



**UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE
AUTOMAÇÃO E SISTEMAS**

**APLICAÇÃO DE TÉCNICAS DE FUSÃO SENSORIAL PARA
MAPEAMENTO E LOCALIZAÇÃO SIMULTÂNEOS PARA
ROBÔS TERRESTRES**

DANIEL COSTA RAMOS
Engenheiro Eletricista

**Florianópolis/SC
2012**

DANIEL COSTA RAMOS

**APLICAÇÃO DE TÉCNICAS DE FUSÃO SENSORIAL PARA
MAPEAMENTO E LOCALIZAÇÃO SIMULTÂNEOS PARA
ROBÔS TERRESTRES**

Trabalho apresentado ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina - UFSC, como requisito parcial para a obtenção de título de Mestre em Engenharia de Automação e Sistemas.

Área de concentração: Controle, Automação e Sistemas.

Orientador: Prof. Dr. Ubirajara Franco Moreno

**Florianópolis/SC
2012**

APLICAÇÃO DE TÉCNICAS DE FUSÃO SENSORIAL PARA MAPEAMENTO E LOCALIZAÇÃO SIMULTÂNEOS PARA ROBÔS TERRESTRES

DANIEL COSTA RAMOS

‘Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em *Controle, Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina.’

Ubirajara Franco Moreno, Dr., UFSC
Orientador

Jomi Fred Hübner
Coordenador do Programa de Pós-Graduação em
Engenharia de Automação e Sistemas

Banca Examinadora:

Ubirajara Franco Moreno, Dr., UFSC
Presidente

Edson Roberto De Pieri, Dr., UFSC

Eugênio de Bona Castelan Neto, Dr., UFSC

Anderson Luiz Fernandes Perez, Dr., UFSC

AGRADECIMENTOS

Agradeço a Deus pelas oportunidades que me foram dadas nesta grande escola que é a vida, principalmente por ter conhecido pessoas e lugares interessantes, mas também por ter vivido fases difíceis, que foram matérias-primas de aprendizado e me tornaram na pessoa que sou hoje.

Não posso deixar de agradecer aos meus pais Cláudio e Lúcia Ramos, sem os quais não estaria aqui, e por terem me fornecido condições para me tornar o profissional e homem que sou.

Ao meu irmão, aos meus amigos e à minha namorada, que sempre me apoiaram e sempre estiveram ao meu lado, sendo definitivamente uma força impulsionadora para a conclusão deste trabalho.

E ao professor Dr. Ubirajara Moreno, pela paciência e orientação deste trabalho, inclusive nos momentos mais difíceis.

Daniel Costa Ramos

Este trabalho é dedicado a todos os meus familiares e pessoas intimamente ligadas à minha vida, que no período de desenvolvimento deste trabalho me ajudaram com paciência, carinho e compreensão, demonstrando que a superação nos momentos difíceis vale a pena, por estarmos ao lado de quem realmente se importa com nosso sucesso.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia de Automação e Sistemas.

APLICAÇÃO DE TÉCNICAS DE FUSÃO SENSORIAL PARA MAPEAMENTO E LOCALIZAÇÃO SIMULTÂNEOS PARA ROBÔS TERRESTRES

Daniel Costa Ramos

Março/2012

Orientador: Ubirajara Franco Moreno, Dr.

Área de Concentração: Controle, Automação e Sistemas.

Palavras-chave: robótica, SLAM, filtro de partículas, fusão de sensores, filtro de Kalman.

Número de Páginas: xxvi + 98

Um dos problemas que envolvem as soluções para a mobilidade de robôs móveis terrestres é estimar a posição do robô com precisão juntamente com a exploração do ambiente, mapeando-o corretamente (SLAM - Simultaneous Localization and Mapping – Localização e Mapeamento Simultâneo). Embora vários algoritmos tenham sido desenvolvidos nos últimos anos, exigindo uma carga de cálculo computacional cada vez maior dos robôs, estes estão susceptíveis a um mau desempenho quando os sensores apresentam ruídos, quando há problemas nos atuadores, variáveis não modeladas ou em virtude de algum imprevisto momentâneo no ambiente. A proposta deste trabalho é programar um SLAM para robôs móveis interligando-o a uma combinação de sensores inerciais com sensores de odometria através de uma técnica de fusão de sensores conhecida como filtro de Kalman Estendido, para reduzir a incerteza na estimação da posição e melhorar o desempenho do SLAM. Por consequência, o custo computacional é reduzido. O trabalho foi estruturado iniciando por uma revisão a respeito dos conceitos básicos de sensoriamento, a fim de contextualizar o problema e apresentar as nomenclaturas e termos utilizados. A seguir foram abordadas as técnicas de fusão de dados, as representações do robô e do ambiente, as técnicas de mapeamento e exploração e as diversas técnicas de navegação que podem ser utilizadas, para ambientes conhecidos e para ambientes desconhecidos. Essas informações são importantes para um melhor entendimento do problema, de como representá-lo e de como se pode avaliar os resultados obtidos. Na sequência é apresentado o SLAM, destacando as principais técnicas e em detalhes o Grid Based FastSLAM. É demonstrado através de simulações que quanto maior as incertezas sobre a posição do robô, um número maior de partículas é necessário para manter a qualidade do mapa gerado, e como cada partícula possui um mapa associado a si, o custo computacional é consideravelmente aumentado. Outro aspecto analisado foi o impacto na escolha da covariância associada à transição de estados, propondo a utilização da covariância inerente ao cálculo da fusão de sensores como parâmetro de refinamento no SLAM.

Abstract of Thesis presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Automation and Systems Engineering.

SIMULTANEOUS LOCALIZATION AND MAPPING FOR LAND ROBOTS WITH SENSORIAL FUSION TECHNIQUES

Daniel Costa Ramos

Março/2012

Advisor: Ubirajara Franco Moreno, Dr

Area of Concentration: Control, Automation and Systems.

Keywords: robotic, SLAM, particle filter, sensor fusion, Kalman filter.

Number of Pages: xxvi + 98

One of the problems in solutions involving land mobile robots is the estimation of the robot position with precision and at the same time, explore the environment and mapping it correctly (SLAM - Simultaneous Localization and Mapping). Several algorithms were developed in the last years, demanding large computational resources in robots and even so, these may have a bad performance in cases of sensors having noises, problems in actuators, not modeled variables or when there is something in the environment that wasn't expected. This dissertation proposal is to program a SLAM algorithm for mobile robots and connect it with a sensor data fusion, between odometry and inertial sensors, using the Extended Kalman Filter, achieving a reduction of the position uncertainty and improving the SLAM performance, also reducing the need of computational resources. This work begins with a revision of concepts of robot sensors, needed to understand later algorithms and nomenclatures. In the following items it is described the sensor fusion techniques, the robot localization problem, the map and robot representation alternatives, and the navigation problems for explored and non-explored environments. These information are important for a better understanding of the problem, on how represent it and how to evaluate the obtained results. After this introduction, it's described some SLAM algorithms, featuring in details the Grid Based FastSLAM. It's demonstrated by simulations that as high uncertainty about robot position, as large are the number of particles needed to maintain the generated map quality. This implies in a large computational cost, thus improving the uncertainty with sensor data fusion makes the robot work with less particles. It is also showed that choosing the right covariance in robot transition model is very important and finding a way to connect the covariance of sensor data fusion with SLAM can improve performance even more.

SUMÁRIO

LISTA DE ILUSTRAÇÕES.....	XV
LISTA DE QUADROS	XVII
LISTA DE EQUAÇÕES.....	XIX
LISTA DE SIGLAS.....	XXI
1 INTRODUÇÃO.....	1
1.1 JUSTIFICATIVA.....	3
1.2 OBJETIVOS.....	4
1.2.1 Objetivo Geral.....	4
1.2.2 Objetivos Específicos.....	4
1.3 METODOLOGIA.....	5
1.4 ORGANIZAÇÃO DO TRABALHO	5
2 SENSORES	7
2.1 CARACTERÍSTICAS DE DESEMPENHO.....	9
2.2 TIPOS DE SENSORES.....	11
2.2.1 Encoder Óptico (Odometria).....	12
2.2.2 Bússola	14
2.2.3 Giroscópio e Inclinômetro.....	14
2.2.4 GPS	15
2.2.5 Acelerômetro.....	15
2.2.6 Ultrassom.....	16
2.2.7 Laser	17
2.2.8 Sensores Inerciais Integrados	17
3 FUSÃO DE DADOS	21
3.1 MODELOS DE FUSÃO DE SENSORES	22
3.2 INTRODUÇÃO AO FILTRO DE KALMAN	24
3.3 FILTRO DE KALMAN ESTENDIDO	26
3.3.1 Aplicação à Robótica Móvel.....	28
3.4 APLICAÇÕES E CONCLUSÃO.....	30
4 LOCALIZAÇÃO E NAVEGAÇÃO	31
4.1 REPRESENTAÇÃO DE CRENÇAS.....	32

4.1.1 Crença de Hipótese Única.....	33
4.1.2 Crença de Múltiplas Hipóteses.....	34
4.2 REPRESENTAÇÃO DO MAPA.....	35
4.2.1 Representação Contínua.....	35
4.2.2 Estratégias de Decomposição Comuns.....	35
4.2.3 Mapa de Grade.....	37
4.3 MÉTODOS DE NAVEGAÇÃO.....	38
4.3.1 Navegação em Regiões Exploradas.....	39
4.3.2 Navegação em Regiões Não Exploradas.....	41
4.4 APLICAÇÃO E CONCLUSÕES.....	41
5 SLAM.....	43
5.1 FILTRO DE PARTÍCULAS.....	45
5.2 GRID FASTSLAM.....	49
5.2.1 Transição das Partículas.....	50
5.2.2 Mapeamento.....	51
5.2.3 Pesos e Reamostragem.....	52
5.3 INTEGRAÇÃO DA FUSÃO DE DADOS E SLAM.....	54
5.4 CONCLUSÕES.....	57
6 EXPERIMENTOS E RESULTADOS.....	59
6.1 SIMULAÇÃO.....	60
6.2 CASOS SIMULADOS.....	62
6.3 AVALIAÇÃO.....	63
6.4 RESULTADOS.....	65
6.4.1 Caso 1.....	66
6.4.2 Caso 2.....	68
6.4.3 Caso 3.....	69
6.4.4 Caso 4.....	71
7 CONCLUSÃO.....	73
REFERÊNCIAS.....	75
APÊNDICE A – PROBABILIDADE.....	81
APÊNDICE B – DISTRIBUIÇÃO GAUSSIANA.....	83
APÊNDICE C – REDES BAYESIANAS.....	85
APÊNDICE D – SIMULAÇÃO DE TRAJETÓRIAS.....	93

LISTA DE ILUSTRAÇÕES

Figura 1 – Robô Pioneer com seus sensores.	7
Figura 2 - Exemplo de Encoder.....	13
Figura 3 - Exemplo de encoder absoluto.....	13
Figura 4 - Exemplo de uma IMU simples	18
Figura 5 - IMU VN-100 da VectorNav.....	18
Figura 6 - Arquitetura centralizada de IMU/Odometria.....	23
Figura 7 - Arquitetura descentralizada de IMU/Odometria.	23
Figura 8 - Resumo do filtro de Kalman.....	26
Figura 9 - Pose de um robô.	28
Figura 10 - Navegação baseada em comportamentos.	32
Figura 11 - Navegação baseada em localização.	32
Figura 12 - Representação de crenças a respeito da posição do robô...	33
Figura 13 - Exemplo de localização de múltiplas hipóteses.....	34
Figura 14 – Simplificação do mapa contínuo.....	35
Figura 15 - Exemplo de decomposição exata de célula.	36
Figura 16 - Exemplo de mapa topológico.	36
Figura 17 - Exemplo de decomposição fixa.....	37
Figura 18 - Decomposição com células adaptativas.	37
Figura 19 - Exemplo de mapa por ocupação de grade.	38
Figura 20 - Conectividade de 4 e 8 pontos.....	40
Figura 21 – Exemplo do algoritmo Brushfire.	40
Figura 22 - Algoritmo Wave Front.	41
Figura 23 - Exemplo de trajeto no GraphSlam.....	44
Figura 24 - Primeira Etapa do FP para Localização.....	46
Figura 25 - Segundo Etapa do FP para Localização.	47
Figura 26 – Terceira Etapa do FP para Localização.	48
Figura 27 – Quarta Etapa do FP para Localização.	48
Figura 28 - Exemplo de roda de probabilidades com 9 partículas.	54
Figura 29 - Diagrama do algoritmo proposto.....	55

Figura 30 - Simulação das trajetórias no ambiente “real”.....	60
Figura 31 - Efeito do sensor nos obstáculos.	61
Figura 32 - Simulação do Grid Based Fast SLAM.	61
Figura 33 - Barra de tonalidade em relação aos valores.	61
Figura 34 - Mapa com região de avaliação em azul.....	64
Figura 35 - Ambiente a ser explorado pelo robô.	65
Figura 36 - Exemplo de mapa gerado pela odometria somente.	65
Figura 37 - Exemplo de mapa gerado pela fusão fixa.....	66
Figura 38 - Exemplo de mapa gerado pela fusão adaptativa.....	66
Figura 39 - Número de acertos na odometria.....	67
Figura 40 - Número de erros na odometria.	67
Figura 41 - Número de acertos na fusão de dados fixa.	68
Figura 42 – Número de erros na fusão de dados fixa.....	69
Figura 43 - Número de acertos na fusão de dados adaptativa.	70
Figura 44 – Número de erros na fusão de dados adaptativa.	70
Figura 45 - Número de acertos entre os métodos.....	72
Figura 46 - Número de erros na fusão de dados adaptativa.	72
Figura 47 - Distribuição Gaussiana.....	83
Figura 48 - Exemplo de Rede Bayesiana.	85
Figura 49 - Medição de uma variável não observável.	88
Figura 50 - Modelo de transições de uma HMM.	89
Figura 51 – Rede de Bayes aplicada à robótica.	92
Figura 52 - Resultados obtidos com o robô real.	93
Figura 53 - Simulação do deslocamento no tempo.	94
Figura 54 - Erro entre o estado estimado e o estado real no tempo.	94
Figura 55 - Segunda trajetória simulada.	94
Figura 56 - Terceira trajetória simulada.....	95
Figura 57 - Simulação de trajetórias.	95

LISTA DE QUADROS

Quadro 1 - Classificação dos Sensores	8
Quadro 2 – Propriedades da IMU VN-100.	19
Quadro 3 - Algoritmo do FP FastSLAM.	46
Quadro 4 - Algoritmo do GBFS.....	50
Quadro 5 - Algoritmo de Reamostragem.	54
Quadro 6 – Algoritmo da Fusão de Dados.....	56
Quadro 7 – Exemplo de cálculo na rede de Bayes.....	87
Quadro 8 – Exemplo de cálculo utilizando o normalizador.....	88
Quadro 9 – Exemplo de HMM.	90
Quadro 10 – Segundo exemplo de HMM.	91
Quadro 11 – Terceiro exemplo de HMM.....	91

LISTA DE EQUAÇÕES

Equação (1) - Exatidão	10
Equação (2) - Precisão	10
Equação (3) - Equação de Estados	25
Equação (4) - Equação de Observação	25
Equação (5) - Estimador a Priori – KF	25
Equação (6) - Matriz de Covariância a Priori – KF	25
Equação (7) - Ganho do Filtro de Kalman – KF	25
Equação (8) - Estimador a Posteriori – KF	25
Equação (9) - Matriz de Covariância a Posteriori – KF	25
Equação (10) - Modelo de Estados Não Linear	26
Equação (11) - Modelo de Observação Não Linear	26
Equação (12) - Estimador a Priori Não Linear	27
Equação (13) - Modelo de Observação Não Linear	27
Equação (14) - Linearização do Espaço de Estados	27
Equação (15) - Linearização do Observador	27
Equação (16) - Matriz de Covariância a Priori - EKF	27
Equação (17) - Ganho do Filtro de Kalman – EKF	27
Equação (18) - Estimador a Posteriori – EKF	28
Equação (19) - Matriz de Covariância a Posteriori – EKF	28
Equação (20) - Vetor de Estados	28
Equação (21) - Modelo Cinemático de um Robô	29
Equação (22) - Atualização da Matriz de Estados	29
Equação (23) - Vetor de Observação	29
Equação (24) - Estimador da Observação	29
Equação (25) - Estimador a Posteriori Completo - EKF	30
Equação (26) - Formulação do Full SLAM	44
Equação (27) - Formulação do SLAM Online	44
Equação (28) - Transição Partículas – Covariância Fixa	51
Equação (29) - Transição Partículas – Covariância Variável	51

Equação (30) - Regra da Probabilidade Total	52
Equação (31) - Regra de Bayes	52
Equação (32) - Peso das partículas	53
Equação (33) - Passo da Reamostragem	53
Equação (34) - Função de Amostragem Gaussiana	83
Equação (35) - Regra de Bayes	86
Equação (36) - Regra da Probabilidade Total	86
Equação (37) - Regra de Bayes Simplificada	86
Equação (38) - Regra de Bayes Simplificada Negada	86
Equação (39) - Normalizador para Regra de Bayes	86
Equação (40) - Regra de Bayes com Normalizador	86

LISTA DE SIGLAS

EC	Exteroceptivos.
EKF	<i>Extended Kalman Filter</i> – Filtro de Kalman Estendido.
FP	Filtro de Partículas.
GBFS	<i>Grid Based FastSLAM</i> – FastSLAM baseado em grade de ocupação.
GPS	<i>Global Positioning System</i> – Sistema de Posicionamento Global
HMM	Modelo oculto de Markov (<i>Hidden Markov Model</i>)
IMU	<i>Inercial Measurement Unit</i> - Unidade de Medição Inercial.
KF	<i>Kalman Filter</i> – Filtro de Kalman.
MEMS	Sistemas Mecânicos Microeletrônicos
MM	Modelo de Markov (<i>Markov Model</i>).
PC	Proprioceptivos.
PF	<i>Particle Filter</i> – Filtro de Partículas.
RB	Rede Bayesiana.
SDF	<i>Sensor Data Fusion</i> - Fusão de Sensores / Fusão de Dados.
SLAM	<i>Simultaneous Localization and Mapping</i> – Localização e Mapeamento Simultâneo.

