reais. O resultado deste teste é apresentado pelas Figuras 3.5(a) e 3.5(b), sendo que a primeira apresenta as posições estimadas e a segunda o erro de localização dos nodos desconhecidos no formado de curvas de nível. Como principais resultados desta simulação destacam-se o erro de localização médio (ELM) do cenário igual a $1,78m$, e que o este erro aumenta de maneira acentuada conforme se afasta horizontalmente ou verticalmente do centro do cenário. Através de várias simulações variando o w com passos de $0,1m$, encontrou-se o menor ELM, igual a $1,55m$, para $w = 2,04$.

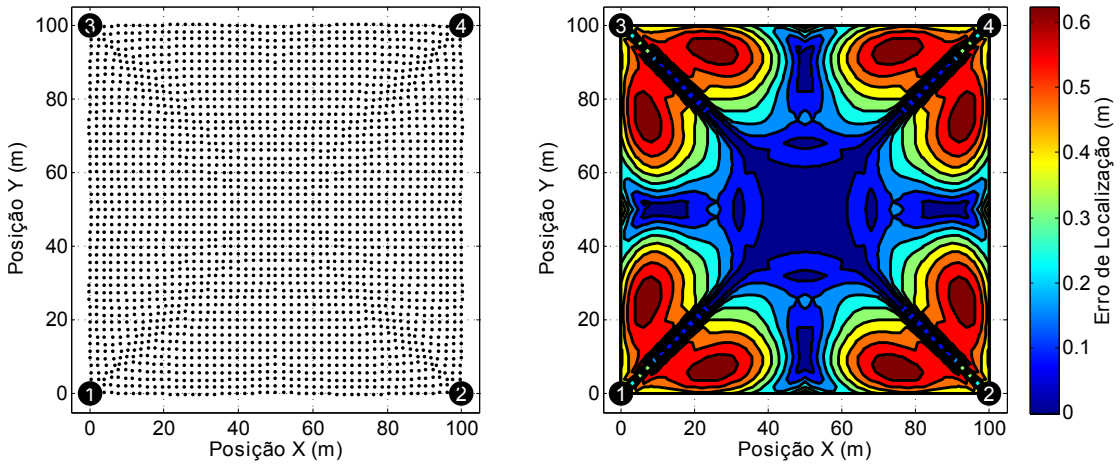


(a) Posições calculadas. (b) Curvas de nível do erro de localização.

Figura 3.5: Execução de uma iteração para cada nodo desconhecido pelo método MVI, considerando o ponto central com inicial e $w = 2$.

Visando reduzir o EL das posições estimadas geradas com $w = 2$, vistas na Figura 3.5(a), aplicaram-se duas correções por aproximações polinomiais sobre estes resultados através da função *polyfit* do Matlab. Ambas utilizam polinômios de segunda ordem, sendo este método de multilateração, denominado de aproximação polinomial (AP), apresentado na forma de *script* na linguagem Matlab no apêndice B. Estas correções utilizam como ponto de partida para os cálculos as distâncias e angulações destas primeiras estimativas de localização em relação ao ponto central do cenário. O resultado deste método é apresentado nas Figuras 3.6(a) e 3.6(b), destacando-se o ELM igual a $0,33m$ e que o EL máximo foi reduzido em quase dez vezes.

Na Figura 2.11 é comparado o resultado do método de multilateração AP com os MQL e MV na presença de medidas inexatas entre os nodos desconhecidos e os âncoras, de maneira idêntica a vista na Figura 2.10. Como principal resultado, destaca-se que o ELM do método AP



(a) Posições calculadas. (b) Erro de localização.

Figura 3.6: Avaliação do erro de localização para o método de multilateração AP.

é muito próximo do MV, considerado neste trabalho como o melhor resultado possível. Existem trabalhos, como (KUNG et al., 2009) e (YEDAVALLI, 2002), que visam obter resultados melhores que a MV baseados na compreensão do comportamento das imprecisões nas medidas. Porém, estes benefícios aumentam a complexidade do algoritmo e podem depender de questões físicas de implementação do sistema, o que está fora do escopo deste trabalho.

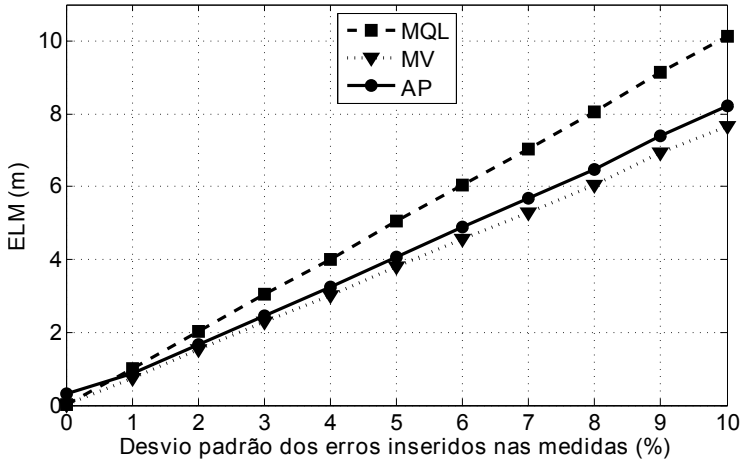


Figura 3.7: Comparação do erro de localização médio entre os métodos MQL e MV na presença de medidas imprecisas.

Como principais vantagens do método de multilateração AP, podem-se destacar a baixa complexidade computacional e o ELM muito próximo do MV. Pelo fato de os nodos âncora serem equidistantes do ponto central, não é necessária a execução da raiz quadrada, o que reduz ainda mais a complexidade deste método. Já como principal desvantagem está a dependência do

cenário, o que inviabiliza a sua utilização na maioria dos cenários reais. Porém, o que se deseja mostrar com o método proposto (MVI) é que em cenários mais genéricos, não necessariamente com disposição quadrada dos nodos âncora, é possível obter-se uma redução expressiva da complexidade mantendo a precisão através de aproximações polinomiais.

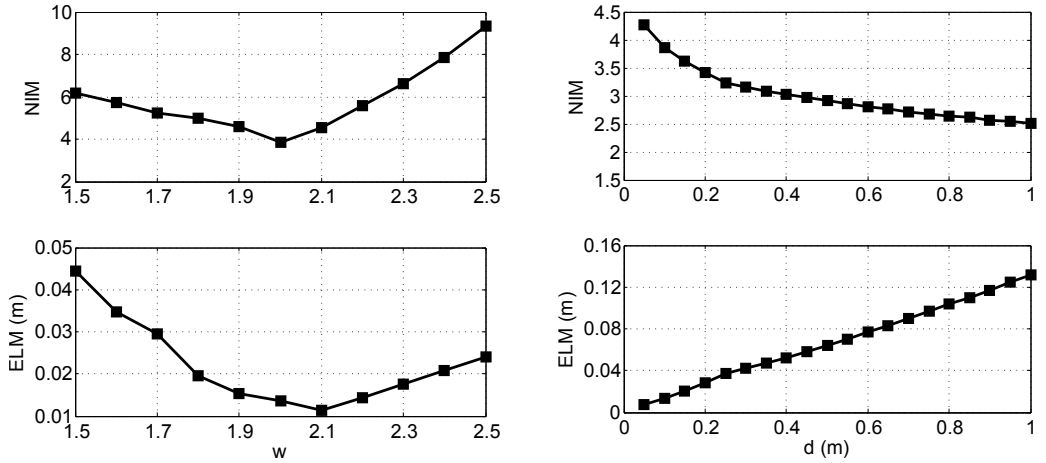
3.3 Caracterização do algoritmo MVI

Como já mencionado na Seção 3.1, o método de multilateração MV possui como principal desvantagem o alto consumo de potência devido a elevada quantidade de operações aritméticas. Deste modo, decidiu-se analisar os aspectos de ELM e número de iterações médio (NIM) do algoritmo MVI alterando-se a disposição e o número de nodos âncora, o ganho w das Equações (3.1) e (3.2), e as variáveis de controle. Estas últimas são a distância mínima entre dois pontos gerados consecutivamente e número máximo de iterações por nodo desconhecido, representadas pelas letras d e m, respectivamente. Como o número de possíveis combinações para serem analisadas é grande, decidiu-se apresentar os resultados mais importantes e esclarecedores.

O primeiro teste realizado foi para o cenário visto na Figura 2.9, utilizado na Seção 3.2, variando-se w e d com o intuito de analisar o comportamento do NIM e ELM. Inicialmente, definiram-se como valores padrão $w=2$, $d=0,1m$ e $m=20$, e utilizaram-se distâncias exatas medidas entre nodos. Como resultados da variação do ganho, conforme a Figura 3.8(a), podem ser destacados que com $w=2$ minimiza-se NIM para $3,85$, e com $w=2,1$ minimiza-se o ELM para $0,012m$. Já para a variação da distância mínima entre dois pontos, conforme a Figura 3.8(b), pode-se destacar que o NIM e o ELM são inversamente e diretamente proporcionais a d, respectivamente, de modo que o primeiro passa de um comportamento exponencial para linear a partir de $d=0,4m$, enquanto que o segundo se comporta apenas linearmente.

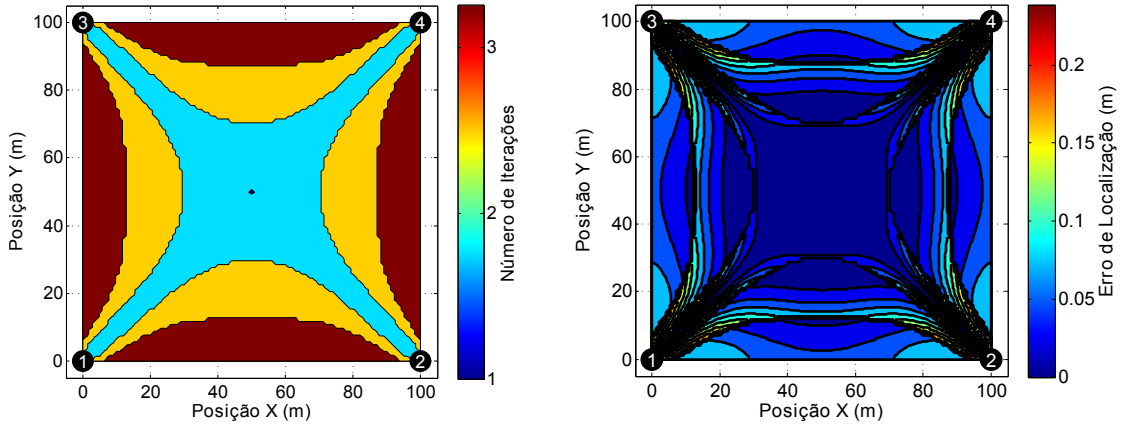
Então, com $d=0,4m$ criaram-se as Figuras 3.9(a) e 3.9(b) para visualizar a distribuição do número de iterações e do erro de localização, respectivamente, em todo o cenário. O objetivo destas simulações é procurar regiões com comportamentos exagerados, sendo que estas não foram encontradas, tendo-se como resultados $NIM=3,03$ e $ELM=0,05m$. Apenas para deixar claro, o ponto inicial não é considerado como iteração para o cálculo do NIM.

A última simulação do cenário visto na Figura 2.9 para o algoritmo MVI está relacionado à adição de imprecisão nas medidas entre os nodos, com as mesmas variáveis de controle utilizadas para gerar as Figuras 3.9(a) e 3.9(b). Como os resultados dos métodos MVI e MV são



(a) Variação de w para d=0,1m. (b) Variação de d para w=2.

Figura 3.8: Simulações do cenário da Figura 2.9 com m=20.



(a) Número de iterações. (b) Erro de localização.

Figura 3.9: Distribuição do número de iterações e do erro de localização para o cenário da Figura 2.9 com w=2, d=0,4m e m=20.

próximos, decidiu-se apresentar um gráfico subtraindo-se o primeiro do segundo, conforme a Figura 3.10, percebendo-se que a diferença entre eles pode ser desconsiderada. Já através da Figura 3.11, percebe-se que o NIM é diretamente proporcional ao desvio padrão das imprecisões percentuais adicionadas às medidas entre os nodos desconhecidos e os âncoras.

Idealmente, para se definirem os valores padrão das variáveis de controle do algoritmo MVI, seria necessário criar uma função custo com as variáveis de desempenho ELM e NIM de modo que esta fosse minimizada. Ainda poderiam ser utilizadas outras características de desempenho deste algoritmo nesta função, no entanto consideram-se que estas duas sejam as principais, pois referem-se à precisão e ao custo computacional ou energético. Porém, para

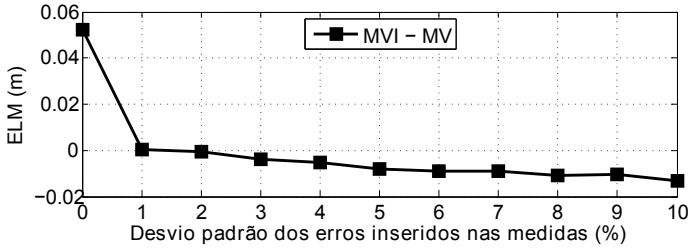


Figura 3.10: Comparação do ELM entre os métodos MVI e MV na presença de imprecisões nas medidas de distâncias entre nodos com *w=2*, *d=0,4m* e *m=20*.

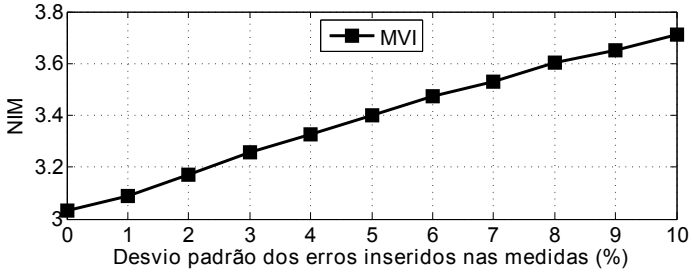


Figura 3.11: Comportamento do NIM na presença de imprecisões nas medidas de distâncias entre nodos com *w=2*, *d=0,4m* e *m=20*.

isto seria necessário um profundo conhecimento da aplicação em questão, pois sabe-se que na área de RSSF a maior parte dos requisitos de projeto dependem desta. Como este trabalho não é direcionado a uma aplicação, na qual se consigam especificações detalhadas, procura-se apenas mostrar o comportamento das variáveis de desempenho em determinados cenários e configurações.

O segundo cenário simulado, visto na Figura 3.12, tem por objetivo avaliar o comportamento do método de multilateração MVI na localização de nodos desconhecidos que estão longe de todos os âncoras. Manteve-se a disposição dos nodos âncora em forma de um quadrado, pois facilitam a identificação e explicação de determinados comportamentos deste algoritmo de multilateração.