***A mente que se abre a uma nova ideia,
Jamais voltará ao seu tamanho original.***

(Albert Einstein)

1 INTRODUÇÃO

Os robôs móveis são robôs capazes de se locomover no ambiente em que estão inseridos a fim de completar uma determinada tarefa. São compostos fundamentalmente de três sistemas: o sistema de locomoção, responsável em fazer o robô se mover pelo ambiente; o sistema de controle, responsável por decidir que ações o robô deverá executar; e por fim, o sistema sensorial, composto por sensores cuja finalidade é analisar os estados internos do robô como orientação e velocidade, além de analisar o ambiente em torno do robô.

Os robôs móveis podem ser classificados de acordo com diferentes taxonomias. Em relação a sua anatomia, são classificados em aéreo, aquático ou terrestre (rodas, esteiras, pernas); quanto ao tipo de controle: tele operados, semiautônomos e autônomos; quanto à funcionalidade: industrial, de serviço (ambiente conhecido), de campo (ambiente desconhecido e imprevisível) e pessoais. Ainda pode-se classificá-los quanto ao seu movimento, em holonômico e não-holonômico, sendo que o primeiro não apresenta restrições em relação ao movimento do robô, ao contrário do segundo.

Um robô móvel é dito autônomo quando o mesmo possui habilidade de tomar decisões, a partir das informações do ambiente. Esta habilidade pode ser completa ou parcial, tendo na segunda a interferência humana nas decisões de alto nível ou decisões críticas.

Segundo Bianchi (2010) há três perguntas básicas na robótica móvel autônoma. A primeira é “Onde estou?” fazendo referência a capacidade de localização do robô, para em seguida questionar “Onde eu estou indo?”, referindo-se ao seu objetivo e posição final. E por último, “Como eu chego lá?” que está relacionado em como planejar a sua trajetória, ou seja, um trajeto válido pelo ambiente, da sua posição atual até o seu objetivo.

Assim o robô para conseguir responder a estas questões necessita ter ou construir um modelo de si, ter a capacidade de perceber e modelar o ambiente e suas variações, saber a sua localização e saber planejar/executar o seu movimento no ambiente, desviando de potenciais obstáculos, sejam estes estáticos ou dinâmicos. O nome do procedimento que realiza estes processos simultaneamente é conhecido como SLAM (Simultaneous Localization and Mapping – Localização e Mapeamento Simultâneo).

A localização e mapeamento simultâneo (SLAM) é um problema fundamental na robótica e as primeiras soluções surgiram da combinação de algoritmos de mapeamento de ambientes com posições conhecidas e da localização de robôs em um mapa pré-definido.

O filtro de Kalman foi um conhecido método na teoria de controle por décadas, enquanto sua formulação probabilística aplicada na robótica surgiu na década de 90 com os trabalhos de Dean (1990) e de Simmons (1995), sendo que alguns trabalhos posteriores introduziram o termo “Localização de Markov”.

Os primeiros modelos de SLAM baseavam-se em variações do Filtro de Kalman como o filtro de Kalman Estendido (EKF – Extended Kalman Filter) (Thrun, 2006). Estes algoritmos necessitam de marcadores ou de detecção de características do ambiente (como a identificação de paredes, rodovias, linhas, lâmpadas, portas, etc).

A partir destas primeiras propostas surgiram SLAM com filtros informativos, SLAM com filtros de partículas, com grafos (GraphSlam), combinações como o SLAM por filtro de partículas Rao-Blackwellizado, dentre outros.

No SLAM a precisão dos atuadores e dos sensores utilizados é de extrema importância, pois o erro é repassado à localização e ao mapeamento. Para isso, algumas técnicas de SLAM tentam compensar os erros, porém com um custo computacional elevado e diretamente associado ao tamanho do erro (Dissanayake, 2002).

Em algumas situações não será possível adquirir um sensor de última geração e de extrema precisão, devido ao seu elevado custo. Como alternativas, existem técnicas que possibilitam melhorar e utilizar sensores mais econômicos e de menor precisão.

Uma destas técnicas é o processo de combinação dos dados provindos de múltiplos sensores de mesma natureza ou de naturezas diferentes, que é denominado fusão de sensores (Luo, 1995). Seu principal objetivo é fornecer ao sistema dados de maior qualidade, unindo as informações provindas dos sensores.

As técnicas de fusão de sensores podem ser utilizadas em uma grande variedade de aplicações como a determinação da trajetória de robôs, reconhecimento de alvos, sensoriamento remoto, monitoramento, entre outros (Solustiano, 2007).

A proposta deste trabalho é mostrar que algoritmos de SLAM para robótica móvel terrestre, podem ter um melhor desempenho se combinados com uma técnica de fusão de sensores, utilizando as informações provindas da odometria e de uma unidade de medição inercial (IMU). Além disso, existem possibilidades de melhoria

potencializada se o SLAM utilizar algumas informações que são calculadas durante a fusão de sensores, como parâmetro de refinamento a cada interação.

1.1 JUSTIFICATIVA

Na sociedade há uma crescente necessidade de se realizar tarefas com eficiência e precisão, inclusive em lugares onde a presença humana se torna difícil, arriscada e até mesmo impossível. Para realizar essas tarefas, se faz cada vez mais necessária a presença de dispositivos automatizados. Existem atualmente robôs prestando serviços em diversas áreas da sociedade, como os robôs desarmadores de bomba, robôs bombeiros, robôs de inspeção de tubulações, robôs de inspeção de grandes profundidades, entre outras atividades.

Os robôs podem assumir diferentes formas e construí-los exige resolver vários problemas de engenharia, ciência da computação, ciência cognitiva, linguagem e muitas outras. Independente da forma do robô ou da tarefa que o mesmo deve realizar, os robôs devem manobrar pelo ambiente em que se localizam.

Para que esta função seja realizada com perfeição, os robôs dependem de um hardware de elevado custo, com sensores precisos e com uma alta capacidade de processamento para lidar com algoritmos complexos.

Além disso, os algoritmos desta área da robótica se tornaram extremamente abrangentes, os livros especializados sobre o assunto reúnem informações de diversos autores e artigos, devido à complexidade do tema. Assim, roboticistas sempre encontram uma grande dificuldade de colocar em prática os algoritmos estudados.

O foco inicial do trabalho foi contextualizar uma sequência com todas as etapas de planejamento de uma navegação completa, revisando e escolhendo desde a representação do robô, a sua navegação local e o SLAM propriamente dito, voltado principalmente para robôs de menor custo, onde os sensores não apresentam informações totalmente livres de ruídos.

Uma oportunidade de aplicar um método conhecido como fusão de sensores foi abordada em seguida, o qual tem sido aplicado recentemente em robôs autônomos de última geração, como o carro autônomo da Google (Thrun, 2006). Este método visa melhorar as

informações obtidas dos sensores a fim de se diminuir o custo computacional e melhorar os resultados no SLAM.

O SLAM possui diversas técnicas que podem ser utilizadas. Segundo Choset et al. (2005), desde a década de 90 até a publicação de seu livro, houve grandes avanços no campo de planejamento de trajetórias e navegação, particularmente nos métodos probabilísticos.

Thrun (2006) confirma em seu livro que a robótica probabilística se tornou uma das técnicas dominantes ao se projetar o movimento e controle de novos robôs, inclusive sendo o algoritmo utilizado no carro automatizado vencedor da famosa corrida do DARPA, onde carros devem completar um longo trajeto no deserto de forma totalmente automatizada.

Portanto, como forma de aprendizado e de se avaliar a fusão de sensores, foram escolhidos modelos avançados que utilizam conceitos de probabilística e filtro de partículas, para realizar o SLAM e aplicada a fusão de sensores para melhorar o seu desempenho. Na prática, isto implica em menor processamento para o robô e uma melhor confiabilidade do mapeamento, exploração e navegação do robô de maneira geral.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um algoritmo de SLAM probabilístico baseado em partículas integrado a um algoritmo de fusão de sensores, para demonstrar que a fusão de sensores melhora o desempenho dos algoritmos de SLAM, reduzindo o custo computacional dos mesmos.

1.2.2 Objetivos Específicos

- Projetar, construir e simular o algoritmo de fusão de dados entre a odometria e sensores inerciais, para analisar se a trajetória do robô predita através da fusão é melhor do que a predita somente através da odometria;
- Projetar, construir e simular um algoritmo de SLAM, realizando a localização e mapeamento de um ambiente simulado controlado, para poder avaliar os diversos resultados igualmente;
- Utilizar ambos os algoritmos e comparar os resultados para

algumas variações: utilizando somente a odometria, utilizando a fusão e por último, utilizando os dados reais sem erros. Com estas simulações é visado avaliar através dos mapas gerados, o impacto da variação do número de partículas, do valor da covariância no modelo de transição do SLAM e os métodos entre si;

- Por fim, pretende-se integrar diretamente a covariância do Filtro de Kalman Estendido (durante a etapa da fusão) no SLAM para analisar se há melhora de desempenho.

1.3 METODOLOGIA

Inicialmente foi construído o algoritmo de fusão de dados para os sensores utilizando uma estrutura centralizada e o algoritmo do Filtro de Kalman Estendido. Os sensores utilizados foram a odometria e uma unidade de medição inercial (IMU).

Em seguida, foi projetado um algoritmo de SLAM compatível com o a fusão de dados, sendo escolhida uma técnica de SLAM probabilístico denominada Grid Based Fast SLAM.

Os algoritmos de fusão de sensores e do SLAM foram projetados utilizando o software Matlab.

Foram criados alguns experimentos utilizando somente a odometria, utilizando a fusão entre odometria e IMU e utilizando esta mesma fusão, mas com a inserção de uma quantidade maior de dados da fusão no SLAM.

Para análise de desempenho dos algoritmos foram variados alguns parâmetros e avaliado o impacto nos mapas gerados. Os mapas gerados por todos os casos sob as mesmas condições também foram avaliados.

1.4 ORGANIZAÇÃO DO TRABALHO

Na etapa inicial, os conceitos básicos de sensores são revisados para melhor entendimento da justificativa do uso da fusão de sensores, sendo apresentada uma revisão sobre os tipos de sensores, suas propriedades e origem dos erros.

No terceiro capítulo, é descrito a fusão de dados, explicando de que maneira e onde é utilizada a técnica e detalhando sobre suas

diferentes estruturas. O capítulo é finalizado com a técnica de fusão de sensores aplicada neste trabalho.

No quarto capítulo são expostas as diferentes técnicas de representação do robô e do ambiente, detalhando em como é realizada a representação escolhida no SLAM deste trabalho. Ainda é apresentado como o robô pode navegar pelo ambiente, tanto em regiões do mapa já exploradas como nas não exploradas, dando ênfase nas considerações que serão utilizadas para avaliar os mapas gerados pelo SLAM.

O quinto capítulo trata a respeito do SLAM propriamente dito, onde as principais técnicas são revisadas, para descrever o funcionamento de um filtro de partículas e o SLAM escolhido (Grid FastSLAM).

Por fim, no último capítulo, são apresentadas a simulação e os resultados obtidos, junto com as conclusões sobre os métodos propostos.

2 SENSORES

Sensores são dispositivos que transformam uma grandeza física como temperatura, pressão, umidade, distância, velocidade e aceleração, em um sinal elétrico, que por sua vez, é interpretado por equipamentos eletrônicos (Borges e Dores, 2010). Na robótica, os sensores são responsáveis tanto pela percepção do ambiente em torno do robô, quanto pelo fornecimento de informações sobre o mesmo.

Este item do trabalho pretende revisar os tipos básicos de sensores, suas propriedades e suas principais fontes de erros, para saber o porquê de se utilizar fusão de sensores e contextualizar sobre os sensores que serão utilizados.

Segundo Sigewart e Nourbakhsh (2004), os sensores podem ser classificados como sensores proprioceptivos (PC) ou exteroceptivos (EC), ou seja, que possui medições de valores internos do robô (como velocidade do motor) ou medições externas ao mesmo (do ambiente), como distância, intensidade da luz, etc.



Figura 1 – Robô Pioneer com seus sensores.

Os sensores Passivos (P) ou Ativos (A) são oriundos de outra classificação atribuída a estes, onde os sensores passivos são os que medem através da captação da energia do ambiente, como por exemplo, sensores de temperatura, microfone, etc. Enquanto os ativos emitem algum tipo de energia no ambiente e medem a reação no mesmo. Geralmente os sensores ativos possuem um desempenho melhor, mas estão sujeitos a interferências externas e/ou a alteração das características do que se está medindo.

Os principais sensores e suas características estão listados no Quadro 1.

Classificação Usual	Sensor ou Sistema de Sensores	PC/EC	A/P
Sensores Tácteis - Detecção de contato físico ou de aproximação; chave de segurança.	Chaves de contato, bumpers.	EC	P
	Barreiras Ópticas	EC	A
	Sensor Proximidade sem contato	EC	A
Sensor das rodas/motor - Velocidade/posição do motor ou roda	Potenciômetro	PC	P
	Resolvedor Síncrono	PC	P
	Encoder com Brush (Dentes)	PC	A
	Encoder Óptico	PC	A
	Encoder Magnético	PC	A
	Encoder Indutivo	PC	A
Sensor de Direção - Mede orientação do robô em relação a um referencial fixo.	Encoder Capacitivo	PC	A
	Bussola	EC	P
	Giroscópio	PC	P
Sinalizadores Terrestres - Fornece localização em um referencial fixo.	Inclinômetro	EC	A/P
	GPS	EC	A
	Ópticos Ativos ou por RF	EC	A
	Sinalizadores Ativos de Ultrassom	EC	A
Sensor de Distância Ativo - Reflexivos, por “tempo de voo” ou por triangulação geométrica.	Sinalizadores Reflexivos	EC	A
	Sensores Reflexivos	EC	A
	Sensores de Ultrassom	EC	A
	Sensores de Laser	EC	A
	Triangulação Óptica (1D)	EC	A
Sensores de Movimento/Velocidade - Mede velocidade em relação ao referencial fixo.	Iluminação Estruturada (2D)	EC	A
	Radar - Doppler	EC	A
Sensores Baseados em Visão - Distância visual, análise de imagem, segmentação, reconhecimento de objetos.	Som - Doppler	EC	A
	Câmera CCD/CMOS	EC	P
	- Pacotes de Distância Visual - Pacotes de Seguimento de Objetos		

Quadro 1 - Classificação dos Sensores

Fonte: Adaptado de Sigewart e Nourbakhsh (2004) e Borenstein (1997).

Na Figura 1 pode-se observar um exemplo de robô comercial (o Pioneer) com sensores típicos encontrados na robótica como sensores inerciais (IMU), sonares, sensor de distância, sensores de contato e encoders nas rodas.

2.1 CARACTERÍSTICAS DE DESEMPENHO

É importante saber as características dos sensores, para melhor avaliar qual é o mais adequado para a aplicação desejada e sua real capacidade.

Uma característica muito importante é o seu alcance dinâmico, que é utilizado para avaliar os limites superior e inferior dos valores de entrada durante a operação normal do sensor.

Na robótica móvel essa característica é vital para saber como os sensores do robô operam em ambientes onde o robô será inserido. Um exemplo é o caso de sensores ultrassônicos que possuem um valor mínimo de distância e que podem fornecer dados falsos ao se aproximar de objeto mais que o limite.

Outra característica é a resolução do sensor, que consiste na diferença mínima entre dois valores que são medidos pelo sensor. Se o mesmo for lido através de uma porta analógica / digital (A/D), a resolução do sensor será quantos bits esta porta está utilizando.

A largura de banda ou frequência é uma indicação da velocidade da medição por segundo. Se a frequência for muito baixa, o desempenho total do robô pode ser limitado, pois sua dinâmica geralmente é baseada no limite máximo de velocidade de sua largura de banda.

Estas características podem ser obtidas em laboratório e estimadas para uma aplicação no mundo real. Mas algumas características não podem ser estimadas e necessitam uma análise mais complexa do sensor e do ambiente.

A sensibilidade do sensor é uma destas características e indica quanto uma mudança no sinal de entrada altera o sinal de saída. Infelizmente os sensores do tipo EC, se possuírem uma sensibilidade razoável, podem sofrer interferência externa ao robô ou de outros sensores do mesmo, prejudicando o desempenho.

A sensibilidade cruzada é um termo técnico utilizado quando há sensibilidade a fatores não desejáveis do ambiente. Um exemplo é uma bússola magnética sendo utilizada para indicar a direção norte, ao passar perto de um ímã, passará a ser influenciado pelo mesmo, indicando uma leitura errada.

O erro no sensor é formalmente definido como a diferença entre a saída medida pelo sensor (m_{sensor}) e o valor verdadeiro (v_{real}). Ou seja, $\text{erro} = m_{\text{sensor}} - v_{\text{real}}$. Os erros podem ser sistemáticos ou aleatórios,

podendo o primeiro ser modelado matematicamente de forma precisa (é determinístico) e o segundo pode ser modelado matematicamente de forma estocástica.

A exatidão é definida como o grau de conformidade entre a medida do sensor e seu valor real, sendo expresso em porcentagem do seu valor verdadeiro. A exatidão é obtida pela expressão (1). O erro pequeno corresponde a uma alta exatidão.

$$\text{exatidão} = 1 - \frac{|\text{erro}|}{V_{\text{real}}} \quad (1)$$

A precisão é geralmente confundida com exatidão, mas está relacionada à capacidade do sensor de reproduzir os mesmos resultados, embora não sejam os verdadeiros.

Se considerarmos um erro com média de valor μ e desvio padrão σ , temos a definição formal de precisão como indicado em (2), que depende de σ e da capacidade de alcance do sensor. Ou seja, não depende de μ .

$$\text{precisão} = \frac{\text{alcance}}{\sigma} \quad (2)$$

Na robótica móvel ocorre uma particularidade, onde os erros aleatórios podem se tornar sistemáticos. Podemos ter como exemplo um sonar ultrassônico, onde o mesmo emite uma onda de ultrassom e mede o tempo para a mesma voltar ao sensor após refletir em algum objeto, sendo que pode ocorrer a medição de ondas secundárias por haverem diferentes tipos e formas de superfícies. Estas ondas podem alterar a leitura do sensor. No caso do robô se movendo, este erro é bem comum e estocástico. Se o mesmo ocorrer enquanto o robô estiver parado, esta leitura errada se torna sistemática.

Ainda para simplificação da análise de erros, os mesmos geralmente são considerados de média zero e tendo probabilidade de ocorrência com distribuição gaussiana.

Os erros, determinísticos e estocásticos, podem ser classificados em diversos tipos, de acordo com a sua origem [Nassar 2003], os principais são:

- **Erro de Sensibilidade:** quando na prática, o valor é diferente do valor especificado, mas o sensor ainda é linear;
- **Erro de BIAS:** ocorre quando a saída do sensor não é zero quando a propriedade medida é zero;

- **Erro Não Linear:** quando o sinal de saída não é constante na faixa de atuação do sensor;
- **Erro Dinâmico:** ocorre quando a propriedade muda mais rápido do que o sensor consegue acompanhar;
- **Erro por Drift:** ocorre quando o sinal do sensor sofre uma deterioração ao longo do tempo, sem que haja modificação da propriedade medida;
- **Erro por Ruído:** erro aleatório no sinal que varia com o tempo, geralmente causado por interferências ou pelo meio em que se encontra;
- **Erro por Histerese:** ocorre quando a propriedade medida muda a direção da variação e o sensor possui um atraso para perceber a mudança, criando um offset em uma direção;
- **Erro por Digitalização:** ocorre quando a saída do sensor é digital e há uma aproximação da propriedade medida.

2.2 TIPOS DE SENSORES

Existem diversos tipos de sensores. Porém, a construção adequada da fusão de sensores depende do entendimento a respeito do seu funcionamento básico, incluindo os principais pontos fortes e fracos de cada tipo de sensor. As principais categorias e a descrição dos detalhes dos principais sensores estão listadas abaixo.

- **Sensores das Rodas e Motores:** Estes sensores são dispositivos que medem o estado interno do robô. Possuem ampla aplicação além da robótica móvel, sendo de alta qualidade, de baixo custo e excelente resolução;
- **Sensores de Direção:** Sensores de direção podem ser do tipo EC (giroscópio e inclinômetro) ou PC (bússola), e são utilizados para saber a orientação e inclinação do robô. Estes possibilitam com a informação de velocidade, integrar o movimento e estimar a posição. Este procedimento é conhecido como *dead reckoning*;
- **Sinalizadores Terrestres:** Os sinalizadores terrestres utilizam a interação dos sensores internos com sinalizadores em terra, possibilitando saber a localização precisamente. A tecnologia atual consegue fazer com que se tenha uma precisão de 5 cm

em áreas com quilômetros de metros quadrados. Este tipo de sensor pode utilizar laser, rádio frequência, entre outros sinais para realizar esta interação. O mais utilizado é o GPS;

- **Sensores de Distância Ativos:** São os sensores mais populares na robótica móvel, medem a distância baseando-se no tempo do sinal emitido ou por triangulação. São utilizados principalmente para detecção e desvio de obstáculos. Fazem parte de sistemas de localização e modelagem de ambientes. Estes sensores só devem perder espaço quando a tecnologia de sensoriamento por visão estiver mais desenvolvida;
- **Sensores de Velocidade:** Alguns sensores medem diretamente o movimento entre o robô e o ambiente, podem utilizar diversas propriedades desde a variação de calor para detectar humanos, a sensores de efeito Doppler. Estes sensores são mais utilizados em veículos autônomos em estradas;
- **Sensores Baseados em Visão:** Sensores baseados em visão estão cada vez mais populares devido à evolução da computação e de rápido processamento. Estes sensores fornecem informações ricas do ambiente, possibilitando uma interação maior. Há áreas na computação totalmente dedicadas para a visão computacional e reconhecimento de padrões, mas esta tecnologia ainda é recente e está em fase de maturação. Mas pode ser utilizada na robótica móvel, para reconhecimento de trajetórias, atualização de posição, detecção de pessoas e obstáculos, entre outras aplicações;
- **Sensores Inerciais Integrados:** Este tipo de sensor mede as unidades inerciais, fornecendo velocidade, aceleração e deslocamento, de forma simples e rápida.

2.2.1 Encoder Óptico (Odometria)

O encoder óptico é o dispositivo mais popular para medir velocidade angular e posição a partir do motor, sendo classificado como um Sensor de Rodas e Motores. É instalado geralmente no eixo do motor e como se trata de um sensor proprioceptivo, a estimativa da posição é melhor se utilizado o referencial local do robô. No caso de ser utilizado para localização, os dados devem passar por correções e filtragens.

Os sensores ópticos mais comuns são o incremental e o de quadratura de fase. O primeiro não detecta sentido e consiste em um

feixe de luz que é interrompido pelo disco. A segunda interrupção do feixe de luz gera pulsos que são contados. E assim, sabendo o tempo entre um pulso e outro, ou a quantidade de pulsos em um determinado tempo, é possível determinar a velocidade.

O segundo tipo, por quadratura de fase, consiste de dois feixes de luz, sendo o segundo posicionado a 90 graus do outro. Assim, pode ser comparado o formato de onda entre os dois para saber em qual sentido o encoder está girando.

Outro sensor é o encoder absoluto, que possui um disco com perfuração complexa, é menos comum e é utilizado principalmente para determinar posição. Assim, é mais adequado para baixas velocidades e uso não contínuo.

Quanto às fontes de erros, segundo Ribeiro (1999), os erros sistemáticos deste tipo de medição ocorrem por haver diâmetro desigual ou desalinhamento das rodas, diâmetro das rodas com valor diferente do nominal e incerteza sobre o ponto de contato da roda. Por outro lado, os erros não sistemáticos acontecem por movimentos em solos não uniformes, sobre obstáculos inesperados no solo e escorregamento das rodas devido a solo escorregadio, grandes acelerações do veículo ou rotações rápidas.

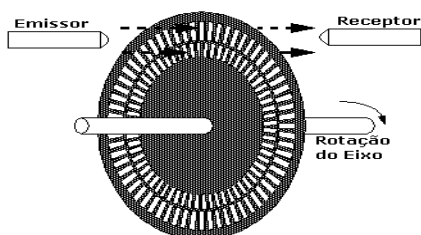


Figura 2 - Exemplo de Encoder

Fonte: Adaptado de Borenstein (1997).



Figura 3 - Exemplo de encoder absoluto

Fonte: Borenstein (1997).

2.2.2 Bússola

É um sensor de direção e os dois tipos de bússolas mais comuns utilizadas na robótica móvel são a de efeito Hall e a de fluxo. O primeiro analisa o comportamento em um semicondutor quando na presença de um campo magnético, o qual possibilita identificar a intensidade e direção do campo magnético analisando a diferença de potencial no sensor.

Mas este sensor, apesar de baixo custo, não possui uma boa resolução e tem erros devido a não linearidade dos semicondutores. Existem filtros e maneiras de lidar com este problema, mas resultam em uma largura de banda baixa, ou seja, o sensor demora em fornecer os dados.

Independente do tipo, o principal problema em usar o campo magnético da Terra para navegação são os distúrbios causados pelos campos magnéticos de outros objetos, além da limitação de largura de banda e sua suscetibilidade a vibrações. Procura-se evitar o uso de bússolas para robôs que são utilizados em ambientes internos.

2.2.3 Giroscópio e Inclinômetro

Os giroscópios são sensores de direção que fornecem orientação em relação a um referencial fixo, fornecendo uma medida absoluta da direção de um sistema móvel. Existem giroscópios mecânicos, piezelétricos e ópticos.

O do tipo mecânico é baseado na conservação de momento, medindo a mudança no momento linear ou angular. Estes possuem o problema de poderem ser afetados por vibrações.

Já no piezoelétrico há um prisma interno e três transdutores piezoelétricos que convertem força em eletricidade. Quando inclinado, haverá uma diferença de oscilação entre dois transdutores, resultando em uma tensão de saída que indica a direção.

Os giroscópios ópticos são relativamente recentes, tendo uso comercial nos anos 80. Estes utilizam dois feixes de luz ou lasers, que são afetados em sua frequência pela velocidade angular do cilindro contendo as fibras ópticas. Os giroscópios mais atuais possuem frequência e largura de banda além das necessidades de aplicações na robótica móvel, tendo valores acima de 100 kHz de largura de banda e resoluções menores que 0,001 graus/h.

Os erros mais comuns neste tipo de sensor são por *drift*.

2.2.4 GPS

O sistema de posicionamento global (GPS) foi inicialmente desenvolvido para uso militar, sendo disponível posteriormente para fins civis. O sistema é constituído de 24 satélites (e mais quatro de reserva) a uma altura de 20.190 km, é considerado um sensor do tipo sinalizador terrestre.

Os satélites mandam um sinal sincronizado a todo instante. Assim, utilizando três satélites e o tempo necessário para que os sinais cheguem ao receptor é possível determinar sua posição. Mas para ser mais preciso e realizar uma correção no tempo, é utilizado um satélite a mais, necessitando, portanto, de quatro satélites.

O fato de precisar ler a informação de quatro satélites simultaneamente é uma limitação significativa, já que o sinal emitido pelo satélite é fraco e qualquer obstáculo pode dificultar a leitura do sinal. Portanto, lugares que possuem edifícios altos, florestas densas e lugares fechados têm dificuldade de sincronização.

A resolução do GPS pode variar conforme a posição na Terra, pois os satélites não estão localizados com simetria perfeita em torno do planeta, possuindo um desempenho melhor de leitura de latitude e longitude nas regiões próximas aos polos e uma qualidade inferior da leitura da altitude em relação às leituras em regiões equatoriais.

Outra consideração no GPS é que o mesmo possui largura de banda por volta de 5 Hz devido a sua latência de 200 a 300 ms. Para robôs rápidos ou aéreos, é necessário realizar uma integração do movimento para se ter um controle adequado.

2.2.5 Acelerômetro

O acelerômetro é um tipo de sensor conforme o próprio nome indica, mede a aceleração do corpo no qual está anexado em comparação com a gravidade. Existem dezenas de tipos de acelerômetros, enquanto o conceito e objetivo do acelerômetro são os mesmos, a detecção da aceleração é feita de maneiras diferentes como por indução magnética, por piezoeletricidade ou ainda por capacitores ou tecnologia de sistemas mecânicos microeletrônicos (MEMS).

Este sensor está se tornando um dispositivo popular principalmente com uso nos smartphones, notebooks, GPS, airbags de carros, câmeras digitais e outros dispositivos móveis.

Graças à tecnologia MEMS, atualmente os acelerômetros com sensibilidade tri-axial estão reduzidos a apenas um componente com cerca de 15 mm³ e pesando apenas alguns miligramas.

O acelerômetro possui algumas limitações segundo Figueireira (2007), principalmente quando se trata de uma medição de deslocamentos e navegação inercial. Para calcular o deslocamento é necessário separar a componente gravitacional da aceleração. E para cálculo do deslocamento, isto não pode ser realizado com eficiência utilizando apenas um acelerômetro.

Segundo Figueireira (2007) seriam necessários pelo menos seis acelerômetros para esta aplicação. Com apenas um acelerômetro, seria possível efetuar medições de deslocamento durante as quais a orientação do referencial do sensor não varia (ou varia de forma conhecida). No entanto, embora haja uma dupla integração para obtenção da posição, um pequeno erro na leitura da aceleração pode provocar erros muito grandes em alguns segundos. Estes ainda podem ser mais acentuados se houver variações de temperatura no sensor.

2.2.6 Ultrassom

O ultrassom é um sensor ativo de distância e o funcionamento básico deste sensor consiste em mandar pacotes de ondas ultrassônicas e medir o tempo que esse pacote de ondas demora a refletir e voltar ao receptor. A distância d do objeto que causa a reflexão pode ser calculada através da velocidade de propagação do som c e do tempo de voo t .

O sensor emite o pacote de ondas e possui um limite onde indica que o pacote está terminando ou começado. O receptor ao reconhecer o padrão e os limites, informa que recebeu o sinal e verifica quanto tempo passou desde a sua emissão, calculando assim com a velocidade do som, a distância percorrida.

O sinal destes sensores possui frequência em torno de 40 a 180 kHz e seus feixes de som se propagam como um cone, com ângulos de abertura de 20 a 40 graus. Possuem a limitação de poderem ser afetados pelo material do objeto e o ângulo de incidência da onda, sendo o melhor caso quando a onda atinge um objeto perpendicularmente, o que não ocorre o tempo todo.

A sua largura de banda fica em torno de 50 Hz no caso de apenas um sensor e um objeto na sua frente, porém no caso de vários sensores e cálculos exatos para evitar interferência, essa taxa pode cair para 2,5 Hz.

2.2.7 Laser

O sensor ativo de distância a laser possui um desempenho melhor que o do ultrassom, pois utiliza a luz ao invés do som. Este sensor pode possuir três maneiras de medir o tempo de propagação do sinal, sendo um método que envia um laser pulsado e mede o tempo de voo diretamente, como no sensor ultrassônico, mas são mais caros por causa dos receptores próprios para este fim.

O segundo método é medir a frequência resultante entre uma onda contínua modulada por frequência (FMCW) e a reflexão recebida. Ainda se pode utilizar uma técnica mais fácil, a de mudança de fase da luz refletida.

A resolução destes sensores pode chegar a valores mínimos como 0,5 graus ou menos, com resolução de profundidade de 5 cm ao longo de alcance de 5 cm a 20m. Ainda existem modelos que fazem 25 varreduras em 180° por segundo.

No caso destes sensores as reflexões indevidas dos sinais ocorrem apenas em superfícies muito polidas como monitor, gabinetes, espelho. Outra limitação em relação aos sensores de ultrassom, é que estes sensores não detectam materiais transparentes, não sendo adequado para ambientes com muitos espelhos.

2.2.8 Sensores Inerciais Integrados

Um sensor de medição inercial integrado possui em seu interior vários sensores diferentes, consistindo geralmente de três giroscópios, magnetômetro (instrumento usado para medir a intensidade, direção e sentido de campos magnéticos em sua proximidade) e três acelerômetros. Alguns modelos também apresentam GPS integrado ou outros tipos de sensores como de temperatura e de pressão atmosférica.

Os sensores inerciais integrados podem receber várias nomenclaturas como IMU, *Inertial Measurement Unit*, *Inertial Motion Unit*, Sensores Inerciais, Unidade de Medição Inercial, entre outros. Para simplificar, é utilizado apenas o termo IMU quando se refere a este sensor neste trabalho.

Este sensor pode atuar de várias formas, conforme o modelo e fabricante. De forma geral operam de duas formas: fornecendo diretamente as unidades inerciais de cada sensor interno ou passando as informações já filtradas e trabalhadas (fornecendo diretamente a velocidade, aceleração e descolamento).

A principal dificuldade encontrada nesse tipo de sensor é a sincronização com sensores externos e a calibração complexa dos sensores de menor custo.



Figura 4 - Exemplo de uma IMU simples

Fonte: Micro Strain (2011)

Apesar de ser uma tecnologia com custo cada vez menor, os preços dos sensores ainda possuem uma grande variação de acordo com a sua precisão.

Uma lista criada pelo pesquisador francês Damien Douxchamps (Douxchamps 2012) e mantida atualizada pelo grupo, mostra os principais sensores atuais, sua capacidade e seu preço (quando disponível).

É possível observar variações do valor entre U\$100,00 à U\$2600,00 em uma IMU. Geralmente as de valor elevado possuem aplicações militares ou aeroespaciais, que exigem extrema precisão.

Para a robótica móvel, principalmente a terrestre, os sensores de baixa precisão possuem uma faixa de operação satisfatória operando com variações de até 150° /s nos giroscópios e no caso dos acelerômetros, dentro da faixa de 5 G.



Figura 5 - IMU VN-100 da VectorNav.

Fonte: Vectornav (2012)

Foi selecionada uma IMU (Vectornav 2012) de baixo custo para demonstrar algumas características comuns nos modelos mais atuais (Quadro 2).

Propriedades	Valores
Alcance Roll, Heading	$\pm 180^\circ$
Alcance Pitch	$\pm 90^\circ$
Precisão Estática (Heading)	$2,0^\circ$
Precisão Estática (P&R)	0,5
Resolução Angular	$0,05^\circ$
Repetitividade	$0,2^\circ$
Largura de Banda	200 Hz
Capacidade Angular	$\pm 500^\circ/\text{s}$
Densidade de Ruído	$0,005^\circ/\text{s}/\sqrt{\text{Hz}}$
Erro de Alinhamento	$\pm 0,05^\circ$
Capacidade Aceleração	$\pm 8 \text{ g}$
Densidade de Ruído	$400 \text{ mg}/\sqrt{\text{Hz}}$
Erro de Alinhamento	$\pm 0,05$
Capacidade Magnética	$\pm 2,5 \text{ Gauss}$
Densidade de Ruído	$140 \mu\text{Gauss}/\sqrt{\text{Hz}}$
Erro de Alinhamento	$\pm 0,05$

Quadro 2 – Propriedades da IMU VN-100.

Fonte: Vectornav (2012)

Este sensor suporta combinações dos seguintes filtros e saídas:

- Ângulos de Euler (Yaw - Guinada, Pitch - Arfagem, Roll - Rolagem);
- Quatérnions;
- Matriz de Rotação;
- Filtros de aceleração, taxa angular, campo magnético e pressão;
- Filtro de Kalman Estendido (EKF);
- Calibração (automática e manual) para prevenção de distúrbios magnéticos ou dinâmicos;
- Processamento de sinal adaptativo;

Neste trabalho foi considerado que no robô há um sensor similar a este, fornecendo informações já filtradas sobre a velocidade angular e linear do robô, não necessitando fazer transformações de coordenadas e nem integrações.

As fontes de erros nos sensores inerciais são constituídas em duas partes: uma parte constante (ou determinística) e uma parte estocástica. A parte determinística inclui fatores de BIAS e de escalonamento, que podem ser determinados por calibração e removidos das leituras.

A parte estocástica consiste de pequenas variações aleatórias dos erros durante o tempo que podem ser modeladas estaticamente e incluídas no modelo de erro da IMU para ser utilizado em um filtro de Kalman.

Estes erros podem ser representados por ruídos brancos, constante aleatória, caminho aleatório, Gauss-Markov e por período aleatório.

Segundo Nasser [2003], para boa parte das aplicações de sistemas de IMU (com *drift* de 0,005-0,01 graus/h) um modelo de primeira ordem de Gauss-Markov (GM) é utilizado para representar o erro. Isto também é verdade para IMU de baixo custo (com *drift* de 100-1000 graus/h), no entanto, um ruído branco também pode ser utilizado para estes casos.