Então, para esta nova simulação definiram-se como valores padrão *w=1*, *d=0,1m* e *m=100*, mantendo-se como ponto inicial o centro geométrico dos nodos âncora. Como resultados da variação de *w*, Figura 3.13(a), podem ser destacados que com *w=1,1* minimiza-se NIM para *9,74*, e com *w=2* minimiza-se o ELM para *0,07m*. Também notou-se que na transição de *w=1,9* para *w=2* o NIM aumentou abruptamente, do mesmo modo que a redução do ELM, sendo que para *w>2* os nodos desconhecidos próximos das bordas do cenário pararam de convergir para o ponto de MRQ, gerando erros de localização elevados. Quanto à variação

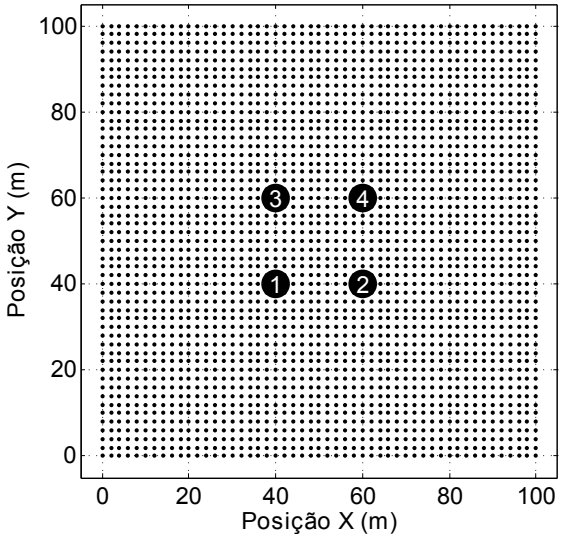
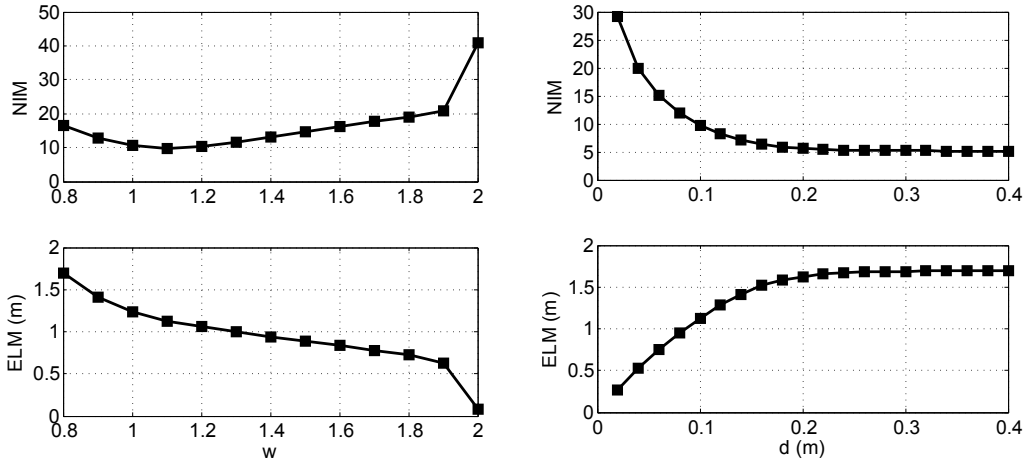


Figura 3.12: Segundo cenário simulado para avaliar o comportamento do método de multilateração MVI.

de *d*, Figura 3.13(b), percebeu-se um comportamento exponencial oposto das duas variáveis de desempenho avaliadas.

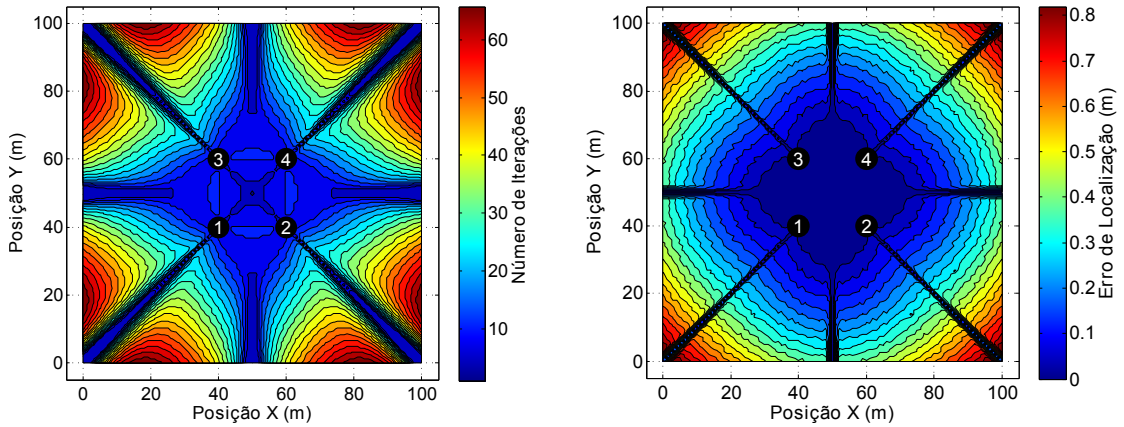


(a) Variação de *w* para *d=0,1m*. (b) Variação de *d* para *w=1,1*.

Figura 3.13: Distribuição do número de iterações e do erro de localização para o cenário da Figura 3.12 com *m=100*.

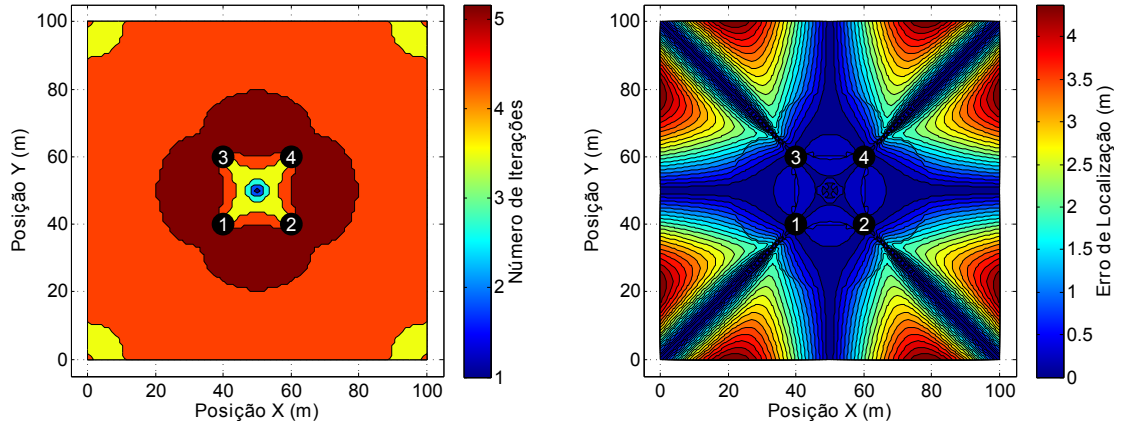
Para compreender melhor o comportamento da simulação variando *d*, analisou-se a distribuição do NIM e do ELM ao longo do cenário em questão para os casos extremos da Figura 3.13(b), ambos com *w=1,1* e mantendo *m=100*. Os resultados para *d=0,02m* são apresentados nas Figuras 3.14(a) e 3.14(b), enquanto que para *d=0,4m* nas Figuras 3.15(a) e 3.15(b).

Analisando estas figuras, percebe-se uma distribuição semelhante das regiões com os



(a) NIM igual a *29,24*. (b) ELM igual a *0,27m*.

Figura 3.14: Distribuição do número de iterações e do erro de localização para o cenário da Figura 3.12 com *w=1,1*, *d=0,02m* e *m=100*.

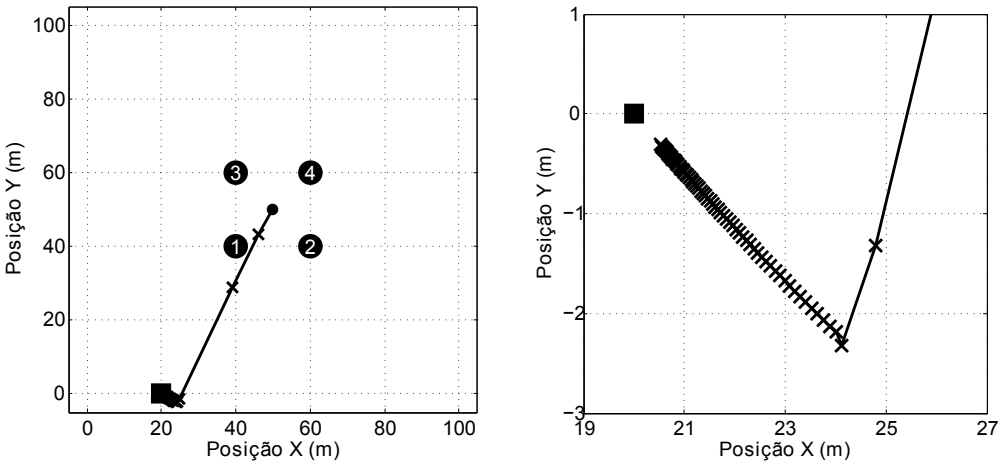


(a) NIM igual a *5,12*. (b) ELM igual a *1,7m*.

Figura 3.15: Distribuição do número de iterações e do erro de localização para o cenário da Figura 3.12 com *w=1,1*, *d=0,4m* e *m=100*.

maiores números de iterações na Figura 3.14(a) e com os maiores erros de localização da Figura 3.15(b). Assim, decidiu-se avaliar a sequência de pontos gerados durante a execução da multilateração de apenas um nodo desconhecido posicionado numa destas regiões, escolhendo-se o nó com as coordenadas *(20,0)*. O resultado desta nova simulação, a qual utilizou as mesmas variáveis de controle do teste visto na Figura 3.14(a), é apresentado nas Figuras 3.16(a) e 3.17(a), sendo esta última apenas uma aproximação do local em que ocorre uma sequência de iterações com pequenas distâncias entre consecutivos pontos.

Definiu-se o comportamento visto na Figura 3.16(b) como problemático, pois fornece uma redução do erro de localização ao custo de um elevadíssimo número e iterações. Assim, um

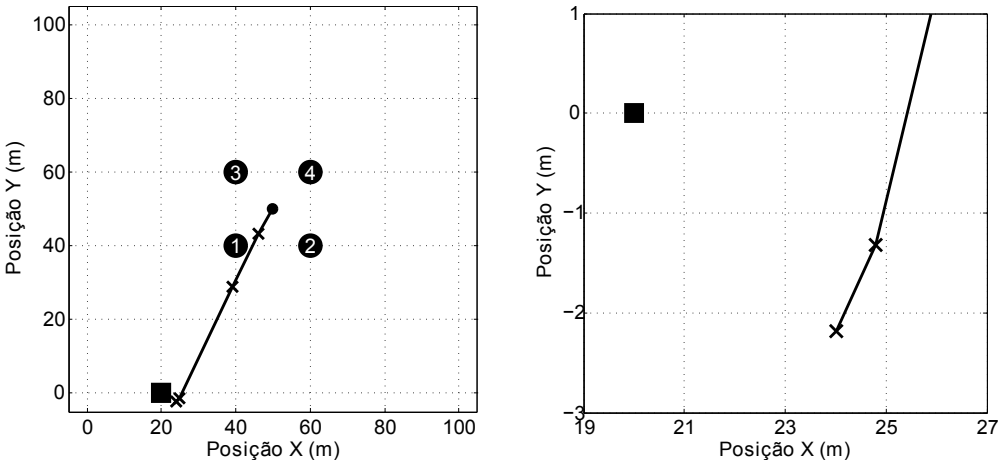


(a) Vista de todo o cenário.

(b) Aproximação da região com pequenas distâncias entre sucessivas iterações.

Figura 3.16: Sequência de iterações (marcadas com "x") do método MVI para um nodo desconhecido (marcado com um quadrado) posicionado nas coordenadas *(20,0)* definindo-se *w=1.1*, *d=0,02m* e *m=100*.

meio para evitar este comportamento é utilizando uma variável *d* grande o suficiente para que o algoritmo seja interrompido antes de iniciar a sequência de iterações com pequenas distâncias entre si, a qual precisa ser maior que *0,2* conforme a Figura 3.13(b). Então, aumentando *d* para *0,4m* reduz-se o total de 68 iterações, vistas nas Figuras 3.16(a) e 3.16(b), para apenas 5, vistas nas Figuras 3.17(a) e 3.17(b), ao custo de elevar o erro de localização de *0,6m* para *4,6m*.

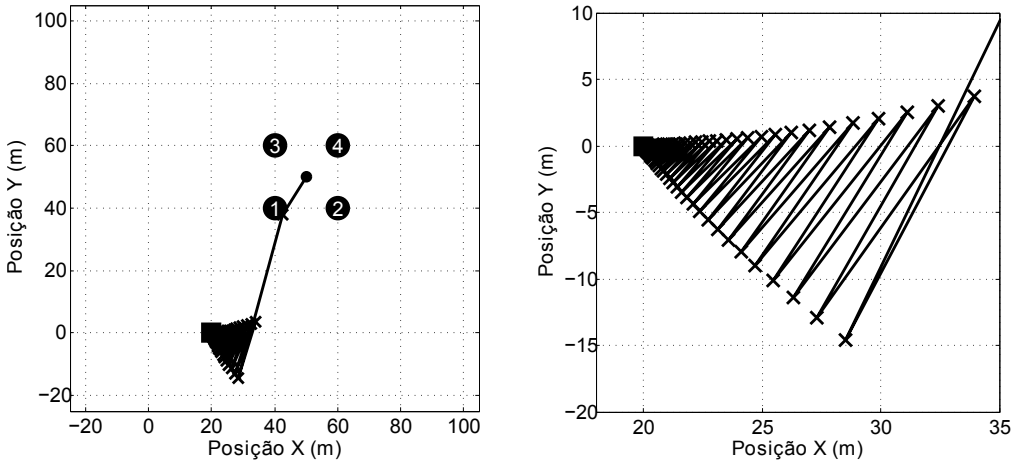


(a) Vista de todo o cenário.

(b) Aproximação da região na qual foi eliminado o elevado número de iterações, visto na Figura 3.17(a), aumentando-se *d* para *0,4m*.

Figura 3.17: Sequência de iterações (marcadas com "x") do método MVI para um nodo desconhecido (marcado com um quadrado) posicionado nas coordenadas *(20,0)* definindo-se *w=1.1*, *d=0.4* e *m=100*.

Outra questão importante que deve ser compreendida está relacionada ao efeito que ocorreu na simulação variando-se *w* visto na Figura 3.13(a). Diferentemente da simulação em que os nodos âncora estavam posicionados nos extremos do cenário, na qual o ganho que minimizou o NIM foi 2, percebe-se nesta figura que a partir do ganho igual a 1,1 o NIM aumenta, culminando numa variação extrema em *w=2*. Este efeito deve-se ao fato de que considerando a sequência de pontos gerados pelo método MVI como uma sucessão de vetores até o ponto de MRQ, a mudança de direção destes vetores só será suave (pequenas angulações entre si) para valores de *w* menores ou iguais a 1. Esta característica não causa problemas para nodos desconhecidos posicionados na área interna à localização dos âncoras, mas sim para os posicionados na externa.