3 FUSÃO DE DADOS

A literatura sobre os processos de Fusão de Dados é relativamente recente e há muita imprecisão e divergência quanto aos termos relacionados à fusão de dados de sensores.

Os termos Fusão de Dados (*Data Fusion*), Fusão de Sensores (*Sensor Fusion*), Integração de Múltiplos Sensores (*Multi-sensor Integration*), Fusão de (Dados de) Múltiplos Sensores (*Multi-sensor (Data) Fusion*) e Fusão da Informação, são utilizados para se referir à variedade de técnicas, sistemas e aplicações que visam à combinação de informações providas de múltiplas fontes de dados (Salustiano 2006).

A ideia precursora da Fusão de Sensores é que através desta técnica, se obtenha um sistema que forneça informações úteis sobre algum aspecto do ambiente ou no caso, do robô.

As vantagens desse tipo de integração podem ser resumidas como sendo:

- Melhor tolerância a falhas devido ao uso de múltiplos sensores, pois caso algum falhe, os outros fornecem a informação correta;
- Maior confiabilidade, pois a leitura de um sensor pode ser confirmada por outros sensores que efetuam a leitura do mesmo ambiente;
- Combinação de informações, pois pode haver sensores que meçam diferentes aspectos do ambiente, combinando para fornecer uma nova informação;
- Redução de ambiguidade e na incerteza;
- Robustez contra interferência, pois pode haver diversos tipos de sensores diferentes que não são afetados pelo mesmo tipo de ruído;
- O custo de processamento, recursos computacionais ou materiais podem ser menores ao se utilizar esse tipo de técnica.

Matematicamente é utilizada alguma técnica que considera incerteza na informação (como redes neurais ou filtro de Kalman) para produzir um conjunto de dados combinados (como se houvesse um 'sensor virtual').

Na robótica, os robôs tem que ser capazes de observar o ambiente, tomar uma decisão baseada nestas observações e atuar de diferentes maneiras para atingir um objetivo. Assim, para poder navegar de uma maneira mais eficiente pelo ambiente o robô pode possuir

diversos sensores, de diversos tipos, medindo o mesmo estado ou estados diferentes.

Mas o próprio modelo matemático do robô pode possuir alguma imprecisão associada ou a leitura de cada sensor possuir algum tipo de perturbação, ruído ou falha. Isto causa um mau desempenho do robô.

Para diminuir os erros, nos sistemas lineares pode ser utilizada a fusão de dados através do filtro de Kalman (KF), e para o caso não linear pode ser utilizado filtros como o Filtro de Kalman Estendido (*Extended Kalman Filter - EKF*) ou o *Unscented Kalman Filter* (UKF).

A formulação da fusão de sensores deste trabalho foi adaptada principalmente de Menegaldo (2008) e Welch (2006).

Assim, neste capítulo é feita uma introdução sobre a arquitetura da fusão de dados, sendo em seguida detalhado o funcionamento sobre o algoritmo de combinação (o filtro de Kalman), sendo em seguida detalhado o filtro de Kalman Estendido, o qual foi utilizado para realizar a fusão neste trabalho.

3.1 MODELOS DE FUSÃO DE SENSORES

Os modelos de integração dos sensores podem ser agrupados pela extensão da ajuda que um componente tem na função do outro. O primeiro tipo de integração é o acoplamento de sistemas, ou seja, cada sensor gera as suas próprias estimativas, que dependem do modelo do sistema, para ao final serem combinados. O segundo tipo de integração é a combinação dos dados “brutos” dos sensores, independente do sistema, realizando uma fusão baseada, por exemplo, na correção da velocidade medida por um sensor dado a leitura de outro sensor de velocidade de maior precisão, mas com menos disponibilidade.

Há vários tipos de arquiteturas para realizar a fusão, em Abdel-Hamid (2005) são citadas algumas para os casos de fusão entre GPS e IMU, para obtenção das matrizes de posição, velocidade e aceleração (r , v , a) de um veículo terrestre autônomo. Apesar de que para o GPS algumas considerações adicionais devem ser tomadas como a largura de banda, as arquiteturas definidas podem ser adaptadas para o caso de uma fusão com odometria e IMU.

Assim, a estruturas dos filtros podem ser caracterizadas de duas maneiras: centralizado e descentralizado (Figura 6 e Figura 7 respectivamente). No primeiro, os dados de cada um dos sensores são

recebidos “brutos” (sem nenhum processamento) na central, para depois obter a solução, que no caso, é a estimação da posição do robô.

Em um sistema descentralizado é usada uma abordagem mais sequencial, onde o resultado de sistemas individuais serve como entrada de sistemas subsequentes. Como exemplo, a saída da odometria passar por um filtro de Kalman para diminuir o erro e logo em seguida entrar no filtro de Kalman da IMU no formato de correção. Para alguns casos como sistema de detecção de falhas, de isolamento e de correção, a abordagem descentralizada é mais simples e geralmente é escolhida.

Mas segundo Abdel-Hamid (2005), o processo centralizado oferece melhor desempenho em soluções de navegação que implementam um modelo de filtro de Kalman robusto. Os dois modelos estão exemplificados pela Figura 6 e Figura 7, sendo p_{est} , v_{est} e a_{est} , os vetores de posição, velocidade e aceleração estimados pelos sensores para os três eixos cartesianos.

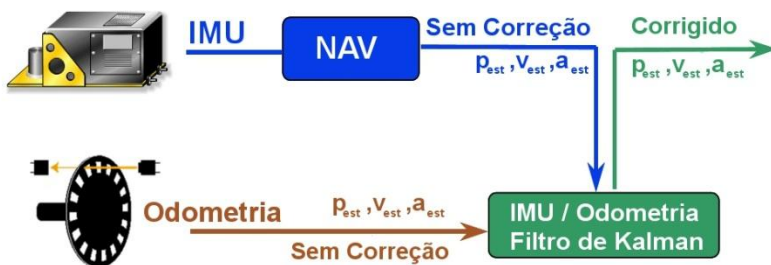


Figura 6 - Arquitetura centralizada de IMU/Odometria.

Fonte: Adaptado de Abdel-Hamid (2005).

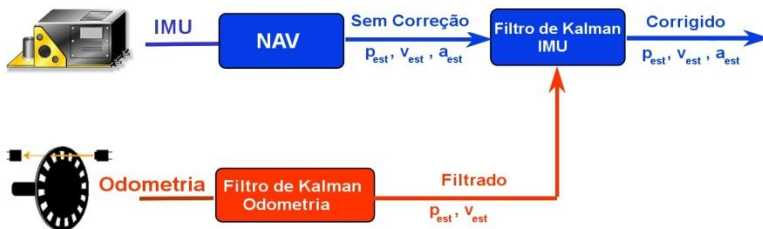


Figura 7 - Arquitetura descentralizada de IMU/Odometria.

Fonte: Adaptado de Abdel-Hamid (2005).

Há também algumas variações destes modelos, principalmente realizando laços entres os filtros ou diretamente com a IMU, ajudando a refinar ainda mais os filtros. Apesar desta abordagem não ser utilizada neste trabalho, a ideia de se utilizar a informação contida no filtro para melhorar o desempenho do robô foi construída de maneira similar, mas visando o melhor desempenho do algoritmo de SLAM.

A estrutura utilizada neste trabalho é semelhante ao da Figura 6, com a diferença nos tipos de dados fornecidos por cada sensor, que no caso deste trabalho, não foi utilizado o vetor de aceleração a_{est} da IMU e desconsiderado os valores referentes à terceira dimensão (eixo z).

Considerou-se também que a IMU não necessita de um processo de interpretação de seus sensores internos (a mecanização “NAV” nas figuras, que transforma, por exemplo, força em aceleração) e que sua saída já fornece os dados na forma de p_{est} , v_{est} e a_{est} . Como mencionado anteriormente, os modelos atuais de IMU já trazem essa função de fábrica, não necessitando construir essa etapa de processamento.

Visto como se organiza a estrutura da fusão de dados, os próximos itens abordam como são os algoritmos que realizam a combinação dos sensores.

3.2 INTRODUÇÃO AO FILTRO DE KALMAN

O filtro de Kalman (Kalman Filter - KF) surgiu na década de sessenta, na área da engenharia elétrica relacionada à teoria do controle de sistemas. Posteriormente, o KF foi inserido em diversas áreas, como inteligência artificial e robótica.

Resumidamente, o KF é um conjunto de equações matemáticas que constitui um algoritmo recursivo e eficiente para estimação, uma vez que o erro quadrático é minimizado.

Há dois tipos principais de variáveis utilizadas para a estimação, as variáveis diretamente mensuráveis e as variáveis que representam os estados do robô e que não são diretamente mensuráveis.

No KF podem ser estimados os estados passados, o estado presente e mesmo previstos os estados futuros, dependendo da aplicação. Além disso, é um procedimento aplicável quando os modelos do sistema estão escritos sob a forma de espaço-estado.

As equações podem ser agrupadas em dois tipos distintos: equações de predição e equações de atualização da medição ou de

correção. Estes dois grupos de equações funcionam em conjunto, uma alimentando a outra com novas informações.

As equações de predição são responsáveis pelo avanço das variáveis de estado e das covariâncias no tempo para se obter as estimativas a priori. As equações de atualização trazem informações das medições e as incorporam nas estimativas a priori para obter um ganho, que é utilizado para fazer a estimação posterior.

Considera-se um sistema na forma de (3) e (4), onde X_k é o vetor de estados do sistema, F é o modelo de transição de estados, B é o modelo das entradas de controle, u é o vetor de controle, m é o ruído do processo, assumido como sendo amostrado de uma distribuição normal multivariada de média zero e covariância Q , Y é o vetor de observação, H é o modelo de observação, n é o ruído da observação, assumido como sendo um ruído branco gaussiano de média zero e covariância R .

O primeiro passo é atualizar as equações para o novo instante, estimando um novo X . A equação do estimador a priori é dada em (5), sendo \tilde{X}_{k-1}^+ o estimador corrigido de X_{k-1} (do instante anterior), baseado nas informações obtidas ao longo das interações.

$$X_k = FX_{k-1} + Bu_{k-1} + m_{k-1} \quad (3)$$

$$Y_k = HX_k + n_k \quad (4)$$

$$\tilde{X}_k^- = F\tilde{X}_{k-1}^+ + Bu_{k-1} \quad (5)$$

Em seguida é calculada a matriz de covariância a priori dos erros das variáveis de estados, dada por (6).

$$P_k^- = FP_{k-1}^+ F' + Q \quad (6)$$

Estas duas últimas equações são responsáveis pela atualização no tempo, ou seja, pela predição do instante $k-1$ para o instante k . Quando é recebida uma observação Y_k , o estimador pode ser corrigido utilizando as equações (7), (8) e (9), que representam o ganho de Kalman, o estimador a posteriori (corrigido) e a matriz de covariância a posteriori.

$$K_k = P_k^- H' (HP_k^- H' + R_k)^{-1} \quad (7)$$

$$\tilde{X}_k^+ = \tilde{X}_k^- + K_k (Y_k - H\tilde{X}_k^-) \quad (8)$$

$$P_k^+ = (I - K_k H) P_k^- \quad (9)$$

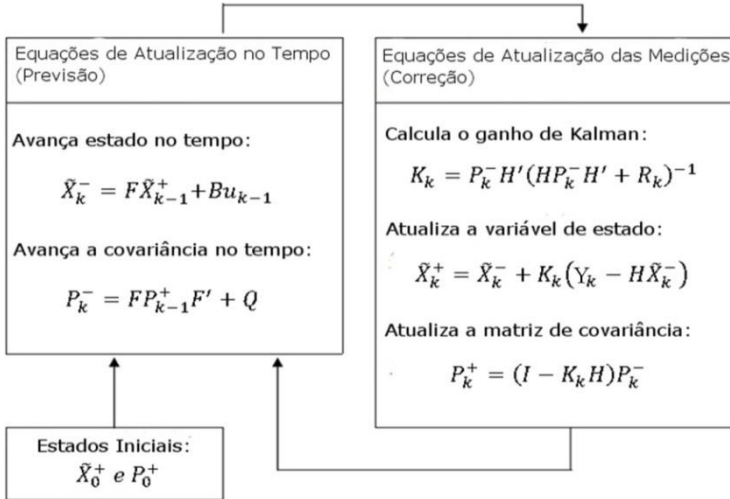


Figura 8 - Resumo do filtro de Kalman.

O resumo sobre o algoritmo do filtro de Kalman está apresentado na Figura 8. Esse procedimento é utilizado para sistemas lineares, para casos não lineares é necessário realizar uma linearização do sistema, quando a aproximação é feita por expansão da série de Taylor recebe a denominação de Filtro de Kalman Estendido (*Extended Kalman Filter - EKF*). Há outras formas de linearização como, por exemplo, o Unscented Kalman Filter (UKF), no qual realiza uma aproximação estocástica através do uso de processos estocásticos de regressão linear. Mas neste trabalho será apenas descrito em detalhes o EKF.

3.3 FILTRO DE KALMAN ESTENDIDO

Este é o algoritmo de fusão de dados utilizado neste trabalho. Primeiramente o EKF necessita que o sistema esteja na forma discreta, em forma de espaço de estados (como o KF no item anterior). Assim o espaço de estados X e modelo de observação Y , podem ser representados como (10) e (11).

$$X_k = f(X_{k-1}, u_{k-1}, m_{k-1}) \quad (10)$$

$$Y_k = h(X_k, n_k) \quad (11)$$

Onde f e h são funções não lineares, u é o vetor de controle, m e n são ruídos do processo e das medidas respectivamente. Estes são aleatoriamente gaussianos, de média zero e não correlacionados. Como m e n , não são conhecidos para toda amostra, o modelo do sistema pode ser aproximado para (12) e (13).

$$\tilde{X}_k^- = f(\tilde{X}_{k-1}^+, u_{k-1}, 0) \quad (12)$$

$$\tilde{Y}_k = h(\tilde{X}_k^-, 0) \quad (13)$$

Onde \tilde{X}_{k-1}^+ é o estado estimado a posteriori da amostra $k-1$ (anterior) e \tilde{X}_k^- é o estado estimado a priori da amostra atual. Estas equações podem ser linearizadas a cada amostra em torno de (12) e (13), obtendo as matrizes jacobianas A e C linearizadas a cada interação. Assim, temos as matrizes Jacobianas A e M como sendo as derivadas parciais de f em relação à X e m , e as Jacobianas C e N as derivadas parciais de h em relação à X e n respectivamente, deixando o sistema na forma de (14) e (15).

$$X_k \approx \tilde{X}_k^- + A_{k-1} \cdot [X_{k-1} - \tilde{X}_{k-1}^+] + M_{k-1} \cdot m_{k-1} \quad (14)$$

$$Y_k \approx \tilde{Y}_k + C_k \cdot [X_k - \tilde{X}_k^-] + N_k \cdot n_k \quad (15)$$

A partir destas equações, pode se chegar às equações do algoritmo. Assim, o algoritmo do filtro EKF, como descrito em Welch (2006) e Menegaldo (2008), segue a sequência:

- Escolha de um valor inicial para \tilde{X}_0^+ e P_0^+ (0).
- Calcule os estados estimados, equação (12).
- Medir os valores de Y_k dos sensores.
- Estimar a matriz de covariância de erro Q_{k-1} e R_k (processo e medição).
- Calcular no instante anterior as matrizes jacobianas A_{k-1} , M_{k-1} e do instante atual, C_k e N_k .
- Cálculo dos resíduos da covariância a priori, equação (16).

$$P_k^- = A_{k-1} \cdot P_{k-1}^+ \cdot A_{k-1}^T + M_{k-1} \cdot Q_{k-1} \cdot M_{k-1}^T \quad (16)$$

- Achar o ganho de Kalman, equação (17).

$$K_k = P_k^- \cdot C_k^T \cdot [C_k \cdot P_k^- \cdot C_k^T + N_k \cdot R_k \cdot N_k^T]^{-1} \quad (17)$$

- Calcular o estado a posteriori, equação (18).

$$\tilde{X}_k^+ = \tilde{X}_k^- + K_k \cdot [Y_k - h(\tilde{X}_k^-, 0, k)] \quad (18)$$

- Cálculo da covariância a posteriori, equação (19).

$$P_k^+ = [I - K_k \cdot C_k] \cdot P_k^- \quad (19)$$

- Retorna ao segundo item.

3.3.1 Aplicação à Robótica Móvel

Para a simulação neste trabalho foram feitas algumas considerações, como por exemplo, a simplificação de Q e R para matrizes fixas. Essas matrizes representam a confiabilidade no modelo e no sistema de medição, pois ambos afetam diretamente o ganho de Kalman K_k . Quanto maior for o ganho K_k , maior será o peso das correções das medições sobre as estimações do modelo.

Ainda é importante mencionar que os valores iniciais de P e X, e as matrizes Q e R só são possíveis de serem exatamente estimadas através de uma identificação do sistema, portanto para cada aplicação é necessária uma calibração dos valores.

O modelo robô utilizado foi baseado no modelo cinemático de um robô diferencial. Assim, a matriz de estados do sistema fica na forma da equação (21) que equivale à equação (22).

$$X_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ w_k \end{pmatrix} \quad (20)$$

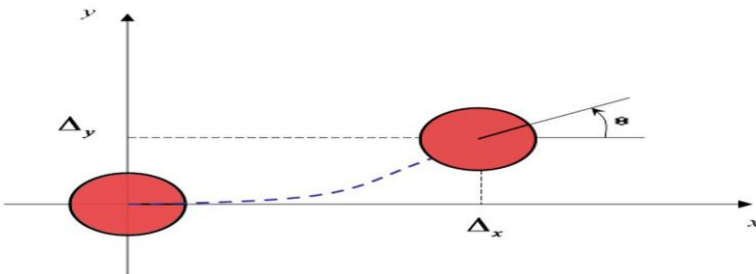


Figura 9 - Pose de um robô.

A pose do robô consiste em sua posição no eixo x , sua posição no eixo y e sua orientação em relação ao eixo x , como demonstrado na Figura 9.