(a) Simulação variando *w*

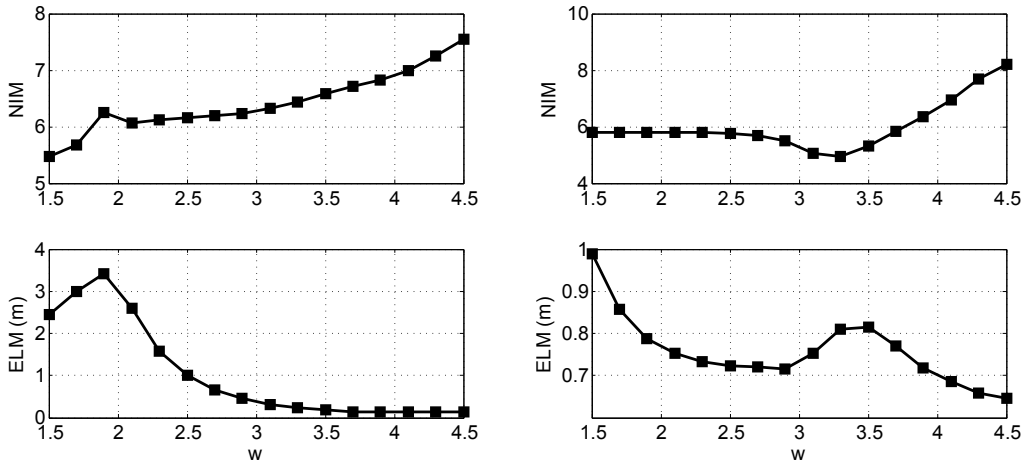
(b) Aproximação da região com a sequência de iterações oscilatórias.

Figura 3.18: Sequência de iterações do método MVI para um nodo desconhecido posicionado nas coordenadas *(20,0)* (marcado com um quadrado), com *w=2*, *d=0,4m* e *m=100*.

Para compreender melhor o que seria uma trajetória não suave, apresenta-se a sequência de iterações da simulação vista nas Figuras 3.16(b) e 3.17(b), apenas aumentando *w* de *1,1* para *2*, sendo os resultados apresentados nas Figuras 3.18(a) e 3.18(b). Destaca-se que o aumento do *w* gera um comportamento oscilatório, sendo este o pior caso de não suavidade da sequência de iterações, impedindo que o processo seja interrompido pela variável *d*, pois a distância entre consecutivos pontos continua sendo grande. Nestes casos, reduz-se o NIM do cenário limitando-se o número máximo de iterações permitidas.

Uma alternativa para reduzir o erro de localização ao custo de um menor aumento do número de iterações, evitando as oscilações vistas na Figura 3.18(b), é alternar o ganho entre 1 e

um valor maior a cada iteração. Porém, isto pode ser realizado de duas maneiras, uma mantendo as iterações pares e a outras as impares com ganho 1. Então, realizaram-se duas simulações, ainda para o cenário visto na Figura 3.12, sendo a primeira variando o ganho das iterações pares e a segunda das ímpares conforme as Figuras 3.19(a) e 3.19(b), respectivamente. Percebe-se que ambas possuem comportamentos distintos, de modo que a primeira permite uma maior redução do ELM e a segunda do NIM.

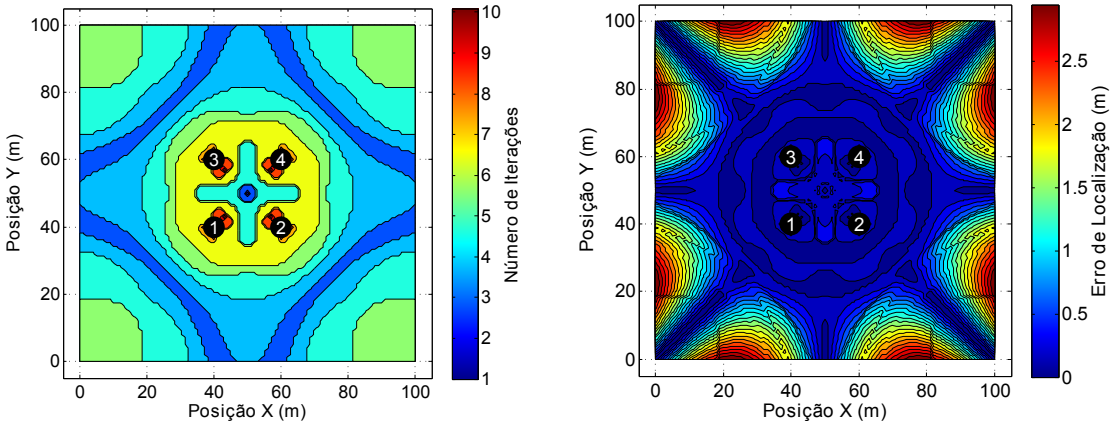


(a) Variação de *w* para as iterações ímpares. (b) Variação de *w* para as iterações pares.

Figura 3.19: Avaliação do comportamento do NIM e ELM no cenário da Figura 3.12 para um comportamento alternado de *w* com *d=0,4m* e *m=100*.

Escolheu-se o caso que minimiza o NIM visto na Figura 3.19(b), utilizando *w=3,3* nas iterações pares, para avaliar as distribuições do número de iterações e do erro de localização ao longo do cenário. Os resultados destas são apresentados pelas Figuras 3.20(a) e 3.20(b), respectivamente, destacando-se o elevado número de iterações no centro do cenário, se comparado à simulação com ganho igual a 2 vista na Figura 3.9(a) para o cenário com nodos âncora posicionados nos extremos.

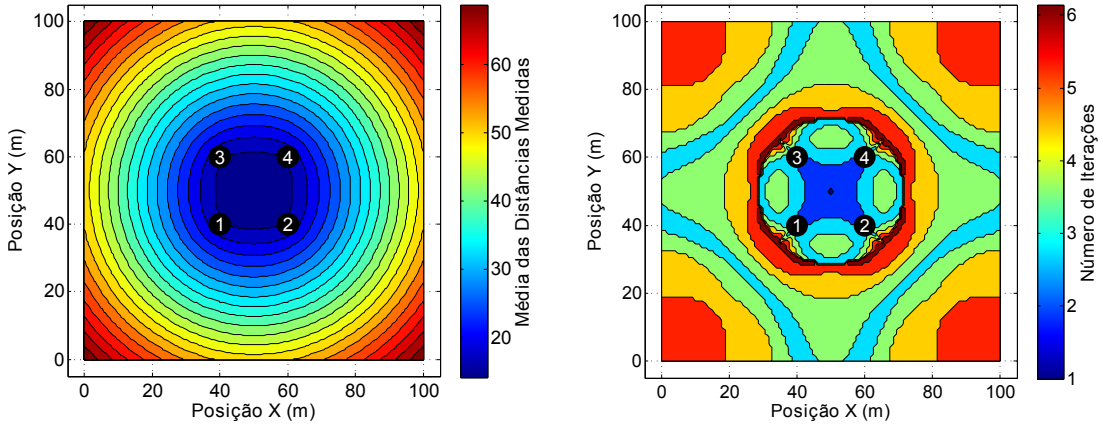
Deste modo, um meio para melhorar este problema de otimização do ganho ao longo de todo o cenário é dividi-lo em regiões de acordo com a média das distâncias medidas entre o nodo desconhecido em questão e os âncoras, diferenciando-se a região interna da externa aos âncoras. Para compreender esta questão, realizou-se uma simulação que calcula esta média para todos os nodos do cenário, conforme visto na Figura 3.21(a), na qual as curvas de nível representam pontos que possuem a mesma média. Assim, foi possível definir *w=2* para a região central, na qual a média das distâncias medidas em relação aos nodos âncora é menor que *24m*, e um *w* alternado, igual ao utilizado nas Figuras 3.20(a) e 3.20(b), para o restante do cenário.



(a) Número de iterações. (b) Erro de localização.

Figura 3.20: Distribuição do número de iterações e do erro de localização para o cenário da Figura 3.12 .

Como a distribuição do erro de localização permaneceu praticamente a mesma, é apresentado apenas a do número de iterações na Figura 3.21(b), destacando-se a redução deste na região com ganho igual a 2.



(a) Média das distâncias medidas em relação aos nodos âncora. (b) Número de iterações.

Figura 3.21: Simulação com configuração diferenciada do w a partir da média das distâncias medidas em relação aos nodos âncora para cenário visto na Figura 3.12.

Por fim, decidiu-se simular o algoritmo MVI para o cenário visto na Figura 3.12 com imprecisão nas medidas entre nodos desconhecidos e âncoras, utilizando as mesmas variáveis de controle da Figura 3.21(b). Os resultados são apresentados nas Figuras 3.22 e 3.23, percebendo-se que o erro de localização do método MVI acompanha o do MV com uma pequena diferença se comparado ao método MQL, e que o número de iterações aumenta proporcionalmente ao

aumento das imprecisões nas medidas de maneira similar à vista na Figura 2.10.

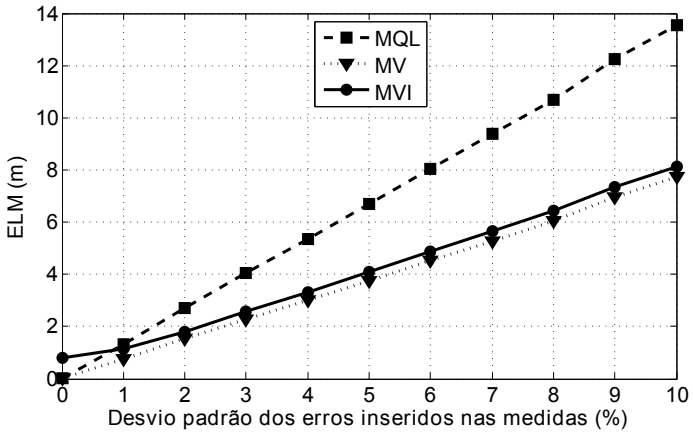


Figura 3.22: Comparação do ELM entre os métodos MQL, MVI e MV na presença de imprecisões nas medidas de distâncias entre nodos.

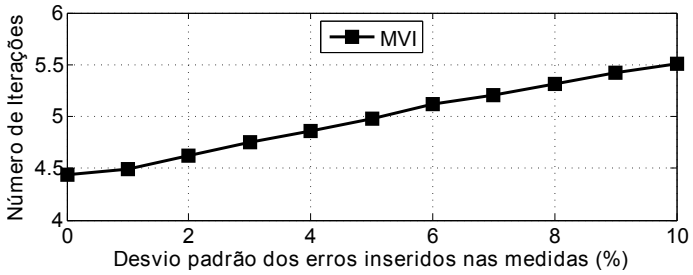


Figura 3.23: Comportamento do NIM na presença de imprecisões nas medidas de distâncias entre nodos.

O último cenário simulado visa mostrar que o posicionamento aleatório dos nodos âncora faz com que seja necessário modificar o ponto inicial do central para a posição do âncora mais próximo do nodo desconhecido. Isto pode ser evidenciado através da simulação do cenário visto na Figura 3.24 para estes dois caso de pontos iniciais, sendo que em ambos foram utilizadas as mesmas variáveis de controle. Como resultados destas simulações têm-se as posições estimadas dos nodos desconhecidos apresentadas nas Figuras 3.25(a) e 3.25(b), respectivamente. Destacam-se as regiões vistas na parte superior da primeira figura, inexistentes na segunda, nas quais os nodos desconhecidos não convergem para o ponto de MRQ do cenário devido ao ponto inicial não ser o adequado.

Para cenários em que há um grande número de nodos âncora disponível para a multilateração, pode-se reduzir a carga computacional iniciando-se esta com apenas 3 ou 4 deles, e após um determinado número de iterações adicionam-se os outros para se obter um refinamento da posição final com algumas iterações. Um bom critério para a escolha dos nodos utilizados

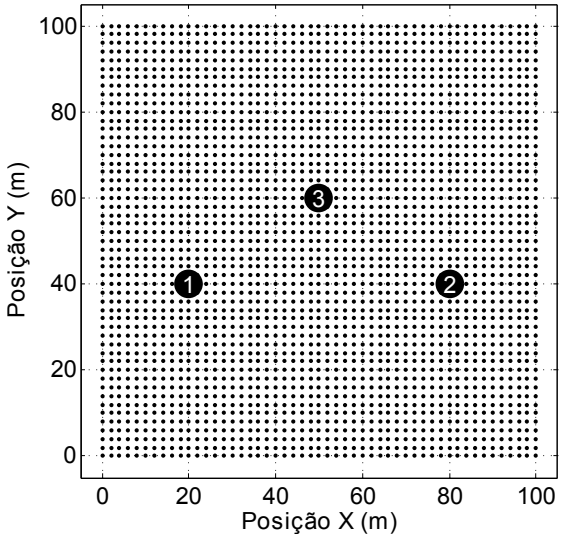
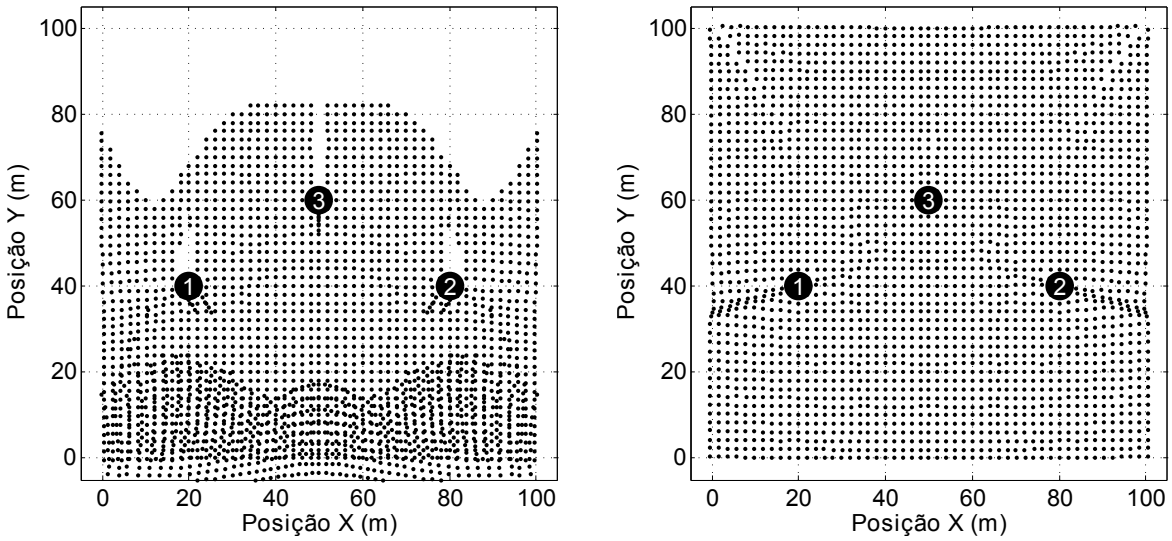


Figura 3.24: Terceiro cenário simulado para avaliar o comportamento do método de multilateração MVI.



(a) Simulação utilizando o centro geométrico dos nodos âncora como ponto inicial.

(b) Simulação utilizando a posição do nodo âncora com a menor distância medida em relação do nodo desconhecido com o ponto inicial.