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ w_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ w_k \end{pmatrix} + \begin{pmatrix} v_k \cdot \Delta_T \cdot \cos\left(\theta_k + \Delta_T \cdot \frac{w_k}{2}\right) \\ v_k \cdot \Delta_T \cdot \sin\left(\theta_k + \Delta_T \cdot \frac{w_k}{2}\right) \\ \Delta_T \cdot w_k \\ 0 \\ 0 \end{pmatrix} \quad (21)$$

$$X_{k+1} = X_k + f(X_k) \quad (22)$$

As variáveis x , y e θ correspondem à posição e ao ângulo do robô, v e w são as velocidades linear e angular do robô, Δ_T o tempo de amostragem e X o vetor de estados do sistema.

O modelo de observação, baseada na leitura da IMU e da odometria, é descrito em (23) para o caso da leitura dos sensores e (24) para o caso da estimação baseado nos estados estimados.

$$Y_k = \begin{pmatrix} \Omega_{d_k} \\ \Omega_{e_k} \\ \lambda_k \\ \alpha_k \end{pmatrix} \quad (23)$$

$$h(\tilde{X}_k^-, 0, k) = \begin{pmatrix} 0 & 0 & 0 & 1/r_d & \beta/(2 \cdot r_d) \\ 0 & 0 & 0 & 1/r_e & -\beta/(2 \cdot r_e) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \tilde{x}_k^- \\ \tilde{y}_k^- \\ \tilde{\theta}_k^- \\ \tilde{v}_k^- \\ \tilde{w}_k^- \end{pmatrix} \quad (24)$$

No modelo de observação, as variáveis medidas pela odometria são as velocidades angulares da direita e esquerda (Ω_d e Ω_e). As variáveis medidas pela IMU são a velocidade angular α e a velocidade linear λ . A variável β é a distância entre as duas rodas, r é o raio das rodas e Y é o vetor de observação.

Um aspecto importante nesta formulação é a correção do filtro que é baseada na diferença entre a leitura obtida dos sensores Y_k e um vetor de observação \tilde{Y}_k criado a partir dos estados estimados, utilizando a função $h(\tilde{X}_k, 0, k)$. Algumas outras variações também podem ser utilizadas, como por exemplo, fazer a correção no filtro baseado no erro resultante da diferença entre a leitura das velocidades da odometria e a leitura das velocidades pela IMU.

Adicionando os valores na equação (18) (do EKF), a nova equação fica na forma de (25).

$$\begin{pmatrix} \tilde{x}_k^+ \\ \tilde{y}_k^+ \\ \tilde{\theta}_k^+ \\ \tilde{v}_k^+ \\ \tilde{w}_k^+ \end{pmatrix} = \begin{pmatrix} \tilde{x}_k^- \\ \tilde{y}_k^- \\ \tilde{\theta}_k^- \\ \tilde{v}_k^- \\ \tilde{w}_k^- \end{pmatrix} + K_k \cdot \left[\begin{pmatrix} \Omega_{d_k} \\ \Omega_{e_k} \\ \lambda_k \\ \alpha_k \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 1/r_d & \beta/(2 \cdot r_d) \\ 0 & 0 & 0 & 1/r_e & -\beta/(2 \cdot r_e) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \tilde{x}_k^- \\ \tilde{y}_k^- \\ \tilde{\theta}_k^- \\ \tilde{v}_k^- \\ \tilde{w}_k^- \end{pmatrix} \right] \quad (25)$$

3.4 APLICAÇÕES E CONCLUSÃO

A aplicação de fusão de dados em sensores já é realidade em muitos sensores de última geração. As IMU fornecem opção para que se conectem sensores de odometria e se realize a fusão no próprio sensor, evitando gastos computacionais para o controle central.

Esta nova maneira de se projetar os sensores também possibilita uma maior modularização do robô, sendo que vários componentes podem ser reaproveitados em diversos robôs sem a necessidade de uma total reconfiguração.

Este conceito também foi utilizado por Norvig (2010) em seus carros automatizados, campeões do desafio DARPA nos Estados Unidos.

Neste trabalho é realizada a fusão de dados por EKF entre a odometria e a IMU para melhorar a estimação da trajetória do robô. Assim, o vetor de estados corrigido a cada instante k é repassado para um algoritmo de SLAM que utiliza estes dados em vez de uma leitura direta do sensor de apenas um sensor.

4 LOCALIZAÇÃO E NAVEGAÇÃO

O entendimento da localização e da navegação é importante para a compreensão do SLAM. A localização auxilia na construção do algoritmo do SLAM e na forma que o robô e o mapa são representados. Os conceitos da navegação, por outro lado, são utilizados para construção da trajetória e para criar critérios para avaliar os mapas gerados pelo SLAM.

Nesta primeira parte, é descrito a navegação de forma global e os conceitos que são utilizados na navegação e na localização. Após a representação do robô e do ambiente são descritas, com foco na representação por grade de ocupação, que é a representação utilizada neste trabalho. Ao final é discutido em detalhes o método de criação de caminho pelo Wavefront e dos algoritmos que originaram o mesmo.

Quando se trabalha com navegação em robótica móvel, pode-se ter dois tipos de abordagens: uma baseada em localização e mapas pré-definidos; e outra estabelecida por comportamentos (Choset 2005).

A navegação baseada em comportamentos assume que os sensores e efetadores possuem muitos ruídos, informações limitadas ou baixa capacidade de processamento, assim se deve evitar construir um mapa geométrico para localização. Alternativamente, sugere-se projetar um conjunto de comportamentos para navegar. Nesta técnica é suposto que exista uma solução pré-definida para o problema de navegação apresentado. Como por exemplo, ao se tentar ir de uma sala A a uma sala B, ao invés de se projetar um mapa geométrico e se localizar, pode-se fazer o robô seguir a parede e ao chegar à sala ativar algum sensor, indicando ao robô que a navegação foi concluída.

A solução pode ser obtida de maneira simples e rápida para ambientes pequenos e com pequenos números de objetivos. Mas sofre grandes desvantagens, a primeira é que o método de navegação não pode ser transportado facilmente para outros ambientes, ou ser aplicado a ambientes maiores.

Em contraste, a navegação baseada em localização inclui um módulo de cognição e localização. Neste tipo de navegação o robô explicitamente tenta se localizar a partir dos dados do sensor, atualizando assim uma crença (*Belief*) a respeito de sua posição no mapa (Choset 2005).

Ao ser explícita, a localização do robô fica disponível a qualquer momento para o operador humano e a própria existência de um mapa

possibilita a mudança fácil de comandos durante a navegação e a adaptação para um novo ambiente.

A desvantagem está no grande custo de desenvolvimento inicial,, que pode ser compensado pela facilidade de adaptação. Além disso, é assumido pelo robô que o mapa foi desenvolvido e informado corretamente (Thrun 2005).



Figura 10 - Navegação baseada em comportamentos.

Fonte: Adaptado de Choset (2005).

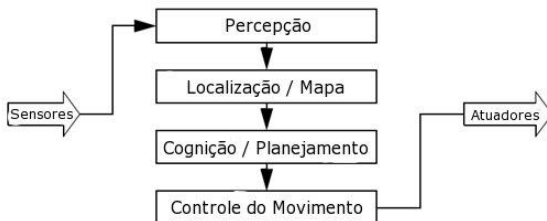


Figura 11 - Navegação baseada em localização.

Fonte: Adaptado de Choset (2005).

4.1 REPRESENTAÇÃO DE CRENÇAS

A principal questão que diferencia os vários tipos de sistema de localização baseados em mapas é a representação do ambiente e a representação das crenças.

A primeira está relacionada à como o ambiente no qual o robô está inserido será representado e a segunda está relacionada à crença de sua posição no mapa, se o robô identifica apenas uma posição como sua posição atual ou se há múltiplas opções, e neste caso, como as múltiplas posições estão organizadas.

As decisões envolvendo estas duas questões resultam em uma vasta gama de complexidade arquitetural, computacional e de precisão. Os dois principais ramos existentes são o de hipótese única e de múltiplas hipóteses.

Na Figura 12 são indicados alguns tipos de representação de crenças a respeito da posição do robô, sendo (a) um mapa contínuo com crença de hipótese única, (b) mapa contínuo com crença de múltiplas hipóteses, (c) mapa de grade discreto com valores probabilísticos de todas as possíveis posições do robô e (d) um mapa topológico discreto com valores probabilísticos para os possíveis nós.

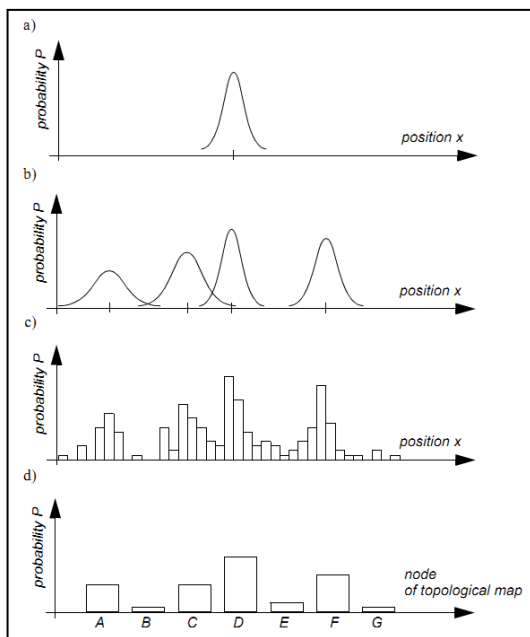


Figura 12 - Representação de crenças a respeito da posição do robô.

Fonte: Siegwart (2004)

4.1.1 Crença de Hipótese Única

Este tipo de representação é a mais direta possível da posição do robô, pois a apresenta como um único ponto no mapa.

A principal vantagem desta representação é que não há abertura para ambiguidades, portanto é facilitada a atualização da crença do robô em relação a sua posição. Isto ajuda na decisão no nível cognitivo do

robô, pois este assume que a crença está correta e baseia suas ações futuras na sua única posição.

A desvantagem está no fato de que os sensores e atuadores geralmente possuem ruídos que geram incertezas a respeito da posição do robô. Portanto é muito difícil atualizar uma única posição e assumir que está correta.

4.1.2 Crença de Múltiplas Hipóteses

Nesta representação são assumidas várias hipóteses da posição atual do robô. A representação da posição do robô pode ser feita por vários tipos de distribuições, como a gaussiana. A distribuição em cada posição representa a probabilidade do robô estar naquele local.

Alternativamente podem-se utilizar marcadores discretos para cada possível posição. Neste caso cada posição do robô possui um nível de confiança ou de probabilidade.

A vantagem das técnicas de múltiplas hipóteses é que o robô explicitamente mantém a incerteza em relação a sua posição, permitindo que informações parciais sejam incorporadas a atualização da crença.

Como observado na Figura 13, a imagem da esquerda representa o movimento real do robô, já a partir da segunda imagem é representada as possibilidades sobre a posição do robô que é cada vez mais restrita a medida que este se desloca e recebe a leitura dos sensores.



Figura 13 - Exemplo de localização de múltiplas hipóteses.

Fonte: Adaptado de Siegwart (2004)

4.2 REPRESENTAÇÃO DO MAPA

O problema de representar o ambiente no qual robô se locomove afeta diretamente na dificuldade de representação da posição do robô. Frequentemente a fidelidade da representação da posição do robô está ligada à fidelidade do mapa.

Três relações devem ser corretamente observadas ao se escolher uma representação do mapa: a precisão deve corresponder à precisão necessária para o robô atingir seus objetivos; a precisão do mapa e o tipo de recursos representados devem corresponder à precisão e aos tipos de dados retornados pelos sensores do robô; e por último, a complexidade da representação do mapa tem um impacto direto na complexidade computacional de processar o mapeamento, localização e a navegação.

4.2.1 Representação Contínua

A representação contínua é um método de decomposição exata do ambiente, ou seja, uma posição no ambiente é descrita precisamente no espaço contínuo, representando todos os objetos do ambiente. Atualmente, esta técnica é utilizada apenas para mapas 2D devido ao custo computacional. caso a redução do processamento seja necessária, o mapa pode ser simplificado/decomposto em apenas o necessário para a navegação do robô, como no caso da Figura 14, onde o mapa é decomposto em algumas linhas.

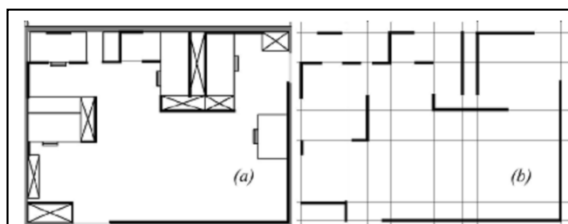


Figura 14 – Simplificação do mapa contínuo.

Fonte: Choset (2005)

4.2.2 Estratégias de Decomposição Comuns

Uma forma de simplificação de mapas é a abstração, onde se faz uma decomposição selecionando apenas algumas características importantes do ambiente. A desvantagem deste tipo de técnica é a perda

de fidelidade entre o mapa e o mundo real e a principal utilidade é a redução computacional e simplificação, caso o planejamento da abstração seja cuidadosamente feito (Choset 2005).

Uma das técnicas de abstração é a de decomposição por células exatas, onde o ambiente é aproximado pelos limites definidos pelas células em pontos geométricos críticos, como indicado na Figura 15.

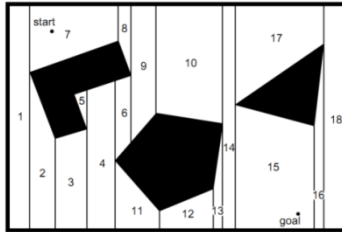


Figura 15 - Exemplo de decomposição exata de célula.

Fonte: Choset (2005)

Cada área livre é representada como um nó, desta forma o robô não necessita saber a sua posição na área, e sim apenas a habilidade do robô para atravessar de uma área para qualquer área adjacente.

Para casos em que a geometria do ambiente não é necessária, há uma alternativa chamada de decomposição topológica, que evita a medição geométrica direta do ambiente.

Formalmente a representação topológica é um grafo que especifica duas coisas: nó e conectividades entre estes nós. Os nós são utilizados para denotar áreas no ambiente e os arcos são usados para denotar adjacência entre um par de nós. Quando um arco conecta dois nós, significa que o robô pode passar de um nó a outro sem atingir obstáculos (Choset 2005), um exemplo é a Figura 16.

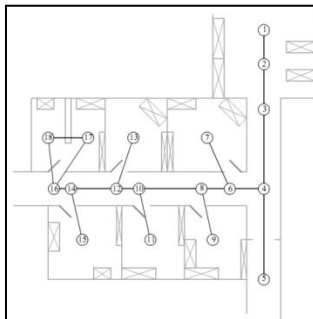


Figura 16 - Exemplo de mapa topológico.

Fonte: Choset (2005)

Para navegar pelo mapa topológico robustamente, o robô deve satisfazer duas condições. Primeiro que o mesmo deve ter meios para detectar a sua posição atual em termos de nó do grafo topológico. Segundo, ele deve ter meios para navegar entre os nós.

Um meio de contribuir para mapas topológicos é a inserção de marcadores visuais, para sistemas que possuem sensores visuais, auxiliando assim, na criação de nós para o grafo.

4.2.3 Mapa de Grade

Uma forma popular de decompor o ambiente é a decomposição física onde o mundo é discretizado em células que serão preenchidas de acordo com a presença de obstáculos. A desvantagem é que lugares estreitos que poderiam oferecer passagem ao robô não são mostrados na decomposição (Figura 17). Há também uma versão com tamanho de células variável chamada de decomposição com células adaptativas (Figura 18).

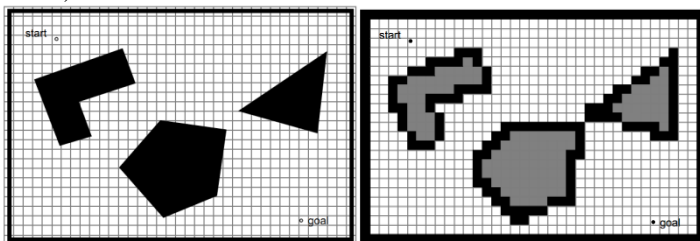


Figura 17 - Exemplo de decomposição fixa.

Fonte: Choset (2005)

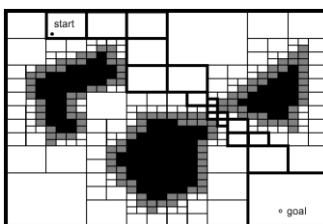


Figura 18 - Decomposição com células adaptativas.

Fonte: Choset (2005)

Uma versão popular da discretização fixa é a grade de ocupação (*Occupancy Grid*) onde o ambiente é discretizado em pequenas células indicando se há presença ou não de obstáculos (Figura 19). Esta técnica

é particularmente eficaz se o robô está equipado com sensores de distância.

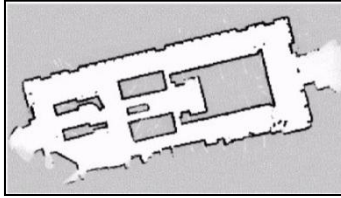


Figura 19 - Exemplo de mapa por ocupação de grade.

Fonte: Grisetti (2007)

Nesta técnica as células são atualizadas diretamente pelo sensor, que são preenchidas podem assumir valores entre 0 e 1, sendo o valor “0” quando há certeza que não há obstáculos, 0,5 quando a presença de obstáculo é desconhecida e o valor de “1” quando se tem certeza que há um obstáculo. A célula pode ser decrementada ao receber leitura de livre pelo sensor ou incrementada se receber um valor positivo. Isto possibilita detectar obstáculos transientes.

A desvantagem ao se usar decomposição fixa é a necessidade de se utilizar uma memória a parte para armazenar os dados da célula e que é necessário estabelecer um tamanho de grade *a priori*, independente dos detalhes do ambiente. Também não é indicado para casos onde a geometria não é a informação mais saliente do ambiente.

Pela facilidade de representação que este método possui, este foi escolhido como forma de representação neste trabalho. O robô percorre as posições de forma contínua que então são discretizadas para o seu equivalente no mapa de grade.

4.3 MÉTODOS DE NAVEGAÇÃO

A navegação na robótica móvel pode ser dividida em duas grandes áreas: a navegação em uma área já conhecida, onde é necessário apenas calcular a melhor trajetória até o ponto desejado e a navegação em regiões desconhecidas, onde busca-se realizar um planejamento dentro do que se conhece do ambiente para explorar adequadamente e atingir um ponto desejado na região desconhecida. Esta seção revisa brevemente os métodos de navegação propostos nestas duas áreas, focando apenas no Wavefront. Foi utilizado como referência o material de Choset et al. (2005), Norvig (2011) e Thrun (2005).

4.3.1 Navegação em Regiões Exploradas

As técnicas mais utilizadas de navegação em regiões conhecidas são as funções potenciais (ou de custo) e os Roadmaps. A primeira é baseada na representação em grade e a segunda pode ser utilizada para representações topológicas, contínuas ou de grade.

Os Roadmaps são algoritmos que criam rotas e nós, o robô vê um algoritmo de Roadmap como um carro vê uma rota de viagem, podendo somente utilizar as rodovias construídas para transitar entre as cidades.

Os nós e os trajetos podem se basear na forma geométrica do ambiente, em pontos de referência ou análise probabilística.

A função potencial é uma função diferencial com valor real que pode ser vista como uma energia e, portanto o seu gradiente é a força. O gradiente é um vetor que aponta em direção na qual localmente maximiza uma função. Esse gradiente é usado para criar um campo vetorial, que atribui um vetor para cada ponto no mapa.

Os gradientes podem ser vistos intuitivamente como funções que direcionam o robô como se fosse uma força atuando em uma partícula carregada positivamente sendo atraídas para um objetivo carregado negativamente.

O problema desta técnica é a existência de mínimos locais, que não levam o robô ao ponto desejado. Para resolver este problema se pode construir o campo vetorial com alguma modalidade de busca planejada. Apesar de ser uma solução completa, este exige o conhecimento completo do ambiente antes do planejamento.

Os algoritmos de planejamento podem ser estocásticos ou determinísticos.

O planejador estocástico considera que há uma probabilidade do movimento planejado para o robô não sair conforme o planejado, fazendo com que o robô possa ir para uma célula diferente.

O algoritmo estocástico mais simples utilizado atualmente é chamado de processo de decisão de Markov (Markov Decision Process - MDP) (Thrun 2006).

Aplicado à navegação, este algoritmo assume que ao mudar de estado, o robô pode ter uma probabilidade grande de seguir o caminho determinado previamente. Porém há chances do mesmo se desviar do caminho e parar em outras células.

O algoritmo determinístico não considera várias possibilidades, sendo que o seu planejamento é simples e direto. Foi escolhido trabalhar

com algoritmo determinístico para não inserir um grau de complexidade além do necessário, já que o trajeto a ser realizado é pequeno e é utilizado apenas para criar a trajetória de referência.

O planejador determinístico aplicado para os mapas de grade possui algumas formas de calcular a função de custo que podem variar de acordo com a heurística do planejador.

Primeiramente, o planejador escolhe como determinar a relação entre as vizinhanças de uma célula, podendo optar pela escolha de vizinhança direta ao norte, sul, leste e oeste (conectividade de 4 pontos) ou considerar também as diagonais (conectividade de 8 pontos) (Figura 20).

n1	n2	n3	n1	n2	n3
n4	n5	n6	n4	n5	n6
n7	n8	n9	n7	n8	n9

Figura 20 - Conectividade de 4 e 8 pontos.

Um dos primeiros algoritmos de planejamento é o Brushfire (Figura 21), onde é atribuído o valor 1 para as células ocupadas, sendo incrementado o valor de suas vizinhanças livres, que depende da conectividade. Assim, segue-se a propagação dos valores e o incremento das vizinhanças livres até o mapa estar completamente preenchido.

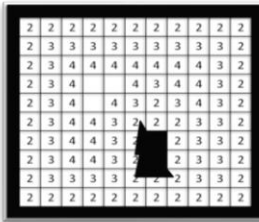


Figura 21 – Exemplo do algoritmo Brushfire.

Este método até o momento não trata com o problema dos mínimos locais, para isto é utilizado o método Wave Front. Este método assume que as células da posição inicial e final são conhecidas, atribuindo ao objetivo o valor de 2 e preenchendo a vizinhança de forma semelhante ao Brushfire, mas partindo do objetivo, até todos os espaços livres obterem um valor. Essa solução resolve o problema de mínimos,

mas requer tempo e armazenamento que é exponencial aos numero de dimensões do espaço.

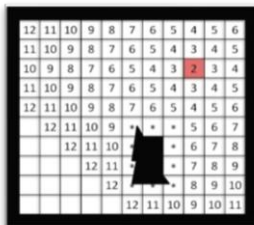


Figura 22 - Algoritmo Wave Front.

Este algoritmo considera células parcialmente preenchidas como obstáculos e dependendo da margem de segurança que se deseja ter e do tamanho das células, pode-se determinar uma margem de uma ou mais células como células com obstáculo para evitar possíveis contatos entre o robô e um obstáculo.

4.3.2 Navegação em Regiões Não Exploradas

Para regiões não exploradas geralmente são utilizados algoritmos que buscam fronteiras, ou seja, o limiar entre a região conhecida (valores com 0 e 1 na grade) e o valor de região desconhecida (0,5).

Um algoritmo que se destaca nesta formulação, é o de Fronteira mais próxima, o qual sempre direciona o robô para a fronteira com menor distância ao robô. Assim que este atinge a posição determinada, faz-se uma nova leitura do ambiente, são verificadas se novas informações foram adquiridas e determinam-se a existência de outras possíveis fronteiras próximas e segue para o seu próximo objetivo.

Uma vasta revisão destes métodos pode ser obtida em Holz (2010).

4.4 APLICAÇÃO E CONCLUSÕES

Este capítulo teve como objetivo, primeiramente revisar alguns conceitos de representação de mapas e de representação de crenças para um melhor entendimento sobre a atuação do SLAM, já que o algoritmo

utilizado neste trabalho tem uma abordagem de múltiplas crenças e utiliza mapas de grade para representar o ambiente e o robô.

Como mencionado anteriormente, foi escolhido utilizar a representação por mapa de grade pela facilidade de incorporação ao SLAM e para ser possível construir uma ferramenta para avaliação das simulações realizadas. Este tipo de representação também pode ser facilmente integrado a redes Bayesianas, e no Apêndice C é realizada uma revisão sobre as redes e de como podem ser integradas ao mapa de grade.

A navegação do robô é necessária para fazer o mesmo se locomover pelo ambiente, sendo utilizado o algoritmo Wave Front para realizar um movimento básico, podendo futuramente ser trocado por algoritmos probabilísticos para uma melhor integração com o algoritmo de SLAM. Vale ressaltar a importância de saber como avaliar é possível avaliar, de uma forma autônoma, os resultados obtidos pelo mapeamento de uma região, principalmente, de como isto afetaria uma possível navegação do robô.

A margem de segurança geralmente criada por planejadores possibilita a suposição que a navegação não sofrerá prejuízo caso haja um erro de mapeamento com espaço de uma célula. A avaliação dos mapas é descrita em detalhes em capítulos posteriores.

No próximo capítulo é apresentada uma introdução ao SLAM e ao algoritmo utilizado no trabalho.

5 SLAM

O capítulo engloba uma breve revisão sobre o problema do SLAM e as atuais soluções para que na sequência os conceitos básicos de um filtro de partículas sejam revistos, por estes serem a base do algoritmo utilizado neste trabalho, o qual é detalhado no item subsequente. Finalmente, é descrita a integração entre o algoritmo de SLAM e o de fusão de dados e as considerações finais.

A localização e mapeamento simultâneo (SLAM) é um problema fundamental na robótica e provém do problema do robô não ter acesso ao mapa do ambiente e de não saber a sua própria localização. As soluções para o SLAM surgiram da evolução dos algoritmos para mapeamento de ambientes com posições conhecidas e da localização do robô em um mapa pré-definido.

Em relação ao mapeamento, a sua origem começou de duas formas, sendo a primeira com o trabalho de Smith e Cheeseman (1986), no qual sugeriram o filtro de Kalman para o problema do SLAM. Esse algoritmo foi primeiramente implementado por Moutarlier e Chatila (1989), possuindo algumas variações e aprimoramentos nos anos posteriores. A segunda origem começou com o desenvolvimento da representação por grades de ocupação, para mapas probabilísticos, no qual específica à probabilidade de cada posição (x, y) estar ocupada por um obstáculo (Moravec e Elfes, 1985).

Quanto à localização, as primeiras técnicas que surgiram são revisadas por Borestein (1996). Uma destas, o filtro de Kalman, foi um conhecido método na teoria de controle por décadas, mas a formulação da teoria no problema da localização não apareceu na literatura até muito tempo depois, através do trabalho de Dean (1990) e de Simmons (1995). Alguns trabalhos posteriores introduziram o termo “Localização de Markov” e a primeira aplicação foi feita por Burgard (1999), aplicando os algoritmos desenvolvidos em robôs para museu. A localização de Monte-Carlo por filtro de partículas foi desenvolvido por Fox (1999) e hoje é amplamente utilizado na localização (Murphy 2001) (Montemerlo 2002).

Também há abordagens probabilísticas para o SLAM que possibilitam a quantificação das variações e da incerteza através do uso de distribuições em vez de valores fixos. Uma ampla revisão sobre essas técnicas probabilísticas pode ser obtida em Thrun (2006).

Do ponto de vista probabilístico, há dois tipos diferentes de problemas de SLAM. Um é conhecido como SLAM Online e envolve

estimar as variáveis posteriores baseado nas variáveis no instante k (26).

$$p(X_k, \phi | Y_{1:k}, u_{1:k}) \quad (26)$$

Onde X_k é a pose no instante k , ϕ é o mapa, e $Y_{1:k}$ e $u_{1:k}$ são as medições e os controles, respectivamente. Algoritmos nesta forma são incrementais, ou seja, eles descartam medições passadas e controles assim que foram processados.

O segundo tipo de SLAM é chamado de Full SLAM, nesse procura-se estimar os estados posteriores baseado em todo o caminho de $X_{1:k}$, junto com o mapa, em vez de apenas se basear em X_k .

$$p(X_{1:k}, \phi | Y_{1:k}, u_{1:k}) \quad (27)$$

Os primeiros modelos de SLAM baseavam-se nos filtros de Kalman estendidos (EKF – Extended Kalman filter) e usavam esse filtro no SLAM online utilizando o método de associação de dados por máxima verossimilhança (*maximum likelihood*) (Thrun, 2006). Estes algoritmos necessitam de marcadores ou de detecção de características do ambiente (como a identificação de paredes, rodovias, linhas, lâmpadas, etc.).

Uma alternativa para o EKF SLAM é o GraphSLAM. Em contraste, este pode ser utilizado para o problema de SLAM completo. Nesse tipo de SLAM é formada uma espécie de grafo, contendo pontos e arcos. Cada ponto representa a pose do robô em dado momento, e os arcos são as restrições dadas pelo modelo do movimento e pela leitura de marcadores.

Uma analogia comum é a associação a cordas de elástico representando o caminho total, sendo formado de pequenos arcos que representam a ligação entre as poses e as leituras. Esse caminho “elástico” poderá ser “movimentado” para melhor adequar as ligações.

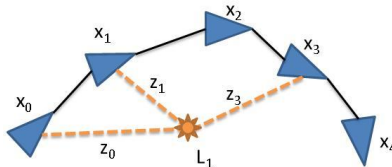


Figura 23 - Exemplo de trajeto no GraphSlam.

Apesar de serem soluções aceitáveis de SLAM, esses algoritmos dentre outros, foram gradualmente substituídos por algoritmos mais complexos, de critério de múltiplas crenças, como o filtro de partículas.

5.1 FILTRO DE PARTÍCULAS

O filtro de partícula (*PF – Particle Filter*) é um método numérico de integração adequado para solução de problemas não lineares. Nesta seção é apresentado o filtro de partículas em seus conceitos básicos, baseado em Aiube (2005) e Thrun (2006), que consiste na base do algoritmo descrito no próximo item, que é algoritmo utilizado neste trabalho.

O filtro de partículas tem a mesma concepção básica do filtro de Kalman, sendo o KF visto como um caso particular em que o modelo é linear e as distribuições das variáveis de estado e de observação são Gaussianas.

O PF aborda uma classe mais ampla de problemas, pois não tem as mesmas restrições do KF, mas por outro lado, possui o problema de ter um custo computacional alto.

De forma sucinta, um algoritmo de PF possui várias partículas que representam várias possibilidades sobre o estado do sistema. Na localização de robôs, por exemplo, cada partícula tem a pose estimada do robô para o instante k .

A cada interação é feita uma predição do novo estado individualmente em cada partícula, baseado em um modelo de transição associada a uma distribuição. Ao final das interações, cada partícula recebe um peso, este peso é dado comparando a predição do modelo com a medição atual nos sensores. No final da interação ou a cada n interações, é feito um algoritmo de reamostragem, onde as partículas com maior peso têm maior chance de serem escolhidas e gerarem descendentes não idênticos, mas próximos de seus valores.

Esse procedimento é comumente conhecido como método sequencial de Monte-Carlo (MC), mas é citado na literatura também como filtros *bootstrap*, condensação, filtro de partículas ou filtro de Monte-Carlo (Aiube 2005).

Em Thrun (2006) é apresentado o algoritmo denominado Fast SLAM, baseado em FP para estimar o trajeto do robô. Para cada uma das partículas o erro individual de mapeamento é condicionalmente independente, ou seja, o problema de mapeamento pode ser dividido em pequenos problemas separados, um para cada marcador no mapa. O FastSLAM estima a localização dos marcadores usando um EKF de pequena dimensão para cada marcador.

Há variações no algoritmo do FastSLAM, mas no Quadro 3 é descrito os principais pontos do algoritmo, no qual apenas é modificado a forma de predição ou de correção. Quando o algoritmo é alterado para o PF acompanhar todo o trajeto do robô, relacionando as variáveis restantes do sistema por distribuições gaussianas, o método recebe o nome de filtro de partículas Rao-Blackwellizado.

Faça i vezes:

- Aquisição: Adquira a pose $X_{k-1}^{[n]}$ do conjunto de partículas Y_{k-1} .
- Predição: Amostre uma nova pose baseado no estado anterior e o sinal de controle: $X_k^{[n]} \sim p(X_k | X_{k-1}^{[n]}, u_k)$.
- Correção: Para cada marcador ou nova informação Y_k , corrija os valores. Atualize o EKF correspondente ao marcador atualizando a média e a variância.
- Peso: Calcule o peso da partícula $\omega^{[n]}$ para a nova partícula.
- Re-Amostre: Amostre i partículas (Com substituição) com probabilidade proporcional ao peso.

Quadro 3 - Algoritmo do FP FastSLAM.

Fonte: Thrun (2006)

Através da Figura 24, Figura 25, Figura 26 e Figura 27, é possível observar um exemplo de solução da localização do robô através de um filtro de partículas.

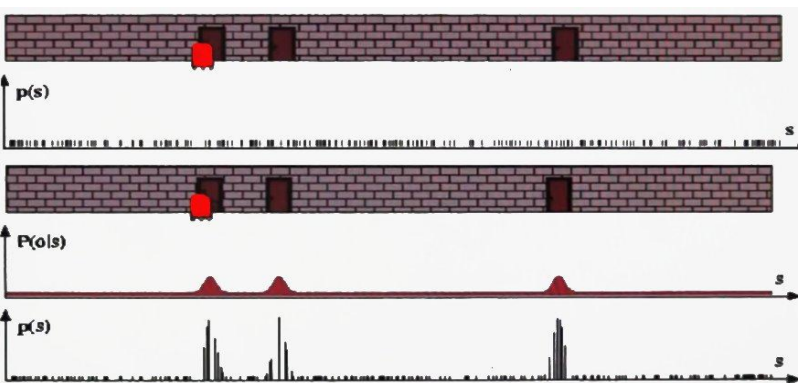


Figura 24 - Primeira Etapa do FP para Localização.

Fonte: Thrun (2012)

Neste exemplo é assumido um robô que possui sensores que apenas detectam portas, sendo que cada pequena barra vertical é uma hipótese (partícula) a respeito da posição atual do robô, que foi distribuída aleatoriamente pelo ambiente.

Na Figura 24 é realizada a leitura de uma porta, sendo atribuído um peso maior para as partículas localizadas nas regiões que possuem portas.

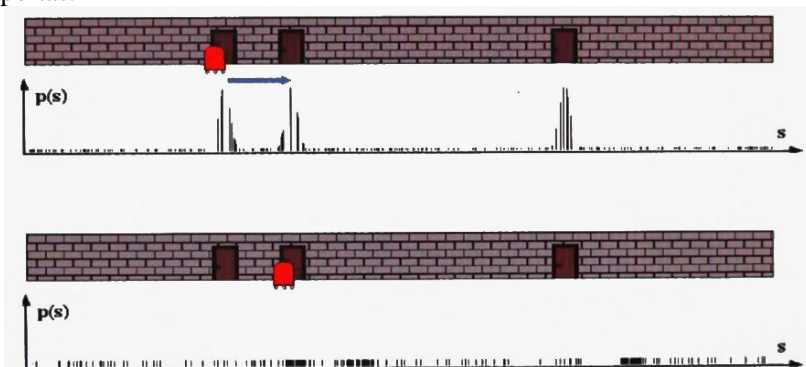


Figura 25 - Segundo Etapa do FP para Localização.

Fonte: Thrun (2012)

Na segunda etapa (Figura 25) é realizada a transição das partículas que acompanham o movimento do robô. Ao final da transição é realizada a reamostragem, onde é gerado um novo conjunto de partículas. As partículas de maior peso possuem uma chance maior de sobreviver e de gerar partículas descendentes, onde haverá uma maior concentração de partículas onde havia partículas com maior peso.

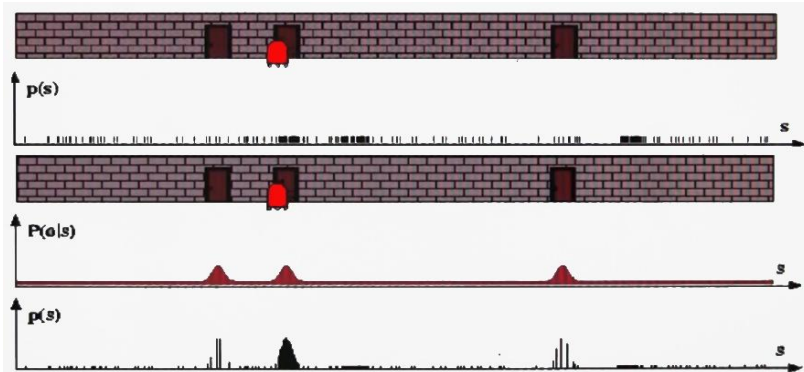


Figura 26 – Terceira Etapa do FP para Localização.