Figura 3.25: Comparação da simulação do cenário visto na Figura 3.24 para os dois casos de pontos iniciais, sendo ambas com com as mesmas variáveis de controle.

no início da multilateração pode ser o espaçamento entre eles, pois quanto mais ao centro da área compreendia pelos âncoras estiver o nodo desconhecido, menor será o número de iterações necessárias e o erro de localização.

3.4 Redução de complexidade computacional do método de multilateração MVI

Nesta seção são detalhadas algumas estratégias para reduzir a carga computacional de cada iteração do método de multilateração MVI, sendo cada uma delas representadas por (3.1) e (3.2). Para facilitar a compreensão da carga computacional total de cada iteração, decidiu-se unir estas duas equações utilizando a representação das variáveis x e y na forma de um vetor $[x, y]$, resultando em

$$[x_{t+1}, y_{t+1}] = [x_t, y_t] + \frac{w}{n} \sum_{i=1}^{n} \left\{ [(x_t - x_i), (y_t - y_i)] \left(\frac{\hat{d}_i}{d_i} - 1 \right) \right\}. \tag{3.4}$$

Avaliar a complexidade computacional e o consumo de potência gerado por por uma determinada equação é uma tarefa fortemente dependente do hardware utilizado para sua execução, por exemplo, um sistema embarcado ou um circuito integrado dedicado. Assim, como uma implementação está fora do escopo deste trabalho, deseja-se apenas citar as principais características de algumas otimizações e os seus impactos na resolução de (3.4).

Conforme (KARP; MARKSTEIN, 1997) e (KWON; SONDEEN; DRAPER, 2007), as operações de divisão e raiz quadrada com números em ponto flutuante são consideravelmente mais complexas que as de adição e multiplicação, o que pode ser ser generalizado para implementações em ponto fixo, conforme (KARALAR et al., 2004). Já em (GRACIOLI et al., 2011), evitou-se apenas esta primeira operação devido à necessidade de utilização de determinadas bibliotecas matemáticas nos nodos. Deste modo, definiram-se como metas para as otimizações de (3.4) a eliminação do operador raiz quadrada e a minimização do operador de divisão.

O primeiro algoritmo estudado foi o CORDIC, do inglês *COordinate Rotation Dlgital Computer*, apresentado por (VOLDER, 1959) para determinar o módulo de um vetor, definido em duas dimensões, apenas com operações de soma e deslocamento. Este é um método iterativo que possui um total de iterações igual ao número de bits utilizado para representar as variáveis, sendo que em cada iteração são necessárias duas operações soma e deslocamento. Para simplificar o texto, entende-se por soma também as operações de subtração, visto que ambas possuem a mesma complexidade. O algoritmo CORDIC é uma boa opção para uma implementação em hardware, o que não pode ser generalizado para um microcontrolador sem uma devida comparação com outras técnicas.

Uma segunda opção para calcular a raiz quadrada é a utilização do método iterativo Newton-Raphson conforme (GRACIOLI et al., 2011), sendo que segundo o autor com até 20 iterações obtém-se uma precisão de uma unidade. Este é um método numérico clássico, con-

forme (PRESS et al., 2007) e (ERCEGOVAC; LANG, 2004), amplamente utilizado para redução de complexidade computacional na resolução de operações aritméticas e sistemas complexos, tendo como equação característica

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{3.5}$$

Para a implementação deste método, (GRACIOLI et al., 2011) utilizou a função $f(x) = x^2 - n$, sendo n o valor do qual se deseja determinar a raiz quadrada x, de modo que n deve ser um número real maior que zero. Aplicando esta função em (3.5) e executando algumas simplificações, encontra-se

$$x_{i+1} = 0.5\left(x_i + \frac{n}{x_i}\right). \tag{3.6}$$

Assim, cada iteração deste método necessita de uma operação de divisão, uma de soma e uma de deslocamento, sendo esta última equivalente a multiplicação por 0,5.

O número total de iterações de (3.6) necessárias para se atingir determinada precisão, neste caso definida como $0,1\%$ a partir de simulações, é dependente da precisão do valor inicial utilizado no processo. Conforme (ITO; TAKAGI; YAJIMA, 1997) e (ERCEGOVAC; LANG, 2004), um dos métodos padrões para se gerar estes valores é o armazenamento de uma série de constantes resultantes da execução ideal da raiz quadrada numa determinada memória, comumente chamada de *look-up table*. Ainda segundo estes autores, outra opção seria armazenar apenas coeficientes para uma aproximação linear por partes, ou seja, realizar sucessivas aproximações com polinômios de segunda ordem.

Neste trabalho, utiliza-se a raiz quadrada para determinar o módulo da soma de dois vetores (dx_i e dy_i) que partem da origem de um sistema em duas dimensões, conforme dado por

$$[dx_i, dy_i] = [(x_t - x_i), (y_t - y_i)] \tag{3.7}$$

e

$$d_i = \sqrt{{dx_i}^2 + {dy_i}^2}. \tag{3.8}$$

Assim, podem ser utilizadas propriedades trigonométricas para gerar um valor inicial para o método iterativo Newton-Raphson evitando o uso de uma *look-up table*. Então, realizou-se uma simulação variando o ângulo de 0 a 90 graus com d_i fixado como 1, conforme a Figura 3.26, para avaliar se a soma de dx_i e dy_i poderia ser utilizada como valor inicial.

O resultado desta simulação é mostrado na Figura 3.27, destacando-se que o erro máximo entre d_i e $(dx_i + dy_i)$ ocorre para o ângulo de 45^{o}, sendo este de 41%. Também, percebe-se

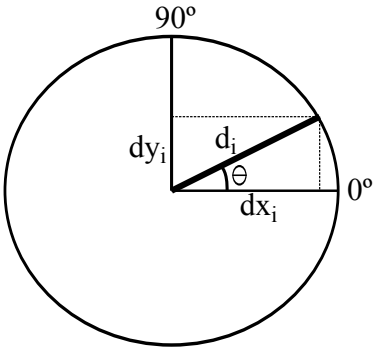


Figura 3.26: Variação do ângulo de 0 a 90 graus.

que o comportamento de 0^{o} a 45^{o} é o inverso de 45^{o} a 90^{o}, decidindo-se considerar apenas o primeiro intervalo nos próximos testes.

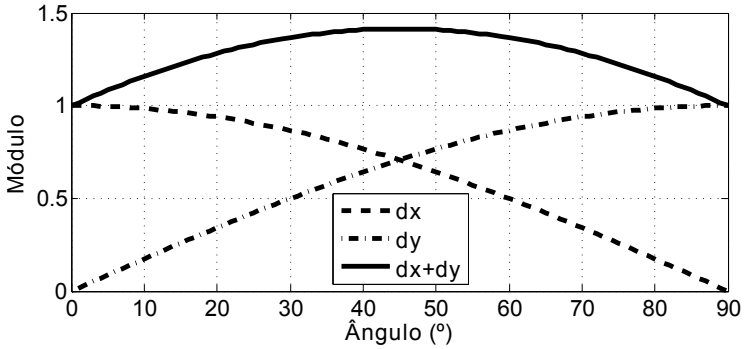


Figura 3.27: Comportamento de $(dx_i + dy_i)$ de 0 a 90 graus.

O método mais simples encontrado para reduzir o erro entre d_i e (dx_i+dy_i) foi modificar esta soma nos dois intervalos, sendo que de 0^{o} a 45^{o} é utilizado $(dx_i + dy_i/2)$ e de 45^{o} a 90^{o} $(dx_i/2 + dy_i)$. Para diferenciar os dois intervalos basta saber que no primeiro dx_i é maior que dy_i, e o contrário para o segundo. Esta modificação na soma não acarreta num aumento considerável da complexidade, pois a divisão por dois consiste apenas em um deslocamento de 1 bit para a direita. O resultado é apresentado na Figura 3.28, na qual percebe-se que o erro máximo reduziu para aproximadamente 12%.

Definido o valor inicial, o próximo passo é a execução das iterações do método Newton-Raphson conforme (3.6). Os resultados para 1 e 2 iterações são apresentados nas Figuras 3.29(a) e 3.29(b), respectivamente. Assim, com apenas duas iterações atinge-se um erro máximo de aproximadamente $0,002\%$, o que é 50 vezes menor que a meta de $0,1\%$.

Percebe-se que o método Newton-Raphson, representado por (3.6), utilizado para substituir a raiz quadrada é mais simples de ser implementado que o CORDIC, porém adiciona mais

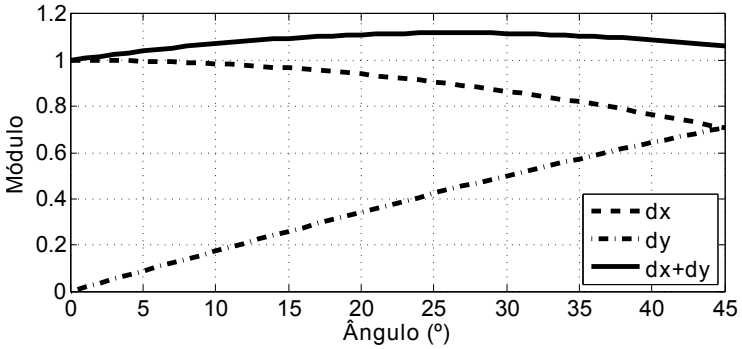
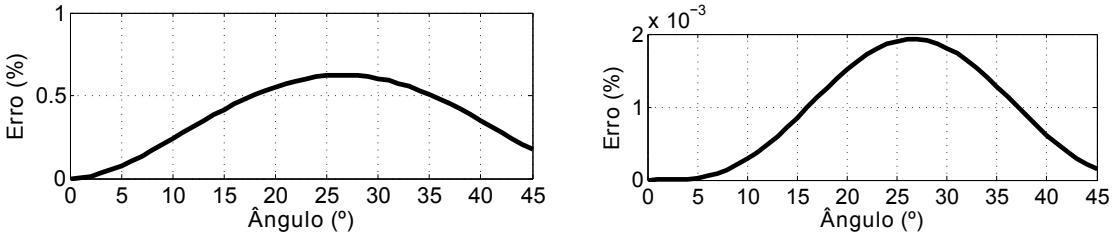


Figura 3.28: Comportamento de $(dx_i + dy_i/2)$ de 0 a 45 graus.



(a) Erro após uma iteração.

(b) Erro após duas iteração.

Figura 3.29: Erro entre o resultado ideal e o do método iterativo (3.6) após uma e duas iteração com ponto inicial definido pela Figura 3.28.

duas operações de divisão referentes às duas iterações necessárias, o que o torna computacionalmente menos eficiente. Conforme (KARP; MARKSTEIN, 1997) e (ERCEGOVAC; LANG, 2004), alterando-se a operação para a raiz quadrada recíproca $(1/\sqrt{n})$ é possível executar este método numérico apenas com operações de soma e multiplicação. Para este caso, utilizando a função $f(x) = 1/x^2 - n$, encontra-se a equação iterativa

$$x_{i+1} = 0.5x_i \left(3 - nx_i^2\right). \tag{3.9}$$

Para determinar a raiz quadrada através da sua recíproca basta realizar mais uma operação de multiplicação conforme

$$\sqrt{n} = \frac{1}{\sqrt{n}} \times n. \tag{3.10}$$

Assim, a principal vantagem de (3.9) para (3.6) é a substituição de uma divisão por 4 operações de multiplicação a cada iteração. Isto é considerado vantajoso para processadores e microcontroladores que possuam estruturas específicas para realizar multiplicações com poucos ciclos de *clock*. Como exemplo disto pode-se citar o microcontrolador ATmega128, conforme (ATMEL, 2013), que permite realizar uma multiplicação em apenas dois ciclos de *clock*, sendo que este

microcontrolador é utilizado no nodo comercial para RSSF MICAz, visto em (CROSSBOW, 2013).

Outra otimização que pode ser realizada no algoritmo MVI é a substituição da divisão por n, vista em (3.4), por uma *look-up table* com os resultados pré-calculados. Isto é viável pois n representa o número de nodos âncoras considerados na execução da multilateração, sendo necessariamente um número inteiro e limitado a apenas algumas unidades, como em (KARALAR et al., 2004) no qual limitou-se a 16.

Reescrevendo Equação (3.4) para (3.11), percebe-se que através da determinação da raiz quadrada recíproca é possível eliminar a operação de divisão que está dentro do somatório do método MVI. Isto, juntamente com a substituição de $1/n$ por uma *look-up table*, permite a eliminação da operação de divisão do método de multilateração MVI. Porém, a principal desvantagem da utilização do método Newton-Raphson para calcular a raiz quadrada recíproca é a necessidade de uma *look-up table* para determinação dos valores iniciais, pois não se consegue a geração destes valores de maneira semelhante a 3.28.

$$[x_{t+1}, y_{t+1}] = [x_t, y_t] + w\frac{1}{n}\sum_{i=1}^{n}\left\{[(x_t - x_i), (y_t - y_i)]\left(\hat{d}_i\frac{1}{\sqrt{d_i^2}} - 1\right)\right\} \tag{3.11}$$

4 SIMULADOR DE ALGORITMOS DE LOCALIZAÇÃO PARA REDES DE SENSORES SEM FIO BASEADO NO SOFTWARE MATLAB

Inicialmente, neste capítulo é detalhada a construção de um simulador para RSSF baseado na linguagem Matlab e dedicado a algoritmos de localização. Os principais motivos para a escolha deste software são a sua simplicidade de utilização, facilidade para a geração de gráficos e possibilidade da co-simulação com um bloco desenvolvido em linguagem de descrição de hardware (HDL do inglês *Hardware Description Language*) utilizando o software ModelSim da empresa Mentor Graphics, conforme (Mentor Graphics, 2012). Posteriormente, são realizadas simulações dos algoritmos de localização DV-Hop e Centroid com dois objetivos principais, sendo eles o de avaliar o desempenho do simulador e o de comparar o erro de localização entre os três métodos de multilateração (MQL, MV e MVI) descritos no Capítulo 3. Para este último caso, adicionou-se ao algoritmo Centroid a capacidade de determinação das distâncias entre nodos através da potência do sinal recebido, conforme visto na Seção 2.4.2.

Destaca-se que o simulador desenvolvido neste trabalho foi projetado para a simulação de qualquer algoritmo de localização, e não apenas para validar o método de multilateração desenvolvido no Capítulo 3. Deste modo, este capítulo foi dividido em duas seções, sendo na primeira detalhada a construção e as principais características do simulador, enquanto que na segunda simulam-se algoritmos de localização para avaliar tanto o desempenho deste quanto do método de multilateração desenvolvido.

4.1 Características do simulador

Como já mencionado na Seção 2.4, a linguagem Matlab não é otimizada para se trabalhar como algoritmos baseados em eventos se comparada ao C++, por exemplo. Deste modo, criar um simulador que opere com precisão equivalente à transmissão de um bit, conforme (SIMON et al., 2003), poderia ser inviável para a simulação de uma rede com um grande número de nodos. Estende-se por precisão o passo de tempo mínimo que será considerado pelo simulador na execução dos algoritmos. Então, no desenvolvimento do simulador é considerada uma precisão equivalente ao período do *superframe* no modo *beacon-enabled* do padrão IEEE 802.15.4, conforme já explicado na Seção 2.4.1.