Fonte: Thrun (2012)

Na terceira etapa (Figura 26) é novamente detectada uma porta, sendo que agora um número grande de partículas recebe um peso na segunda porta.

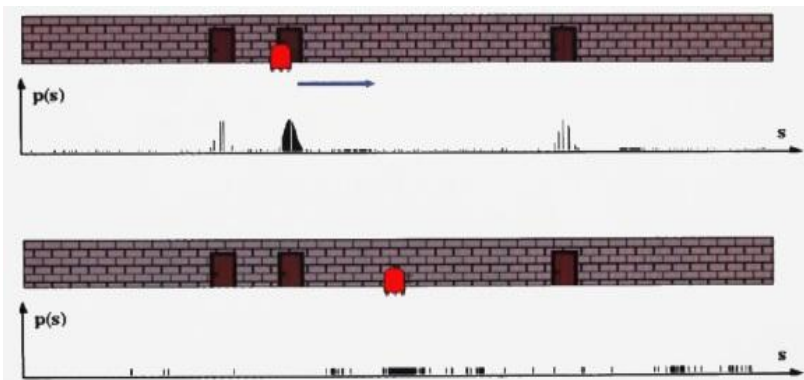


Figura 27 – Quarta Etapa do FP para Localização.

Fonte: Thrun (2012)

Na última etapa deste exemplo, quando o robô realiza o seu deslocamento, as partículas acompanham e geram novos descendentes. Agora a grande maioria das partículas se concentrou em torno de um ponto, ou seja, a maior parte da capacidade de processamento está concentrada em torno do robô.

5.2 GRID FASTSLAM

Para adaptar o FastSLAM para grades de ocupação, foi criado por Thrun (2006) o *Grid Based FastSLAM (GBFS)*. Este é o algoritmo de SLAM utilizado neste trabalho devido ao fato de ser facilmente modularizado em algumas funções, o que possibilitou ser implementado facilmente e incorporado à fusão de dados de diversas formas sem necessidade de grandes alterações.

O GBFS compreende três estágios fundamentais:

- **Transição das Partículas:** é responsável pela computação da evolução das partículas a partir das entradas de controle, modelo do robô e dos dados dos sensores (IMU e odometria). Em outras palavras, atualiza as variáveis de estado de cada partícula do instante anterior para o instante atual;
- **Construção do Mapa:** responsável pela atualização do mapa de grade associado a cada partícula;
- **Reamostragem das Partículas:** é um procedimento que elimina as partículas ruins e replica as partículas que mais contribuem para o resultado correto.

Um resumo do algoritmo está descrito no Quadro 4, sendo $\omega_k^{[n]}$ o peso da partícula (não confundir com “w” utilizado para velocidade angular no capítulo sobre fusão), η é a partícula atual, `num_partic` é o número total de partículas, $\phi_k^{[n]}$ é o mapa gerado do instante 0 ao k para a partícula η , $X_k^{[n]}$ é o vetor de estados do robô, u_k é o sinal de controle e Y_k o vetor de observação.

As implementações de cada estágio serão detalhadas nas próximas seções e mais detalhes sobre a teoria pode ser encontrada em Thrun (2006).

Nesse trabalho assumiu-se que os sistemas de odometria e da IMU possuem uma mecanização, em que a informação fornecida por estes sensores já foi convertida em velocidade angular e linear. Assume-se também que há um conjunto de sensores ao redor do robô que consegue identificar marcadores pré-definidos, fornecendo uma posição relativa do robô ao marcador e que o sistema de sensores que mede a distância consegue medir 360° em torno do robô.

```

1: Algoritmo FastSLAM_grade_ocupação( $X_{k-1}, u_k, Y_k$ ):
2:    $\bar{\Psi}_k = \Psi_k = \emptyset$ 
3:   for k = 1 to num_partic do
4:      $X_k^{[n]} = \text{amostre\_modelo\_transição}(u_k, X_{k-1}^{[n]})$ 
5:      $m_k^{[n]} = \text{atualize\_grade\_ocupação}(Y_k, X_k^{[n]}, \phi_{k-1}^{[n]})$ 
6:      $\omega_k^{[n]} = \text{peso\_dado\_leitura}(Y_k, x_k^{[n]}, \phi_{k-1}^{[n]})$ 
7:      $\bar{\Psi}_k = \bar{\Psi}_k + [X_k^{[n]}, \phi_k^{[n]}, \omega_k^{[n]}]$ 
8:   endfor
9:   for i = 1 to num_partic do
10:    retire partícula i com probabilidade  $\propto \omega_k^{[i]}$ 
11:    add  $\langle X_k^{[i]}, \phi_k^{[i]} \rangle$  a  $\Psi_k$ 
12:  endfor
13:  retorne  $\Psi_k$ 

```

Quadro 4 - Algoritmo do GBFS.

Fonte: Thrun (2006)

5.2.1 Transição das Partículas

A partir de uma condição inicial, com o conhecimento das entradas de controle, estima-se com o modelo cinemático do robô e a leitura dos sensores, o vetor de estados para o instante atual, ou seja, calcular $X_k^{[n]} \sim p(X_k | X_{k-1}^{[n]}, u_k)$.

Neste trabalho são utilizadas três formulações para esta estimativa, duas utilizando o sistema de fusão de dados entre a odometria e a IMU, e uma utilizando apenas a odometria. Para os três casos, este primeiro cálculo é utilizado para estimar o estado atual, utilizando geralmente o modelo do robô, sinais de controle e leitura dos sensores.

Para realizar a estimação, será utilizado o valor fornecido pela fusão de dados (\tilde{X}_k^+) para amostrar uma função de densidade probabilística, que neste trabalho, é uma distribuição Gaussiana (também conhecida como função Normal).

As duas primeiras formulações possuem um comportamento padrão, utilizando o valor fornecido pela fusão (\tilde{X}_k^+) como média na distribuição Normal. A diferença entre ambas um é estimado somente utilizando a odometria e o outro estimado utilizando a fusão entre odometria e IMU.

Nestes casos, a covariância da função Normal é fixa (denominada de \sum_x) e determinada ou por experimentos ou pela incerteza conhecida dos atuadores e sensores (geralmente a covariância da odometria). É importante que este valor tenha um valor mínimo. Do contrário, todas as partículas geradas terão aproximadamente o mesmo valor, tornando inútil o uso de partículas no algoritmo. A formulação final fica na forma de (28).

$$p(X_k | X_{k-1}^{[n]}, u_k) = N(\tilde{X}_k^+, \sum_x) \quad (28)$$

Na terceira formulação, a parte determinística também é obtida através da fusão de sensores entre IMU e Odometria. Mas a covariância utilizada provém da diagonal principal da matriz de covariância a posteriori da fusão de sensores, equação (19). A formulação final fica na forma de (29).

$$p(X_k | X_{k-1}^{[n]}, u_k) = N(\tilde{X}_k^+, P_k) \quad (29)$$

A amostragem dos valores foi feita utilizando a função “*normrnd*” do MATLAB, que gera um número aleatório de acordo com a distribuição normal, dados a média e desvio padrão.

5.2.2 Mapeamento

Após o cálculo da evolução de cada partícula, o mapa de grade associado a cada partícula é atualizado. O mapa associado a cada partícula possui uma estrutura matricial onde cada célula da grade corresponde a um elemento de uma matriz.

A cada elemento atribui-se um valor pertencente ao conjunto $[0,1]$, onde 0 designa um espaço livre, 1 designa um espaço com obstáculo e 0,5 indica que não há informação disponível sobre a célula.

A atualização de cada célula da grade é realizada a partir de um sistema de sensoriamento que detecta e mede a distância de obstáculos, considerando um referencial no veículo.

Ao detectar um obstáculo, a partir da distância e ângulo do obstáculo e da posição estimada de cada partícula, determina-se a qual

elemento pertence o obstáculo. Utilizando um procedimento Bayesiano como descrito em Thrun (2006) e Norvig (2012), a informação da célula é atualizada.

Para o procedimento Bayesiano de atualização, foi considerado que todas as células possuem o valor inicial $P(O) = 0,5$ e que pode ocorrer duas situações: o de haver uma leitura positiva ou negativa dos sensores a respeito da presença de obstáculo. Foi definida a confiabilidade do sensor em 60%, $P(+|O) = 0,6$, com a chance de haver um falso positivo de 40% , $P(+|\neg O) = 0,40$.

Assim, é calculada a probabilidade de haver um obstáculo dado uma leitura positiva, $P(O|+)$, e dada uma leitura negativa, $P(O|-)$.

Para este cálculo são utilizadas as equações da Probabilidade Total (Figura 30) e da Regra de Bayes (Figura 31).

$$P(+)=\sum_a P(+|O=a)P(O=a) \quad (30)$$

$$P(O|+)=\frac{P(+|O)\cdot P(O)}{P(+)} \quad (31)$$

A variação da célula a cada leitura é de aproximadamente 10% (0,1), limitando-se aos valores de 0 e 1. Mais detalhes sobre como é feito o cálculo da atualização estão no Apêndice C.

5.2.3 Pesos e Reamostragem

Após realizar a atualização do mapa de cada célula, é realizado um novo estágio de atribuir um peso a cada partícula e de escolher quais partículas serão preservadas, eliminadas ou duplicadas. A reamostragem é a parte fundamental do algoritmo, pois irá garantir que apenas as partículas de maior peso sobrevivam e gerem descendentes.

A atribuição de pesos para um conjunto de partículas pode ser realizada de diversas formas, dependendo da aplicação. Podem ser utilizados para análise os padrões encontrados no ambiente (como ruas, iluminação, marcadores, etc), ou analisando padrões nos mapas gerados (como paredes, portas, etc.) ou ainda utilizando qualquer outra heurística para determinar quais partículas estarão mais corretas e atribuir um peso maior a estas.

Neste trabalho para dar um peso para as partículas, foi considerado que o robô possui um sistema que detecta marcadores pré-

definidos, e que o sistema pondera a distância do marcador e a estimativa da posição feita por cada partícula.

Cada vez que o robô detecta um marcador, o sistema calcula o erro entre a posição estimada através do marcador e a estimativa da partícula (e_x , e_y), e utiliza a equação (32) para calcular o peso. A variável γ é um parâmetro que varia entre 0,9 e 1, de acordo com a distância do robô ao marcador.

$$\omega_k^{[n]} = \gamma \cdot \left(\frac{100 - 5(e_x^2 + e_y^2)}{100} \right) \quad (32)$$

Este sistema pode ser comparado a um sistema simples de visão computacional que detecta marcadores com posições pré-definidas. A solução por esta função foi considerada aceitável considerando o foco do trabalho em analisar impacto na função de transição com o uso da fusão de sensores. Após todas as partículas receberem um peso, é inicializada a etapa final de reamostragem, que consiste em sortear um novo conjunto de partículas, baseado nas existentes, com probabilidade da partícula ser sorteada proporcional ao seu peso.

A abordagem utilizada, criada por Thrun (2006), cria uma “roda de probabilidades”, onde todas as partículas possuem uma fatia desta roda de acordo com o seu peso (Figura 28).

O algoritmo é simples (Quadro 5) e escolhe uma partícula inicial com índice α para iniciar o sorteio e percorrer a roda com passo “B” (33), que depende de um número aleatório e do maior peso (w_m). Este procedimento é realizado até que o novo conjunto de partículas seja totalmente preenchido e desta forma, a partícula que tem uma fatia maior do círculo tem mais chance de ser escolhida do que as que possuem menor fatia.

$$B = B + \text{random}(0..1) \cdot 2 \cdot \omega_m \quad (33)$$

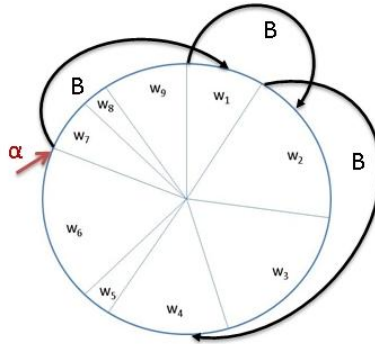


Figura 28 - Exemplo de roda de probabilidades com 9 partículas.

Algoritmo de Re-Amostragem

- Liste todas as partículas, sequencialmente;
- Sorteie um valor entre 1 e o número máximo de partículas (M) para ser o index inicial;
- $B = 0$ e w_m = peso da maior partícula;
- For $k=1:1:M$
 - $B = B + \text{random}(0..1) * 2 * \omega_m$
 - Enquanto $B < \omega$ [index]
 - $B = B - \omega$ [index]
 - Índice = mod(index + 1, M)
 - Novas_Partículas[k] = Velhas_Partículas[index]

Quadro 5 - Algoritmo de Reamostragem.

Fonte: Thrun (2012)

5.3 INTEGRAÇÃO DA FUSÃO DE DADOS E SLAM

O algoritmo proposto neste trabalho é o resultado da integração do filtro de Kalman Estendido descrito no Capítulo 2 e o FastGrid SLAM descrito neste capítulo. Na Figura 29 é possível observar o diagrama do algoritmo.

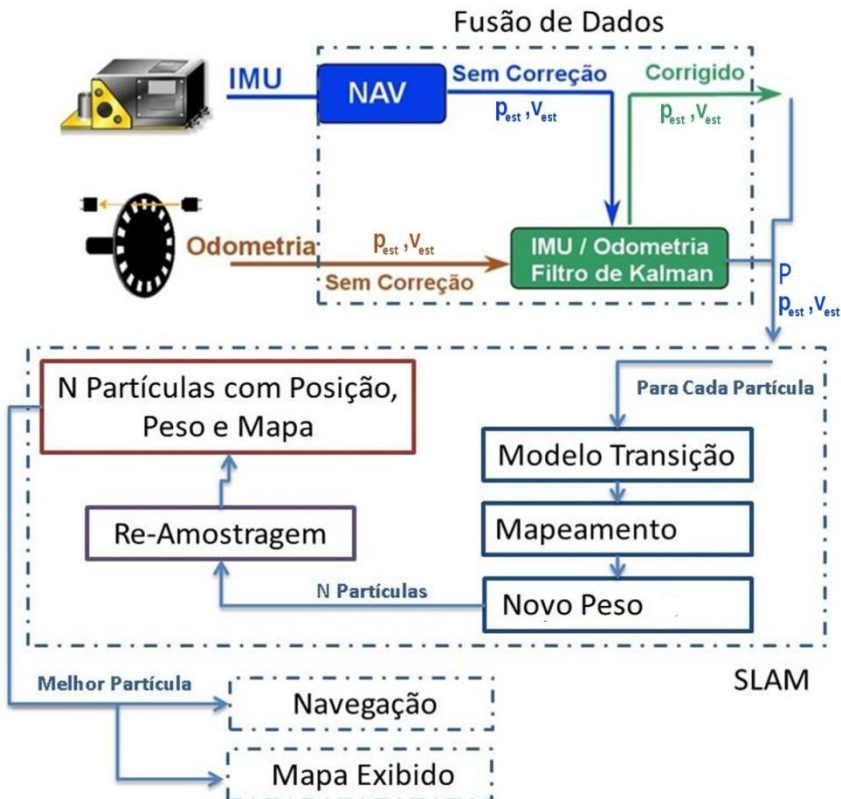


Figura 29 - Diagrama do algoritmo proposto.

Neste diagrama p_{est} representam posições e v_{est} as velocidades, ambos fornecidos pelos sensores e servem de entrada no filtro de Kalman (no caso, o EKF), que fornecerá os estados corrigidos para o SLAM. Paralelamente, foi desenvolvido um algoritmo que também utiliza os valores da matriz de covariância P , como um refinamento para o modelo de transição do robô no SLAM.

O algoritmo da fusão de dados (Quadro 6), como descrito no segundo capítulo, calcula os estados estimados e realiza uma aproximação, fornecendo um vetor de estados e uma matriz de covariância, que são então repassados ao algoritmo do SLAM a cada instante.

- Estado estimado a priori.

$$\tilde{\mathbf{X}}_k^- = f(\tilde{\mathbf{X}}_{k-1}^+, \mathbf{u}_{k-1}, 0) \quad (12)$$

- Cálculo dos resíduos da covariância a priori

$$\mathbf{P}_k^- = \mathbf{A}_{k-1} \cdot \mathbf{P}_{k-1}^+ \cdot \mathbf{A}_{k-1}^T + \mathbf{M}_{k-1} \cdot \mathbf{Q}_{k-1} \cdot \mathbf{M}_{k-1}^T \quad (16)$$

- Ganho de Kalman

$$\mathbf{K}_k = \mathbf{P}_k^- \cdot \mathbf{C}_k^T \cdot [\mathbf{C}_k \cdot \mathbf{P}_k^- \cdot \mathbf{C}_k^T + \mathbf{N}_k \cdot \mathbf{R}_k \cdot \mathbf{N}_k^T]^{-1} \quad (17)$$

- Estado estimado a posteriori.

$$\tilde{\mathbf{X}}_k^+ = \tilde{\mathbf{X}}_k^- + \mathbf{K}_k \cdot [\mathbf{Y}_k - h(\tilde{\mathbf{X}}_k^-, 0, k)] \quad (18)$$

- Covariância a posteriori.

$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C}_k] \cdot \mathbf{P}_k^- \quad (19)$$

Quadro 6 – Algoritmo da Fusão de Dados.

O SLAM utiliza estes valores para calcular a transição de cada partícula, amostrando de uma distribuição normal que utiliza o valor determinístico recebido sobre a posição do robô como média e uma covariância, que geralmente é fixa e baseada no erro ou ruído conhecido. Neste trabalho, além de ser utilizada a covariância fixa, é utilizada uma covariância baseada nos valores de P.

Após esta transição, cada partícula realiza o mapeamento dada a leitura dos sensores e suas posições estimadas. Foi utilizada uma técnica simples por redes Bayesianas para preencher o mapa de grade, sendo descrito com mais detalhes no Apêndice C.

Em seguida, cada partícula recebe um peso de acordo com algum método a escolha, que neste trabalho, foi o reconhecimento de marcadores. Ou seja, quando o robô entra no alcance do marcador, consegue saber aproximadamente a sua posição global, dando um peso maior para partículas que estejam com seus valores próximos a estes.