Este período é denominado de ciclo, e assume-se que sempre será suficiente para suportar uma transmissão para cada nodo, de modo que estas só são recebidas em seus destinos

no próximo período. Assim, percebe-se que este é um simulador baseado no tempo, pois a cada ciclo é verificado se existem mensagens a serem entregues ou se os nodos desejam realizar alguma atividade, como processamentos ou transmissões.

O término da simulação pode ser definido pelo usuário de duas maneiras, sendo a primeira determinando um total de ciclos e a segunda de tempo. Este último refere-se a uma RSSF real e é denominado tempo virtual, obtido multiplicando-se o número de ciclos simulado pelo período que cada um representa. Conforme visto na Seção 2.4.1, a duração do *superframe* segue a equação $15,36 \times 2^{TC} ms$, sendo que TC é uma variável inteira que varia de 0 a 14, fazendo com que cada ciclo possa assumir os valores de tempo vistos na Tabela 4.1.

Tabela 4.1: Possíveis valores de tempo do *superframe* de acordo com o padrão IEEE 802.15.4.

TC	0	1	2	3	4	5	6	7
Tempo (s)	0.015	0.031	0.061	0.12	0.25	0.49	0.98	1.97
TC	8	9	10	11	12	13	14	
Tempo (s)	3.93	7.86	15.73	31.46	62.91	125.83	251.66	

O esquemático de funcionamento do simulador desenvolvido neste trabalho é apresentado na Figura 4.1, destacando-se que o laço criado entre a aplicação dos nodos e o canal de comunicação sem fio é interrompido quando o tempo virtual (t_{atual}) ultrapassar determinado valor predefinido (t_{total}). A partir deste esquemático percebe-se que o simulador pode ser dividido em quatro blocos principais, sendo eles:

- Gerador de cenários;
- Aplicação dos nodos;
- Canal de comunicação sem fio;
- Gerador de gráficos e resultados.

Destaca-se que este simulador foi desenvolvido em caráter de teste, tendo o objetivo maior ser utilizado como uma ferramenta para iniciantes devido à grande facilidade que a linguagem Matlab e a estrutura simplificada trazem. Todos os blocos deste simulador podem ser facilmente expandidos e integrados ao restante para se aprimorar determinadas características de acordo com os interesses do usuário.

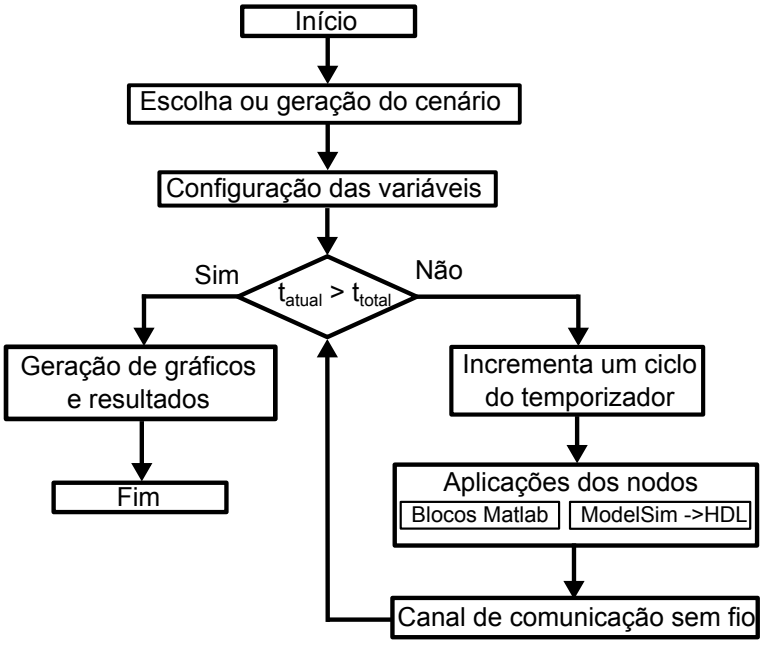


Figura 4.1: Esquemático de funcionamento do simulador.

4.1.1 Gerador de cenários

O bloco gerador de gráficos tem a função de auxiliar o usuário na rápida criação de cenários, sendo que para tal foi necessária a padronização da nomenclatura das variáveis, permitindo que estes fossem salvos e importados pelo simulador facilmente. O *script* criado para este bloco, assim como o resto do simulador, opera de maneira interativa, ou seja, após sua execução o usuário deve responder ao que é perguntado na janela de comandos, como as dimensões do cenário, número de nodos de cada tipo, o nome para salvamento, e assim por diante. Como principais opções deste bloco, destacam-se as de posicionamento automático dos nodos de maneira uniforme ou aleatória com determinada distância entre os nodos, e manual com a utilização do mouse, permitindo grande capacidade de customização dos cenários.

Já a questão da mobilidade não foi muito explorada neste simulador devido à sua complexidade e diversidade de modelos que poderiam ser implementados, conforme visto em (CAMP; BOLENG; DAVIES, 2002), e de ser necessário o estudo de algoritmos de localização próprios para esta condição. Então, agregou-se ao gerador de cenários apenas a capacidade de geração do trajeto de um nodo desconhecido com o auxílio do mouse e tendo a opção de definir a velocidade constante ou aleatória. O resultado disso é apresentado na Figura 4.2, na qual os pontos marcados com circunferências representam os nodos âncora e os com quadrados foram posi-

cionados com o mouse e definem o trajeto do nodo móvel. Esta velocidade aleatória é gerada através da função *rand* a cada intervalo entre dois pontos criados com o mouse, respeitando valores mínimos e máximos pré-definidos pelo usuário. Destaca-se que a mobilidade foi explorada no simulador apenas em caráter de teste, visando criar estruturas que permitirão adaptá-lo facilmente a cenários em que todos os nodos serão móveis no futuro.

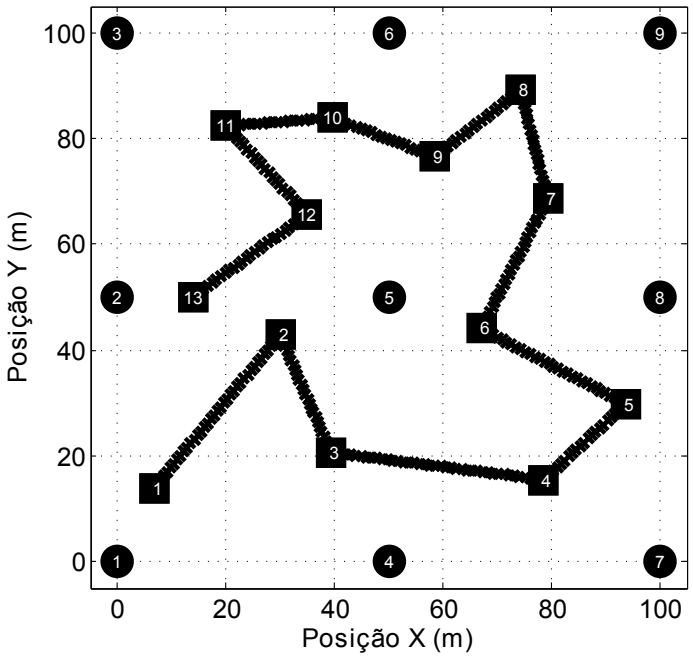


Figura 4.2: Criação do trajeto de um nodo móvel com o auxílio do mouse.

4.1.2 Aplicação dos nodos

O bloco de aplicação dos nodos é responsável pela execução de quaisquer tarefas atribuídas aos nodos. Este bloco é dividido em duas partes, sendo a primeira responsável pela configuração das variáveis de controle do algoritmo e a segunda pela criação das rotinas para cada tipo de nodo. Consideraram-se apenas dois tipos de nodos neste trabalho, denominados como de costume nodos âncora e desconhecidos. Para diferenciá-los, o simulador utiliza identificadores (IDs) sequenciais, de modo que os primeiros são reservados aos nodos âncora e o restante aos desconhecidos.

Uma das características mais importantes para aumentar a velocidade de simulação foi a pré-criação de variáveis, de acordo com (GETREUER, 2012), evitando assim que determinado nodo tenha que realocar espaço na memória do computador para aumentar ou criar novas variáveis durante a execução de suas rotinas. Deste modo, todas as variáveis do simulador são

inicializadas com zeros.

Para avaliar o comportamento e desempenho do simulador foram implementados os algoritmos de localização Centroid e DV-Hop, conforme visto na Seção 2.2. Uma importante característica deste primeiro algoritmo de localização é que o cálculo da posição do nó desconhecido é feito localmente baseado apenas na recepção de mensagens dos âncoras. Neste caso, não há troca de mensagens entre os nodos desconhecidos, os quais geralmente representam a maioria dos nodos do cenário. Isto motivou a implementação do algoritmo de localização DV-Hop para avaliar a capacidade do simulador de manipular trocas de mensagens entre todos os nodos da rede, visto que esta é uma das características mais marcantes deste algoritmo. Os resultados gerados com ambos os algoritmos são apresentados na Seção 4.2 durante a explicação de outras características do simulador.

4.1.3 Canal de comunicação sem fio

Este bloco é responsável pela aplicação do modelo de propagação do canal de comunicação sem fio log-normal com sombreamento, com função característica apresentada na Seção 2.4.2. Este bloco é relativamente simples se comparado aos outros devido à condição de não considerar conflitos, possuindo como funções determinar se uma mensagem será recebida e, em caso afirmativo, com qual potência.

Para melhorar seu desempenho, este bloco foi dividido em suas partes, sendo que na primeira é realizada apenas uma análise de conectividade entre o transmissor e o restante dos nodos do cenário. Esta distância de conectividade dos nodos é definida como sendo o limite em que a taxa de recepção de pacotes (PRR) se torna menor que 10%, segundo visto na Figura 2.16 da Seção 2.4.2, ou outro valor definido pelo usuário. Na segunda etapa é aplicada a equação do modelo de propagação para se obter um valor para a potência recebida no destino, a qual deverá ser maior que o limiar definido pelo usuário para que a mensagem seja entregue.

4.1.4 Gerador de gráficos e resultados

Por fim, tem-se o bloco gerador de gráficos e resultados com a função de permitir ao usuário uma rápida geração de resultados tanto gráficos como no formato de texto acerca do desempenho do algoritmo de localização executado em termos de erro de localização. No caso de um cenário com um nodo móvel, criou-se a possibilidade de gerar uma figura em três dimensões com o erro de localização, permitindo uma rápida visualização da distribuição do erro ao

longo do cenário.

4.2 Simulação de exemplos para avaliação do simulador e do método de multilateração desenvolvidos

Esta seção destina-se à simulação dos algoritmos de localização Centroid e DV-Hop com o intuito de avaliar a facilidade de uso e desempenho do simulador descrito neste capítulo, sendo estes dois algoritmos escolhidos devido à sua simplicidade e ampla utilização na literatura. Paralelamente, são comparados os resultados de ELM provenientes da utilização dos 3 métodos de multilateração vistos no Capítulo 3, em cenários com disposições uniformes e aleatórias dos nodos. A principal motivação destas comparações é provar a efetividade do método de multilateração MVI, desenvolvido neste trabalho, em cenários com distribuições espaciais dos nodos próximas da realidade.

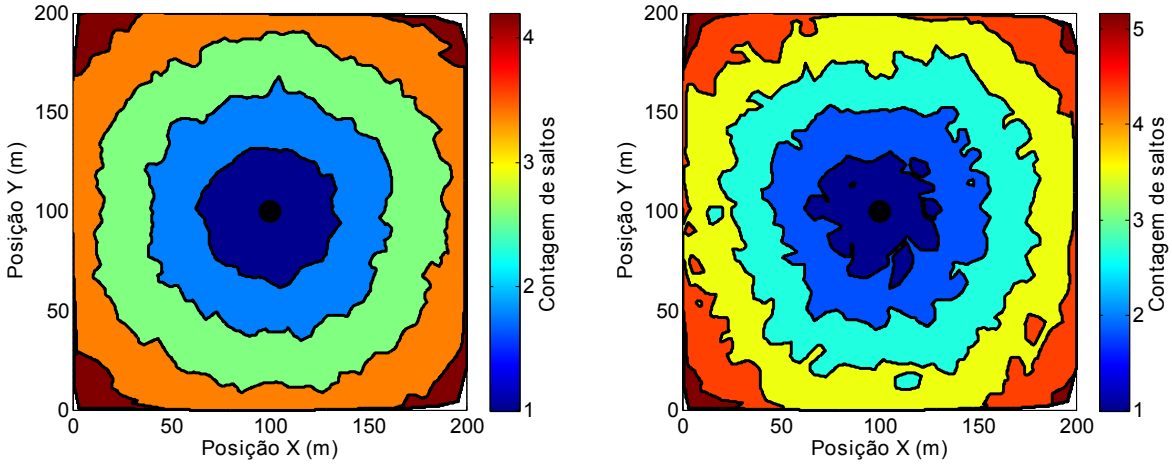
Para facilitar a compreensão dos resultados apresentados nesta seção, foi definida uma configuração padrão do modelo de propagação log-normal utilizada em todas as simulações, a qual é idêntica à vista na Figura 2.17 da Seção 2.4.2. Nesta, têm-se $P_t = 0dBm$, $d_0 = 1m$, $P_r(d_0) = -55dBm$, $\eta = 3$, $X_\sigma(dB) = 4dBm$, sensibilidade do receptor de $-95dBm$ e raio de alcance máximo de $31,9m$, acima do qual o PRR se torna menor que 10%. Para fins de avaliação dos efeitos deste modelo de propagação sobre os algoritmos de localização, algumas simulações utilizam um raio de conectividade ideal com esta mesma distância de $31,9m$.

4.2.1 Simulação do algoritmo de localização DV-Hop

O algoritmo de localização DV-Hop, conforme já detalhado na Seção 2.2, é caracterizado pela criação de estimativas de distância entre os nodos desconhecidos e os âncoras através da contagem das trocas de mensagens entre os nodos, comumente chamadas de saltos. O principal desafio na implementação deste algoritmo no simulador foi o fato de que todos os nodos recebem e transmitem mensagens, diferentemente do Centroid no qual os nodos âncora apenas transmitem e os desconhecidos recebem.

Para compreender melhor o comportamento da propagação dos contadores de saltos pela rede, foram realizadas duas simulações para um cenário quadrado com $200m$ de largura, um nodo âncora posicionado exatamente no centro e 1000 nodos desconhecidos posicionados aleatoriamente. Estas simulações diferem pela configuração do canal de comunicação sem fio,

pois na primeira considerou-se que os nodos possuem um raio de alcance ideal, enquanto que na segunda utilizou-se o modelo de propagação log-normal com sombreamento. Os resultados destas são vistos nas Figuras 4.3(a) e 4.3(b), respectivamente, destacando-se que a utilização do modelo log-normal aumenta a irregularidade da contagem dos saltos, o que deve reduzir consideravelmente a precisão do algoritmo DV-Hop.



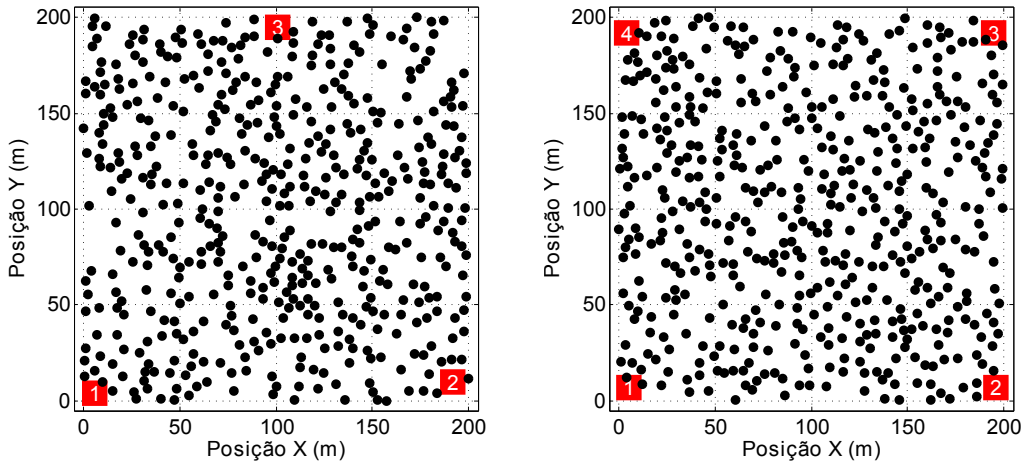
(a) Raio de conexão ideal.

(b) Modelo de propagação log-normal com sombreamento.

Figura 4.3: Propagação do contador de saltos a partir de um nodo âncora posicionado no centro do cenário.

Então, criaram-se os cenários vistos nas Figuras 4.4(a) e 4.4(b) para comparar o ELM encontrado a partir dos três métodos de multilateração utilizados na Seção 3.3. A única diferença entre os cenários é que o primeiro possui três nodos âncora e o segundo quatro, marcados com quadrados, e ambos possuem 500 nodos desconhecidos posicionados de maneira aleatória. Ambos os cenários foram simulados com as variáveis de controle do método de multilateração MVI sendo $w=2$, $d=0,4m$ e $m=20$.

Estes dois cenários foram simulados com o canal de comunicação sendo um raio de conectividade ideal e com o modelo de propagação log-normal, do mesmo modo que as Figuras 4.4(a) e 4.4(b). Para o primeiro caso foi necessário executar o algoritmo DV-Hop apenas uma vez, pois todos as estimativas apresentam o mesmo resultado. Já o resultado da segunda simulação é uma média de 100 estimativas de localização, visto que todas possuem resultados diferentes devido à aleatoriedade atrelada ao modelo de propagação em questão. Os resultados destas simulações são apresentados na Tabela 4.2, destacando-se que, conforme o esperado, os ELMs dos métodos MV e MVI são próximos entre si e consideravelmente menores que o do MQL. Também, deve-se ressaltar o aumento do erro de localização ao considerar o modelo de



(a) Cenário com 3 nodos âncora. (b) Cenário com 4 nodos âncora.

Figura 4.4: Cenários simulados para comparar os ELMs dos métodos de multilateração MQL, MV e MVI.

propagação log-normal.

Tabela 4.2: Resultados das simulações do algoritmo de localização DV-Hop nos cenários vistos nas Figuras 4.4(a) e 4.4(b).

Cenário	Modelo de propagação	ELM (m)			NIM do método MVI
		MQL	MV	MVI	
3 nodos âncora	Raio ideal	13,80	11,35	11,37	6,02
	Log-normal	15,28	14,49	14,46	6,08
4 nodos âncora	Raio ideal	10,34	9,21	9,19	4,62
	Log-normal	13,78	11,88	11,87	4,42

Para permitir a comparação do método MVI com o MV, considerando este último executado pela busca em grade, aproximou-se o custo computacional de uma iteração do primeiro com o da execução do cálculo do residual quadrático de dois ponto da grade no segundo método. Ao avaliar as equações que regem os dois métodos percebe-se que esta consideração é valida. Assim, para o cenário com 3 nós âncora, por exemplo, o método MVI teria o mesmo custo computacional que o MV com uma grade de apenas 12 pontos, a qual causaria um aumento no erro de localização devido ao baixo número de pontos.

4.2.2 Simulação do algoritmo de localização Centroid

Na implementação do algoritmo de localização Centoid, duas novas características do simulador foram avaliadas, sendo elas o processamento paralelo e a co-simulação de linguagem

de descrição de hardware. As análises destas foram realizadas durante a execução do algoritmo Centroid, implementado de acordo com a Seção 2.2, nas quais foram desconsiderados os resultados de erro de localização pois estavam fora do escopo destas simulações.

A utilização do processamento paralelo visando utilizar ao máximo a capacidade dos processadores com múltiplos núcleos, os quais estão cada vez mais populares e acessíveis, possui uma série de restrições para que realmente traga benefícios, isto para o software Matlab. Neste, pode-se tornar a execução de determinado *script* paralelizável substituindo o comando *for* pelo *parfor*, sendo que o primeiro consiste na execução repetitiva de uma rotina com o incremento gradual de determinada variável. Para a execução do segundo, esta rotina deve ser modificada de tal maneira que suas repetições sejam independentes entre si, ou seja, variáveis resultantes da execução de uma não podem interferir na outra. Caso a lógica do *script* permita tal condição, com o comando *parfor* o software Matlab distribui estas diferentes repetições da mesma rotina entre os núcleos disponíveis do processador.

Devesse ressaltar que o processo de distribuição das tarefas entre os núcleos demanda um tempo considerável, viabilizando-se assim apenas a execução em paralelo de rotinas extensas. Para exemplificar este problema, tem-se que o único modo viável de tornar o algoritmo Centroid paralelizável é considerar a execução de uma estimativa de localização como sendo uma rotina do *script*. Como principal limitação desta consideração, ressalta-se que não é possível utilizar dados resultantes de uma estimativa em outra, o que é muito comum em algoritmos de localização que refinam seus resultados a cada nova execução desta.

Realizou-se a simulação de um cenário quadrado com lateral medindo $100m$, 100 nodos âncora e 1000 desconhecidos distribuídos de maneira aleatória e considerando o modelo de propagação log-normal. O algoritmo de localização Centroid foi configurado para realizar uma estimativa de localização a cada 10 mensagens transmitidas pelos âncoras, o que consome exatamente 10 ciclos de simulação. Então, para um total de 200 estimativas de localização foram realizadas simulações variando o número de 1 a 4 núcleos de um processador de modelo i7-3612QM com frequência de operação igual a $2,1GHz$, segundo (INTEL, 2013), sendo os resultados apresentados na Figura 4.5.

Como principal resultado do trabalho com o processamento paralelo no software Matlab, destaca-se que existe uma redução considerável do tempo de simulação com o aumento do número de núcleos utilizados, porém ao custo de enrijecer a lógica do algoritmo, tornando a adição de funções a este algo muito trabalhoso e por vezes inviável. Deste modo, compreende-

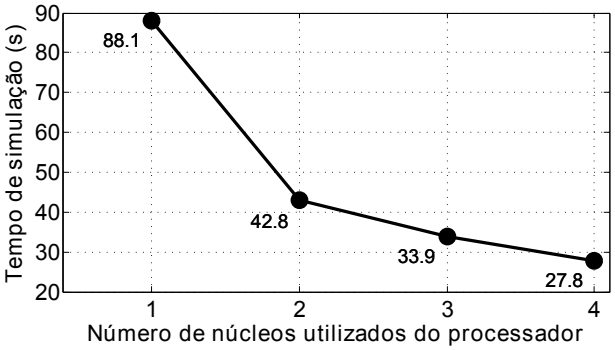


Figura 4.5: Redução do tempo de simulação com a utilização do processamento paralelo.

se que a melhor utilização desta característica ocorre numa fase final de projeto, em que se desejam realizar uma série de simulações independentes variando alguns parâmetros com a finalidade de avaliar a robustez do algoritmo sobre uma grande quantidade de condições.

Já a co-simulação abordada neste trabalho, por sua vez, visa reduzir os esforços necessários na verificação de determinado algoritmos de localização projetado HDL, conforme (KARALAR et al., 2004) e (OLIVEIRA, 2009), pois permite a simulação destes dentro de uma RSSF. O software utilizado neste trabalho para a simulação do HDL é o ModelSim, visto que utiliza-se mais especificamente a linguagem VHDL, cuja sigla é originária do inglês e representa *VHSIC Hardware Description Language*, sendo que VHSIC significa circuitos integrados de alta velocidade ou, do inglês, *Very High Speed Integrated Circuits*.

A comunicação entre os softwares Matlab e o ModelSim é realizada com o auxílio de um bloco específico do Simulink, denominado *HDL Cosimulation*, o qual controla a troca de variáveis entre estes dois softwares de dois modos diferentes. No primeiro, a troca de variáveis é realizada através de um soquete de rede, semelhante ao visto em em (DIDIOUI et al., 2013), enquanto que o segundo consiste no compartilhamento de uma memória, de modo que escolheu-se este último por ser mais rápido. Destaca-se que cabe ao Matlab a função de controlar a simulação da RSSF como um todo, enquanto que o ModelSim é responsável apenas pela execução de determinado código VHDL por um tempo determinado por aquele.

Neste trabalho implementou-se o algoritmo de localização Centroid em linguagem VHDL de maneira idêntica à apresentada na Seção 2.2. O esquemático Simulink visto na Figura 4.6 é o responsável pela adequação das variáveis que transitam entre os softwares Matlab e ModelSim, visto que no primeiro trabalha-se com ponto flutuante e no segundo, fixo.

A principal peculiaridade encontrada na co-simulação abordada neste trabalho é o tempo

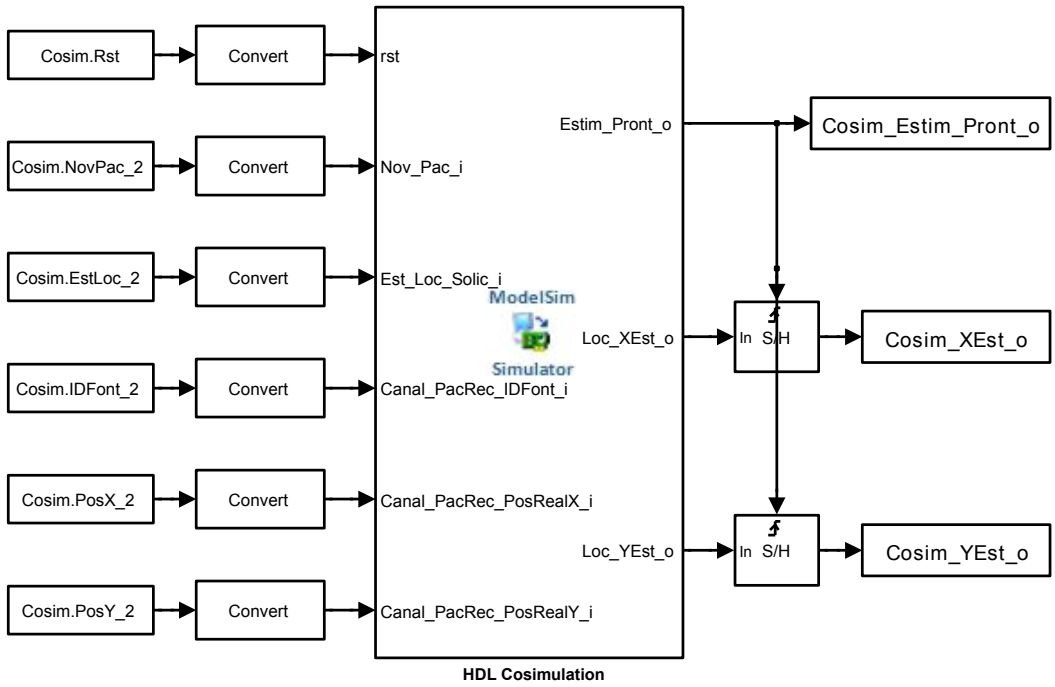


Figura 4.6: Bloco no software Simulink responsável pela interligação entre o Matlab e o ModelSim.

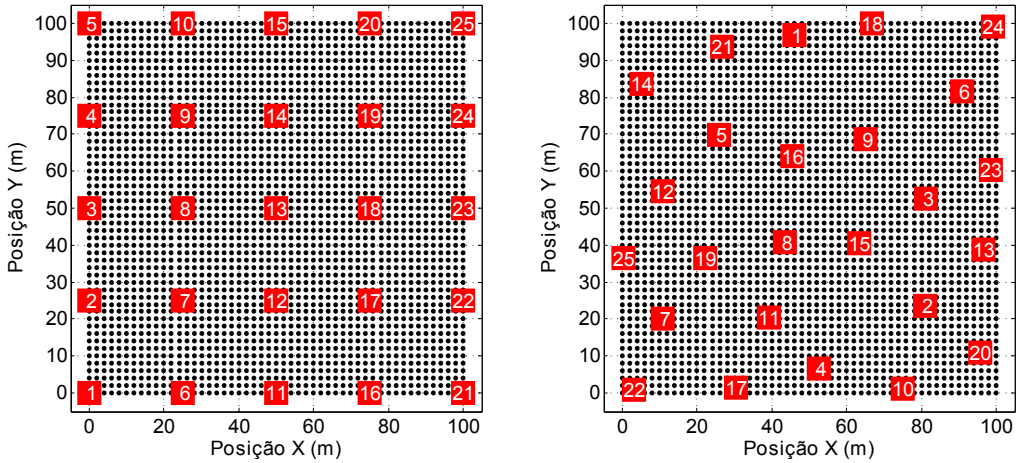
necessário para apenas iniciar e concluir a comunicação entre os softwares envolvidos uma única vez, o qual oscila em torno de $170ms$. Para fins de comparação, a realização de uma estimativa de localização com determinada configuração necessita de aproximadamente 200 ciclos de relógio no código VHDL, o que demora em torno de $40ms$ para ser executada no ModelSim. Está-se ciente de que estes tempos variam de acordo com as configurações do algoritmo e do computador utilizado, porém, o objetivo destas informações é ressaltar que a melhor forma de reduzir o tempo de simulação da RSSF é minimizar o número de chamadas do software ModelSim.

Tendo isto em mente e considerando que o código VHDL representa o algoritmo de localização de apenas um nodo desconhecido, definiram-se dois modos de operação. No primeiro, o usuário define o nodo desconhecido que será co-simulado, então as mensagens que este receber são acumuladas e enviadas para o ModelSim apenas quando for solicitada a execução da estimativa de localização, ou seja, só quando os dados de entrada forem suficientes para que o código VHDL gere algum resultado. Já no segundo, são acumulados sequencialmente os dados de entrada de cada nodo para realização das estimativas de localização, permitindo assim a co-simulação de todos estes em uma única chamada do ModelSim.

Deve-se compreender, que este último modo de operação só foi possível pelo fato de

que todas as variáveis do algoritmo Centroid podem ser descartadas após o término de uma estimativa de localização. Por fim, na questão da co-simulação, para verificar se os resultados vindos do código VHDL estão corretos, é realizada uma comparação entre estes e os seus respectivos vindos do modelo Matlab, de modo que os erros encontrados sejam reportados na forma de texto.

A última tarefa descrita nesta seção é a simulação dos dois cenários vistos nas Figuras 4.7(a) e 4.7(b) com a finalidade de comparar os ELMs resultantes das execuções dos métodos de multilateração MQL, MV e MVI. No primeiro cenário os 25 nodos âncora estão dispostos de maneira uniforme com espaçamento lateral de $25m$, o qual foi definido com base no alcance do modelo de propagação log-normal utilizado, de modo que praticamente todos os nodos desconhecidos estejam no raio de alcance de nodos âncora. Já no segundo, posicionaram-se os mesmos 25 nodos âncora de maneira aleatória com o distanciamento máximo possível entre si de $19m$, pois acima deste valor o algoritmo gerador de posições não consegue encontrar 25 posições que respeitem tal espaçamento em um cenário quadrado com lados medindo $100m$. Já os nodos desconhecidos foram posicionados de maneira uniforme com espaçamento lateral de $2m$, totalizando 2592 no primeiro e 2601 no segundo cenário, sendo esta diferença proveniente da distância mínima de $1m$ entre nodos âncora e desconhecidos exigida pelo modelo de propagação. Para este ambos os cenários, o valor de *w=1,7* minimizou o número de iteração do método MVI, sendo as outras variáveis de controle definidas como sendo *d=0,5m* e *m=20*.



(a) Cenário com 25 nodos âncora dispostos de maneira uniforme com espaçamento lateral entre si de $25m$.

(b) Cenário com 25 nodos âncora dispostos de maneira aleatória com espaçamento lateral entre si de no mínimo $19m$.

Figura 4.7: Cenários simulados para comparar os ELMs dos métodos de multilateração MQL, MV e MVI.

Nestas simulações, modificou-se o algoritmo Centroid de modo que os valores de potências recebidos fossem acumulados juntos com as contagens das mensagens recebidas de cada nodo âncora, definindo-se um total de 10 mensagens transmitidas por cada um destes para se realizar uma estimativa de localização. Para se reduzir os erros nas estimativas de distância geradas através dos valores de RSSI, conforme visto na Seção 2.4.2, são escolhidos para a execução da multilateração apenas os nodos de referência dos quais receberam-se no mínimo metade das mensagens esperadas. Porém, caso não fossem encontrados no mínimo 3 nodos âncora que satisfizessem tal condição, esta é eliminada com o objetivo de minimizar o erro de localização através da utilização do máximo de referências possíveis.

Além dos resultados de ELM e NIM para estas simulações novos relatórios foram criados no simulador para avaliar os métodos de multilateração utilizados, sendo o primeiro deles o número de nodos cujas estimativas tiveram que ser descartadas por causa de erros de localização muito grandes no método MQL. Estes ocorrem devido às condições em que são realizadas divisões por valores muito próximos ou iguais a zero, destacando-se que mesmo nestes casos os métodos MV e MVI operam corretamente. Os outros dois relatórios são o número médio de referências considerado na execução das multilaterações e o de desconhecidos que não conseguiram se localizar por não receberem no mínimo uma mensagem de três nodos âncora distintos.

Todos estes resultados são apresentados na Tabela 4.3 e representam uma média de 100 estimativas de localização dos cenários vistos nas Figuras 4.7(a) e 4.7(b). Destaca-se que, semelhante ao constatado nas simulações realizadas na Seção 4.2.1, os resultados de ELMs dos métodos MV e MVI são praticamente iguais e consideravelmente menores que os do MQL. Também, deve-se ressaltar que a disposição aleatória dos nodos âncora provoca uma piora considerável em todos os resultados se comparado à disposição uniforme. Os principais motivos para tal comportamento são a redução significativa no número médio de âncoras utilizados nas multilaterações e o surgimento de regiões nas quais ocorre uma baixa taxa de recepção de mensagens. Por fim, conseguiu-se provar que o método de multilateração MVI, desenvolvido neste trabalho, possui uma precisão equiparável ao método MV, ao custo de um número de iterações computacionalmente equiparável ao cálculo do residual quadrático de aproximadamente 7 pontos de um busca em grade para o cenário aleatório.

Tabela 4.3: Resultados das simulações do algoritmo de localização Centroid modificado para gerar estimativas de distância a partir de medidas de RSSI nos cenários vistos nas Figuras 4.7(a) e 4.7(b).

		Disposição dos nodos âncora	
		Uniforme	Aleatório
ELM (m)	MQL	4,80	4,94
	MV	3,64	3,73
	MVI	3,64	3,72
NIM do método MVI		2,83	3,39
Média de referências utilizadas		3,99	3,68
Média de erros do método MQL		7,45	38,90
Média de nodos que não se localizaram		2,35	46,20

5 CONCLUSÃO

Neste trabalho foi proposto um novo método de multilateração iterativo para algoritmos de localização em RSSF, tendo como principais objetivos a redução de complexidade computacional com o mínimo de perda de precisão. Através da simulação de cenários com disposição aleatória dos nodos e considerando o modelo de propagação log-normal com sombreamento para a transmissão de mensagens e estimação das distâncias, percebeu-se que este método é robusto e apresenta um erro de localização muito próximo do ideal.

Também foi desenvolvido um simulador para algoritmos de localização em RSSF baseado no software Matlab, utilizado na avaliação do método de multilateração criado. Como principal característica deste simulador destaca-se a co-simulação de linguagem de descrição de hardware, permitindo assim avaliar o comportamento de um algoritmo de localização implementado em hardware dentro de uma RSSF.

5.1 Trabalhos futuros

5.1.1 Método de multilateração MVI

Um dos principais trabalhos futuros para o método de multilateração MVI é a realização da comparação com o método MQL ou outras implementações do MV em termos de consumo de potência e precisão, após a utilização de variáveis em ponto fixo, para implementações em hardware ou microcontroladores. Neste último caso também seria interessante avaliar a quantidade de memória de instrução e execução consumida, visto que os nodos possuem recursos energéticos e computacionais extremamente limitados.

Outra opção para o método de multilateração MVI seria procurar formas para reduzir o número de iterações através do fornecimento de um ponto inicial mais preciso para o processo iterativo, sendo uma opção para tal o método Min-Max utilizado em (GRACIOLI et al., 2011). Também seria interessante procurar relações entre o método MVI e o gradiente, almejando a utilização de propriedades matemáticas na otimização daquele. Por fim, poderia-se ainda avaliar o comportamento do método MVI em ambientes com mobilidade.

5.1.2 Simulador para algoritmos de localização em RSSF

Destaca-se como trabalho futuro para o simulador a implementação da mobilidade para todos os nodos do cenário, de modo que as posições sejam geradas seguindo os padrões de mobilidade mais utilizados na literatura, sendo um dos principais o conhecido como *random walk*. Esta implementação poderia ocorre de dois modos distintos, sendo o primeiro através da criação de todas as posições dos nodos para todos os instantes de tempo no bloco de criação de cenários. Porém, para simulações com grande quantidade de nodos ou por um período de tempo prolongado, este modo exigiria o armazenamento de uma grande quantidade de dados. Assim, para este tipo de simulação sugere-se um segundo modo no qual a criação das posições dos nodos ocorre durante a simulação.

Outra característica interessante para se inserir no simulador seria o trabalho com colisões de mensagens. Porém, este aspecto é um dos mais complexos para se implementar, pois exige uma redução do passo de simulação para algo próximo do período de transmissão de um bit, permitindo assim a implementação da camada MAC do protocolo IEEE 802.15.4. Como isto inviabilizaria a simulação de uma RSSF com muitos nodos tendo como base o software Matlab, teria-se que criar um simulador para fins específicos, como, por exemplo, para avaliar características de sincronismo ou o consumo de potência dos nodos.

Por último, poderiam ser aprimorados os relatórios gerados pelo simulador, trazendo mais detalhes acerca das transmissões e dos passos intermediárias dos algoritmos. Assim, com base nesses dados poderiam ser criadas métricas para se estimar o consumo de potência de um determinado algoritmo.

REFERÊNCIAS

21 ideas for the 21st century. **Business Week**, [S.l.], p.78–167, aug 1999.

AKYILDIZ, I. et al. A survey on sensor networks. **Communications Magazine, IEEE**, [S.l.], v.40, n.8, p.102 – 114, aug 2002.

ALI, Q.; ABDULMAOWJOD, A.; MOHAMMED, H. Simulation amp; performance study of wireless sensor network (WSN) using MATLAB. In: ENERGY, POWER AND CONTROL (EPC-IQ), 2010 1ST INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.307–314.

ANDEL, T.; YASINSAC, A. On the credibility of manet simulations. **Computer**, [S.l.], v.39, n.7, p.48 –54, july 2006.

ATMEL. **ATmega128**. Acessado em Março/2013, http://www.atmel.com/Images/doc2467.pdf.

B. HOFMANN-WELLENHOF, H. L.; COLLINS, J. **Global Positioning System**: theory and practice. 4.ed. [S.l.]: Springer Verlag, 1997.

BACHIR, A. et al. MAC Essentials for Wireless Sensor Networks. **Communications Surveys Tutorials, IEEE**, [S.l.], v.12, n.2, p.222–248, 2010.

BAHL, P.; PADMANABHAN, V. RADAR: an in-building rf-based user location and tracking system. In: INFOCOM 2000. NINETEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES. PROCEEDINGS. IEEE. **Anais...** [S.l.: s.n.], 2000. v.2, p.775 –784 vol.2.

BJORCK, A. **Numerical Methods For Linear Least Squares Problems**. Philadelphia: SIAM., 1996.

BORN, A.; BILL, R. Reducing the Calculation for Precise Localization in Wireless Sensor Networks. In: VEHICULAR TECHNOLOGY CONFERENCE (VTC 2010-SPRING), 2010 IEEE 71ST. **Anais...** [S.l.: s.n.], 2010. p.1–5.

BULUSU, N.; HEIDEMANN, J.; ESTRIN, D. GPS-less low-cost outdoor localization for very small devices. **Personal Communications, IEEE**, [S.l.], v.7, n.5, p.28 –34, oct 2000.

CALLAWAY, E. et al. Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. **Communications Magazine, IEEE**, [S.l.], v.40, n.8, p.70–77, 2002.

CAMP, T.; BOLENG, J.; DAVIES, V. A Survey of Mobility Models for Ad Hoc Network Research. **WIRELESS COMMUNICATIONS & MOBILE COMPUTING (WCMC): SPECIAL ISSUE ON MOBILE AD HOC NETWORKING: RESEARCH, TRENDS AND APPLICATIONS**, [S.l.], v.2, p.483–502, 2002.

CHEN, Y.-S.; CHIN, T.-L.; HUANG, Y.-C. Collaborative localization in Wireless Sensor Networks based on dependable RSSI. In: WIRELESS PERSONAL MULTIMEDIA COMMUNICATIONS (WPMC), 2012 15TH INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2012. p.341–347.

CHONG, C.-Y.; KUMAR, S. Sensor networks: evolution, opportunities, and challenges. **Proceedings of the IEEE**, [S.l.], v.91, n.8, p.1247–1256, 2003.

CROSSBOW. **MICAz**. Acessado em Março/2013, http://www.openautomation.net.

DEMMEL, J. W. **Applied Numerical Linear Algebra**. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics., 1997.

DIDIOUI, A. et al. HarvWSNet: a co-simulation framework for energy harvesting wireless sensor networks. In: COMPUTING, NETWORKING AND COMMUNICATIONS (ICNC), 2013 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.808–812.

Documentação NS-2. **The Network Simulator ns-2: documentation**. Acessado em Agosto/2012, http://www.isi.edu/nsnam/ns/ns-documentation.htm.

EDWARDS, C. Smart dust. **Engineering Technology**, [S.l.], v.7, n.6, p.74–77, 2012.

EGEA-LOPEZ, E. et al. Simulation scalability issues in wireless sensor networks. **Communications Magazine, IEEE**, [S.l.], v.44, n.7, p.64–73, 2006.

ERCEGOVAC, M. D.; LANG, T. **Digital Arithmetic**. [S.l.]: Morgan Kaufmann Publishers, 2004.

FRABOULET, A.; CHELIUS, G.; FLEURY, E. Worldsens: development and prototyping tools for application specific wireless sensors networks. In: INFORMATION PROCESSING

IN SENSOR NETWORKS, 2007. IPSN 2007. 6TH INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2007. p.176–185.

GETREUER, P. **Writing Fast Matlab Code**. Acessado em Setembro/2012, http://www.mathworks.com/matlabcentral/fileexchange/5685-writing-fast-matlab-code.

GRACIOLI, G. et al. Evaluation of an RSSI-based Location Algorithm for Wireless Sensor Networks. **Latin America Transactions, IEEE (Revista IEEE America Latina)**, [S.l.], v.9, n.1, p.830–835, 2011.

GREENORBS. Acessado em Março/2013, http://www.greenorbs.org.

GUSTAFSSON, F.; GUNNARSSON, F. Positioning using time-difference of arrival measurements. In: ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 2003. PROCEEDINGS. (ICASSP '03). 2003 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. v.6, p.VI–553–6 vol.6.

HASAN, M. et al. Co-simulation of wireless networked control systems over mobile ad hoc network using SIMULINK and OPNET. **Communications, IET**, [S.l.], v.3, n.8, p.1297–1310, 2009.

HU, L.; EVANS, D. Localization for mobile sensor networks. In: MOBILE COMPUTING AND NETWORKING, 10., New York, NY, USA. **Proceedings...** ACM, 2004. p.45–57. (MobiCom '04).

IEEE. **IEEE std. 802.15.4 - 2003 Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)**. Acessado em Fevereiro/2013, http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

IMRAN, M.; SAID, A.; HASBULLAH, H. A survey of simulators, emulators and testbeds for wireless sensor networks. In: INFORMATION TECHNOLOGY (ITSIM), 2010 INTERNATIONAL SYMPOSIUM IN. **Anais...** [S.l.: s.n.], 2010. v.2, p.897–902.

INTEL. **Intel Core i7-3612QM Processor**. Acessado em Janeiro/2013, http://ark.intel.com/pt-br/products/67356/Intel-Core-i7-3612QM-Processor-6M-Cache-up-to-3_10-GHz-rPGA.

ITO, M.; TAKAGI, N.; YAJIMA, S. Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification. **Computers, IEEE Transactions on**, [S.l.], v.46, n.4, p.495–498, 1997.

J.W. GARDNER, V. K. V.; AWADELKARIM, O. O. **Microsensors, MEMS and Smart Devices**. 1.ed. New York: Wiley., 2001.

KAHN, J. M.; KATZ, R. H.; PISTER, K. S. J. Next century challenges: mobile networking for *Smart Dust*. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, 5., New York, NY, USA. **Proceedings...** ACM, 1999. p.271–278. (MobiCom '99).

KARALAR, T. et al. An integrated, low power localization system for sensor networks. In: MOBILE AND UBIQUITOUS SYSTEMS: NETWORKING AND SERVICES, 2004. MOBIQUITOUS 2004. THE FIRST ANNUAL INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. p.24–30.

KARP, A. H.; MARKSTEIN, P. High-precision division and square root. **ACM Trans. Math. Softw.**, New York, NY, USA, v.23, n.4, p.561–589, Dec. 1997.

KHAN, M. et al. Limitations of Simulation Tools for Large-Scale Wireless Sensor Networks. In: ADVANCED INFORMATION NETWORKING AND APPLICATIONS (WAINA), 2011 IEEE WORKSHOPS OF INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.820–825.

KIESS, W.; MAUVE, M. A Survey on Real-world Implementations of Mobile Ad-hoc Networks. **Ad Hoc Netw.**, Amsterdam, The Netherlands, The Netherlands, v.5, n.3, p.324–339, Apr. 2007.

KOHTAMAKI, T. et al. PiccSIM Toolchain - design, simulation and automatic implementation of wireless networked control systems. In: NETWORKING, SENSING AND CONTROL, 2009. ICNSC '09. INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2009. p.49–54.

KORKALAINEN, M. et al. Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications. In: NETWORKING AND SERVICES, 2009. ICNS '09. FIFTH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2009. p.102–106.

KOTZ, D. et al. Experimental Evaluation of Wireless Simulation Assumptions. In: ACM INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS AND SIMULATION OF WIRELESS AND MOBILE SYSTEMS, 7., New York, NY, USA. **Proceedings...** ACM, 2004. p.78–82. (MSWiM '04).

KOUBAA, A.; ALVES, M.; TOVAR, E. A comprehensive simulation study of slotted CSMA/CA for IEEE 802.15.4 wireless sensor networks. In: FACTORY COMMUNICATION SYSTEMS, 2006 IEEE INTERNATIONAL WORKSHOP ON. **Anais...** [S.l.: s.n.], 2006. p.183–192.

KUNG, H. T. et al. Localization with snap-inducing shaped residuals (SISR): coping with errors in measurement. In: MOBILE COMPUTING AND NETWORKING, 15., New York, NY, USA. **Proceedings...** ACM, 2009. p.333–344. (MobiCom '09).

KWON, T.-J.; SONDEEN, J.; DRAPER, J. Floating-point division and square root using a Taylor-series expansion algorithm. In: CIRCUITS AND SYSTEMS, 2007. MWSCAS 2007. 50TH MIDWEST SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2007. p.305–308.

LEVIS, P. et al. TOSSIM: accurate and scalable simulation of entire tinyos applications. In: EMBEDDED NETWORKED SENSOR SYSTEMS, 1., New York, NY, USA. **Proceedings...** ACM, 2003. p.126–137. (SenSys '03).

LI, Z. et al. Robust statistical methods for securing wireless localization in sensor networks. In: INFORMATION PROCESSING IN SENSOR NETWORKS, 2005. IPSN 2005. FOURTH INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2005. p.91–98.

MACK, C. Fifty Years of Moore's Law. **Semiconductor Manufacturing, IEEE Transactions on**, [S.l.], v.24, n.2, p.202–207, 2011.

MATHWORKS. **MATLAB and Simulink for Technical Computing**. Acessado em Março/2012, www.mathworks.com.

Mentor Graphics. **ModelSim**. Acessado em Maio/2012, http://www.mentor.com/products/fv/modelsim/.

MOZUMDAR, M. et al. A Framework for Modeling, Simulation and Automatic Code Generation of Sensor Network Application. In: SENSOR, MESH AND AD HOC COMMUNICA-

TIONS AND NETWORKS, 2008. SECON '08. 5TH ANNUAL IEEE COMMUNICATIONS SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.515 –522.

NICULESCU, D.; NATH, B. Ad hoc positioning system (APS). In: GLOBAL TELECOMMUNICATIONS CONFERENCE, 2001. GLOBECOM '01. IEEE. **Anais...** [S.l.: s.n.], 2001. v.5, p.2926–2931 vol.5.

NICULESCU, D.; NATH, B. Ad hoc positioning system (APS) using AOA. In: INFOCOM 2003. TWENTY-SECOND ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS. IEEE SOCIETIES. **Anais...** [S.l.: s.n.], 2003. v.3, p.1734–1743 vol.3.

NS-2. **The Network Simulator - ns-2**. Acessado em Agosto/2012, http://www.isi.edu/nsnam/ns.

OLIVEIRA, L. L. **Algoritmo de Localização de Nodos para Redes de Sensores Móveis**. Tese (Doutorado) — Universidade Federal de Santa Maria. 2009.

OMNET++. **Omnet++ Discrete event simulation system**. Acessado em Agosto/2012, http://www.omnetpp.org.

OPNET. **OPNET**. Acessado em Agosto/2012, http://www.opnet.com.

ORFANUS, D. et al. Performance of Wireless Network Simulators: a case study. In: ND ACM WORKSHOP ON PERFORMANCE MONITORING AND MEASUREMENT OF HETEROGENEOUS WIRELESS AND WIRED NETWORKS, 3., New York, NY, USA. **Proceedings...** ACM, 2008. p.59–66. (PM2HW2N '08).

PANTAZIS, N.; VERGADOS, D. A survey on power control issues in wireless sensor networks. **Communications Surveys Tutorials, IEEE**, [S.l.], v.9, n.4, p.86–107, 2007.

PATWARI, N. et al. Locating the nodes: cooperative localization in wireless sensor networks. **Signal Processing Magazine, IEEE**, [S.l.], v.22, n.4, p.54–69, 2005.

PENG, C.; SHEN, G.; ZHANG, Y. BeepBeep: a high-accuracy acoustic-based system for ranging and localization using cots devices. **ACM Trans. Embed. Comput. Syst.**, New York, NY, USA, v.11, n.1, p.4:1–4:29, Apr. 2012.

PERLA, E. et al. PowerTOSSIM Z: realistic energy modelling for wireless sensor network environments. In: ND ACM WORKSHOP ON PERFORMANCE MONITORING AND MEASUREMENT OF HETEROGENEOUS WIRELESS AND WIRED NETWORKS, 3., New York, NY, USA. **Proceedings...** ACM, 2008. p.35–42. (PM2HW2N '08).

PRESS, W. H. et al. **Numerical Recipes 3rd Edition**: the art of scientific computing. 3.ed. New York, NY, USA: Cambridge University Press., 2007.

RAMACHANDRAN, I.; DAS, A. K.; ROY, S. Analysis of the Contention Access Period of IEEE 802.15.4 MAC. **ACM Trans. Sen. Netw.**, New York, NY, USA, v.3, n.1, Mar. 2007.

RAPPAPORT, T. **Wireless Communications**: principles and practice. 2nd.ed. Upper Saddle River, NJ, USA: Prentice Hall PTR., 2001.

REICHENBACH, F.; BLUMENTHAL, J.; TIMMERMANN, D. Improved Precision of Coarse Grained Localization in Wireless Sensor Networks. In: DIGITAL SYSTEM DESIGN: ARCHITECTURES, METHODS AND TOOLS, 2006. DSD 2006. 9TH EUROMICRO CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. p.630–640.

SAVARESE, C.; RABAEY, J. M.; LANGENDOEN, K. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In: GENERAL TRACK OF THE ANNUAL CONFERENCE ON USENIX ANNUAL TECHNICAL CONFERENCE, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2002. p.317–327. (ATEC '02).

SAVVIDES, A.; HAN, C.-C.; STRIVASTAVA, M. B. Dynamic fine-grained localization in Ad-Hoc networks of sensors. In: MOBILE COMPUTING AND NETWORKING, 7., New York, NY, USA. **Proceedings...** ACM, 2001. p.166–179. (MobiCom '01).

SIMON, G. et al. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In: AEROSPACE CONFERENCE, 2003. PROCEEDINGS. 2003 IEEE. **Anais...** [S.l.: s.n.], 2003. v.3.

SRINIVASAN, K. et al. An Empirical Study of Low-power Wireless. **ACM Trans. Sen. Netw.**, New York, NY, USA, v.6, n.2, p.16:1–16:49, Mar. 2010.

STEHLIK, M. **Comparison of Simulators for Wireless Sensor Networks**. Dissertação (Mestrado) — Masaryk University Faculty of Informatics. 2011.

STOJMENOVIC, I. Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions. **Communications Magazine, IEEE**, [S.l.], v.46, n.12, p.102–107, 2008.

Texas Instruments. **CC2420**. Acessado em Março/2012, http://www.ti.com/lit/ds/symlink/cc2420.pdf.

Texas Instruments. **CC2500**. Acessado em Abril/2013, http://www.ti.com/lit/ds/swrs040c/swrs040c.pdf.

THOMAS, G. B. **Cálculo**. Volume 2.ed. São Paulo, PA, USA: Addison Wesley/Pearson Education do Brasil., 2002.

TINYOS. **TinyOS**. Acessado em Junho/2012, http://www.tinyos.net.

VOLDER, J. E. The CORDIC Trigonometric Computing Technique. **Electronic Computers, IRE Transactions on**, [S.l.], v.EC-8, n.3, p.330–334, 1959.

YEDAVALLI, K. **Location Determination Using IEEE 802.llb**. Dissertação (Mestrado) — The University of Colorado at Boulder. 2002.

YEDAVALLI, K.; KRISHNAMACHARI, B. Sequence-Based Localization in Wireless Sensor Networks. **Mobile Computing, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–1, 2009.

ZHANG, Z. et al. COSMO: co-simulation with matlab and omnet++ for indoor wireless networks. In: GLOBAL TELECOMMUNICATIONS CONFERENCE (GLOBECOM 2010), 2010 IEEE. **Anais...** [S.l.: s.n.], 2010. p.1–6.

ZHAO, J. et al. Localization of Wireless Sensor Networks in the Wild: pursuit of ranging quality. **Networking, IEEE/ACM Transactions on**, [S.l.], v.21, n.1, p.311–323, 2013.

ZHENYU, S. **A Simulation Tool for Wireless Sensor Networks**. Dissertação (Mestrado) — Politecnico di Torino, Torino, Italy. 2009.

ZHOU, G. et al. Models and Solutions for Radio Irregularity in Wireless Sensor Networks. **ACM Trans. Sen. Netw.**, New York, NY, USA, v.2, n.2, p.221–262, May 2006.

ZUNIGA, M.; KRISHNAMACHARI, B. Analyzing the transitional region in low power wireless links. In: SENSOR AND AD HOC COMMUNICATIONS AND NETWORKS, 2004.

IEEE SECON 2004. 2004 FIRST ANNUAL IEEE COMMUNICATIONS SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. p.517–526.

APÊNDICES

APÊNDICE A – Decomposição QR

Conforme (BJORCK, 1996) e (DEMMEL, 1997), a decomposição QR é comumente utilizada na resolução de sistemas de equações, representadas por

$$Au = b, \tag{A.1}$$

pelo método de mínimos quadrados linearizado. Esta decomposição consiste em transformar a matriz A, da equação característica (A.1), em uma ortogonal denominada Q e uma triangular superior R, de modo que $A = QR$. A matriz triangular superior é caracterizada por possuir todos os elementos abaixo da diagonal principal iguais a zero, conforme

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ 0 & a_{22} & a_{23} & \dots & a_{2m} \\ 0 & 0 & a_{33} & \dots & a_{3m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_{nm} \end{bmatrix}. \tag{A.2}$$

Valendo-se das propriedade de que a multiplicação de uma matriz ortogonal pela sua transposta resulta na identidade ($QQ^T = I$), transforma-se (A.1) em

$$Ru = bQ^T \tag{A.3}$$

conforme os passos

$$Au = b \rightarrow QRu = b \rightarrow QQ^T Ru = bQ^T \rightarrow Ru = bQ^T. \tag{A.4}$$

Pelo fato de a matriz R ser uma triangular superior, a última linha do sistema só possui uma variável indeterminada que pode ser resolvida diretamente. Então, substituindo-se esta na linha imediatamente superior e assim sucessivamente, consegue-se resolver todas as variáveis do sistema de equações. Assim, a principal vantagem da decomposição QR é evitar a necessidade da inversão de matrizes presente no método tradicional, caracterizado por

$$u = \left(A^T A\right)^{-1} A^T b, \tag{A.5}$$

para resolver o sistema linear de equações (A.1).

Ainda segundo (BJORCK, 1996) e (DEMMEL, 1997), dois dos métodos padrões para transformar a matriz A nas Q e R são o por reflexões de Householder e o por rotações de Givens, sendo ambos iterativos. A principal diferença entre os dois é que no primeiro zeram-se

as posições necessárias de uma das colunas da matriz R a cada iteração, enquanto que o segundo zera-se posição por posição, necessitando assim mais iterações para concluir o processo. Não é foco deste trabalho aprofunda-se nestes dois métodos, sendo estes são detalhados em (PRESS et al., 2007), (BJORCK, 1996) e (DEMMEL, 1997).

APÊNDICE B – Algoritmo de multilateração por aproximação polinomial

Segue abaixo o código, em linguagem Matlab, criado para a execução do método de multilateração por aproximação polinomial apresentado no Capítulo 3.2. Este algoritmo leva em consideração um cenário quadrado com lateral de $100m$ e quatro nodos âncora nos extremos. Caso estas dimensões sejam modificadas, os coeficientes dos dois polinômios devem ser gerados novamente, sendo que quanto maior a ordem destes, menor será o erro de localização.

```
% [x y] -> Posição estimada de determinado nodo desconhecido após uma ...
    iteração do método MVI
% [x_centr y_centr] -> Centro geométrico dos nodos âncora
x_aux = x - x_centr;
y_aux = y - y_Centr;
%%% Definindo em qual das 4 laterais se encontra o nodo desconhecido
if abs(x_aux) >= abs(y_aux)
    a = x_aux;
    b = y_aux;
    a_sinal = (a>=0) * 2-1;
    sinal = [a_sinal 0];
else
    a = y_aux;
    b = x_aux;
    a_sinal = (a>=0) * 2-1;
    sinal = [0 a_sinal];
end
%%% Correção através do primeiro polinômio
u = abs(a);
if u > 15
    n = 2;                    % ordem do  primeiro polinômio
    p = [0.0085 -0.3  2.93]; % coeficientes do primeiro polinômio ...
        gerado pelo Matlab
    v = u.^2.*p(n-1) + p(n).*u + p(n+1);
    %%% Correção através do segundo polinômio
    a2 = a + a_sinal*v;
    u2 = abs(b/a2);
    if u2 < 0.85
        n = 2;
        p = [-1.91   0.41    0.96];
        v2 = v * ( u2.^2.*p(n-1) + p(n).*u2 + p(n+1) );
%%% Condições em que não se aplicam correções
    else
        v2 = 0;
    end
else
    v= 0;
    v2 = 0;
end
%%% Aplicando a correção a posição estimada
[x_est y_est] = [x_centr y_centr]  + [x y] + sinal.*[v2 v2];
```