Ao final, é realizada uma reamostragem, na qual as partículas com maior peso possuem probabilidade maior de sobreviver e gerar novos descendentes.

Ainda a partícula com maior peso é utilizada como exibição na simulação naquele instante.

5.4 CONCLUSÕES

Neste capítulo foram revistos os principais pontos do SLAM e detalhado o Grid Based FastSLAM pela maior facilidade para implementação e visualização do mapa. Este não é o melhor SLAM disponível, pois possui um alto custo computacional, mas é robusto, resolvendo o problema de SLAM. Como o foco do trabalho é mostrar que a fusão de sensores possui um impacto positivo no resultado de um SLAM, simplificasse visualizado em uma técnica que não seja perfeita e que seja mais fácil de quantificar o desempenho.

Foi descrito também ao final, O diagrama do algoritmo completo e seu funcionamento foram descritos. Importante lembrar que dependendo do algoritmo utilizado, a fusão de sensores também pode ser utilizada para fornecer dados para outras partes do algoritmo (como mapeamento). Porém neste trabalho foi utilizado exclusivamente na etapa de transição dos estados.

A combinação entre estas duas técnicas já foi abordada em alguns trabalhos, principalmente focada em melhorias em seguimento de trajetórias. No SLAM há trabalhos bem atuais, como a utilização no carro autônomo da Google (Thrun, 2012).

No próximo capítulo serão descritos os experimentos realizados utilizando o algoritmo desenvolvido, visando principalmente avaliar quais são os parâmetros que mais afetam o algoritmo e comparar o seu desempenho ao algoritmo de SLAM sem a fusão.

6 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os experimentos realizados, utilizando todo conteúdo visto até o momento. O software escolhido foi o MATLAB pela facilidade em fazer operações com matrizes e a melhor apresentação da simulação.

O algoritmo descrito no capítulo anterior foi implementado em três etapas básicas para todos os casos simulados, sendo a primeira a realização de um trajeto “real” do robô para adquirir todas as informações necessárias para a simulação, já que são armazenados os estados e leituras dos sensores. Este trajeto não possui nenhum ruído e o robô apenas navega pelo ambiente pelos pontos desejados, armazenando informações.

Uma nova simulação é feita, desta vez com a trajetória predita através de modelos e de leituras das informações já armazenadas, porém com acréscimo de ruído.

Foram adotados dois casos base: a modelagem através da fusão de sensores (IMU e Odometria) e a modelagem somente através da odometria. A partir destas escolhas foram elaboradas algumas hipóteses e casos, que serão descritos nas próximas seções.

Para a odometria foi gerado um ruído com intensidade 10 vezes menor do que o gerado para a fusão. Essa diferença nas intensidades foi necessária para não haver uma tendência ao erro nos resultados, já que ao utilizar um ruído maior para a odometria, o trajeto se torna muito mais divergente.

Após esta etapa inicial de construção das trajetórias e com as mesmas armazenadas para cada instante, é realizada a simulação propriamente dita, utilizando o Grid Based Fast SLAM.

Para representar um ambiente “real”, foi feita uma matriz 100x100 representando um ambiente com paredes e com alguns obstáculos soltos, como mostrado na Figura 30.

Nesta mesma figura, as regiões escuras representam obstáculos, o ponto vermelho no canto inferior esquerdo representa o robô, a área em branco os espaços livres e os pontos azuis são os marcadores.

O mapa de referência foi criado baseado em um ambiente fechado, possuindo paredes e obstáculos soltos, sendo a rota do robô criada intencionalmente para que o sensor seja capaz de cobrir a maior parte do ambiente.

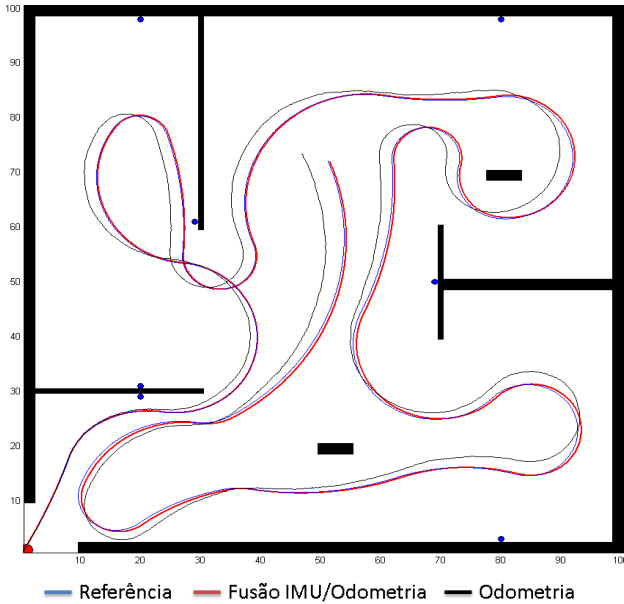


Figura 30 - Simulação das trajetórias no ambiente “real”.

6.1 SIMULAÇÃO

A simulação começa com o ambiente todo desconhecido (células com valor de 0,5), e o robô ao seguir as trajetórias faz o mapeamento e corrige a sua localização no mapa.

A todo instante, as partículas tendem a gerar descendentes diferentes e assim fazer um efeito de espalhamento, atribuindo um maior peso para as que estiverem mais corretas.

Conforme mostrada na Figura 31, a leitura do sensor não atravessa os obstáculos, e à medida que o robô for explorando o ambiente, o mapa vai sendo gerado em tempo real. Apesar dos marcadores estarem visíveis na imagem (pontos azuis), os mesmos são vistos somente pelo robô ao chegar ao seu alcance e caso não houver obstáculos entre eles. A tonalidade variando entre branco e preto, representa a possibilidade de haver ou não obstáculo (Figura 33), sendo 1 para representar a presença de obstáculo, 0 para espaço livre e 0,5 para espaço desconhecido.

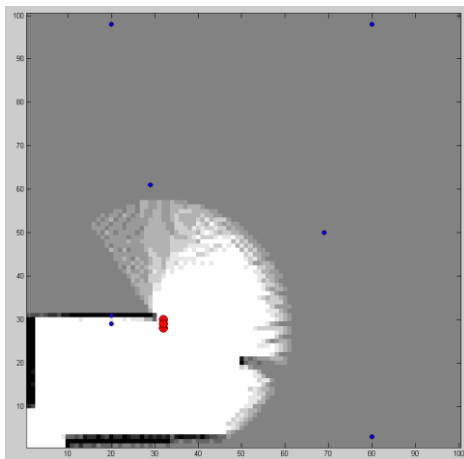


Figura 31 - Efeito do sensor nos obstáculos.

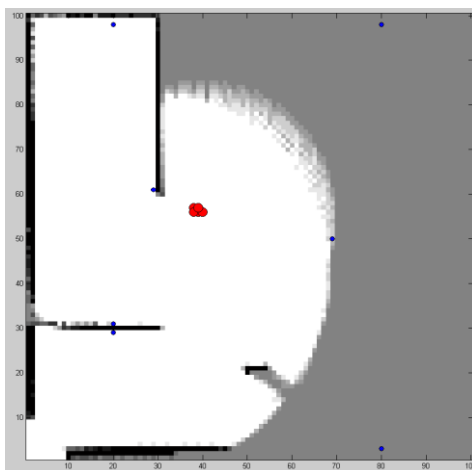


Figura 32 - Simulação do Grid Based Fast SLAM.

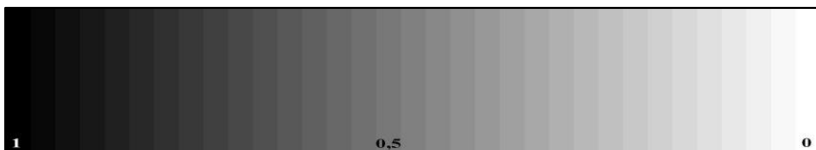


Figura 33 - Barra de tonalidade em relação aos valores.

6.2 CASOS SIMULADOS

Foram escolhidos alguns casos para averiguar o desempenho do algoritmo para as trajetórias da odometria e da fusão. Para cada caso foram feitas 10 simulações, avaliadas e representadas por gráficos indicando a média, mediana e variação dos resultados finais.

Para determinar como os parâmetros afetam o desempenho do algoritmo, foram feitas algumas hipóteses para poder comparar os resultados entre a simulação com a fusão e a que não inclui a fusão de sensores.

Primeiramente, espera-se que o número de partículas influencie diretamente o desempenho do algoritmo, procurando determinar se um número maior de partículas irá proporcionar um resultado melhor a um custo computacional maior.

Pretende-se assim, comparar resultados utilizando a odometria com um número maior de partículas e a simulação com fusão, utilizando um número menor de partículas, provando que o resultado é o mesmo ou ainda é melhor para a fusão, com menos partículas e redução do custo computacional.

Em seguida é avaliado o efeito da escolha da covariância no modelo de transição no SLAM, pelo fato de que se pretende incorporar valores da fusão diretamente no SLAM, é importante saber se há impacto significativo em uma variação grande ou pequena deste valor.

Assim, também se deseja saber qual é o impacto nos resultados para integração de valores da Matriz P da fusão de sensores no algoritmo do SLAM.

Para finalizar, pretende-se avaliar todos os métodos sob as mesmas variáveis, para ver se há um ganho real no desempenho do algoritmo.

Para atingir estes objetivos, foram considerados os 4 casos descritos a seguir.

- **Caso 1**

Avaliar somente o caso com odometria. Para isto, foram realizadas simulações para 20 e 40 partículas, alternando a covariância do modelo de transição das partículas entre os valores de 0,1, 0,18, 0,2 e 0,3.

Espera-se que os resultados com mais partículas tenham um resultado melhor por representarem uma maior diversidade de opções de trajetos.

Quanto à variação da covariância, se a mesma for muito pequena, as partículas não se espalharão o suficiente e o algoritmo trabalharia apenas mapeando uma trajetória fixa, enquanto que, caso a covariância for muito grande, os valores começam a divergir demais.

- **Caso 2**

Avaliar a fusão de dados entre IMU e odometria, e o impacto da covariância fixa. Se com o primeiro caso o número de partículas superior pode resultar em melhores resultados, aqui se procura avaliar que impacto existe em se limitar a covariância a um valor, mesmo que a trajetória se aproxime da real.

- **Caso 3**

Procura-se neste caso averiguar se é possível obter resultados melhores do que o segundo caso, realizando a interação entre a fusão de dados e o algoritmo de SLAM. Para isto são inseridos valores da matriz P da fusão de dados, diretamente no modelo de transição. Ainda, é limitado o valor da covariância para um número mínimo, para preservar a diversificação das partículas. Este valor foi estipulado experimentalmente como 0,18.

- **Caso 4**

Após todas estas simulações, é realizada uma simulação final mantendo o mesmo padrão de valores para os casos da odometria, fusão fixa, fusão adaptativa e para o caso de uma trajetória perfeita.

6.3 AVALIAÇÃO

Para avaliação da eficácia do mapa, foram utilizados dois parâmetros:

- **Total de Acertos**

O total de acertos considera as células que são mapeadas corretamente. A cada espaço livre corretamente mapeado como espaço livre, é somado um ponto ao total. E quando a célula for um obstáculo e

mapeado como obstáculo ou “valor desconhecido”, também é somado um ponto ao total. Este parâmetro tem o valor máximo de 10000, pois a matriz utilizada possui tamanho de 100 linhas por 100 colunas.

- **Total de Erros**

Para o segundo critério foi criada uma faixa de segurança em volta dos obstáculos onde não é dada nota a um mapeamento inadequado. Esta consideração foi feita conforme os conceitos de navegação discutidos nos capítulos iniciais. Atribui-se que ao gerar uma trajetória, podemos manter uma distância de segurança dos obstáculos devido a erros ou imprevistos. Portanto se considerar que embora haja um mapeamento de uma célula livre dentro de um obstáculo, o robô não irá colidir em um objeto não previsto, devido a faixa de segurança utilizada na navegação.

O mesmo acontece ao mapear um obstáculo na área livre a uma célula de distância, pois não estão sendo abordados os casos de navegação em lugares muito estreitos (que fecharia a passagem do robô).

Assim, esse segundo critério avalia erros na região azul indicada na Figura 34, que consiste em mapeamento de obstáculos deslocados em mais de uma célula. A cada célula de obstáculo mapeado dentro da zona azul, soma um ponto ao total, sendo que o melhor resultado é o de 0 pontos.

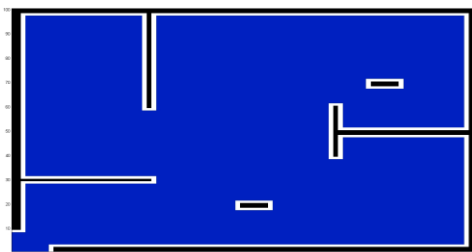


Figura 34 - Mapa com região de avaliação em azul.

Além destes parâmetros, optou-se por representar graficamente os resultados, mas devido a quantidade de simulações (10) para cada caso e como sempre se obterão resultados diferentes por se tratar de um algoritmo probabilístico, foi incluído no gráfico a variação dos resultados e apresentadas a média e a mediana de cada conjunto.

A mediana representa o valor no qual 50% das amostras estão acima do seu valor e 50% estão abaixo. É uma medida útil para quando se quer eliminar amostras que estão muito distantes da média.

6.4 RESULTADOS

Os gráficos dos resultados obtidos para os casos discutidos nas seções anteriores e suas respectivas análises e conclusões são apresentados nas próximas subseções.

Os mapas gerados pelas simulações foram os mais diversos possíveis. Para demonstrar o tipo de variação encontrada é exposto o mapa original na Figura 35, e os mapas gerados pela odometria, fusão fixa e fusão variável, com 20 partículas e covariância de 0,18. Como são realizadas 10 simulações para cada caso, ficaria inviável colocar o mapa de cada situação de cada caso.

Por esta razão, foram criados gráficos que dispõem do valor máximo, mínimo, média e mediana para cada caso.

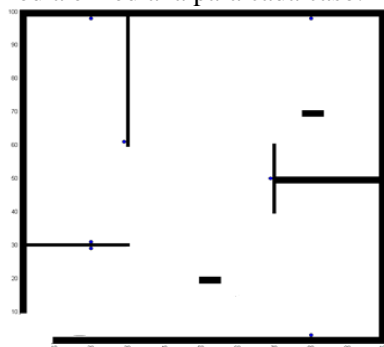


Figura 35 - Ambiente a ser explorado pelo robô.

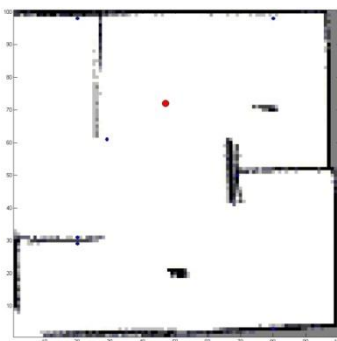


Figura 36 - Exemplo de mapa gerado pela odometria somente.

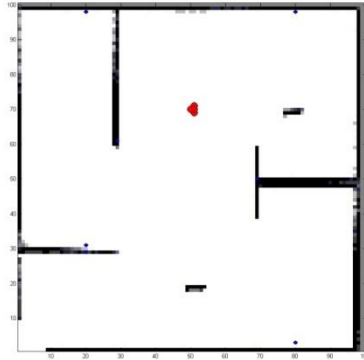


Figura 37 - Exemplo de mapa gerado pela fusão fixa.

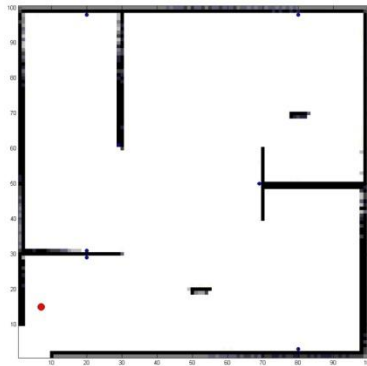


Figura 38 - Exemplo de mapa gerado pela fusão adaptativa.

6.4.1 Caso 1

Foi possível observar neste caso, utilizando somente a odometria, que o maior número de partículas (os três pontos da direita, Figura 39 e Figura 40) possibilitou o algoritmo obter médias melhores nos resultados do que com menos partículas. Isto já era esperado devido ao fato de haver uma maior diversificação de opções para o algoritmo convergir para um resultado melhor. Isto também ocasiona uma menor variação nos resultados com número maior de partículas, pois os resultados conseguem convergir mais facilmente.

São obtidos resultados melhores com uma covariância de valor 0,2 do que a de 0,3 ou 0,1; isto se explica o fato de que uma covariância muito pequena não gerar partículas diferentes o suficiente para corrigir o

trajeto da odometria e um número grande na covariância causa o efeito contrário, as partículas geradas estão muito afastadas da trajetória real.

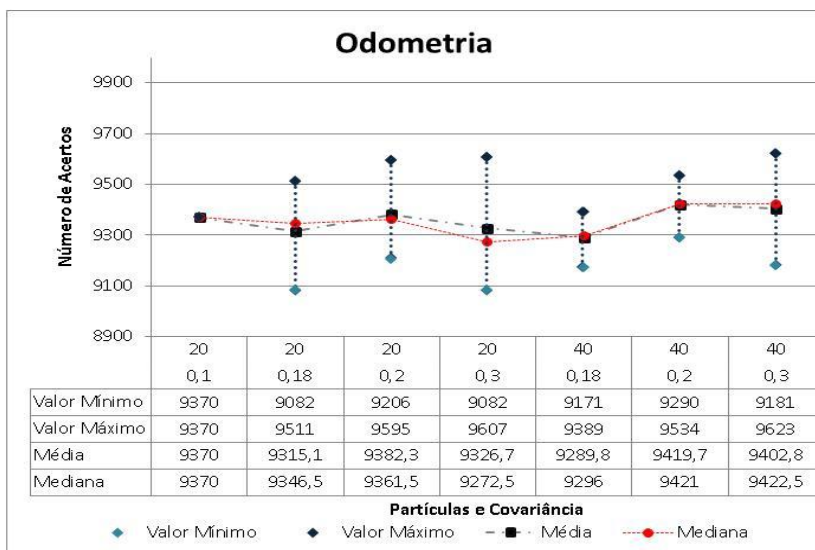


Figura 39- Número de acertos na odometria.

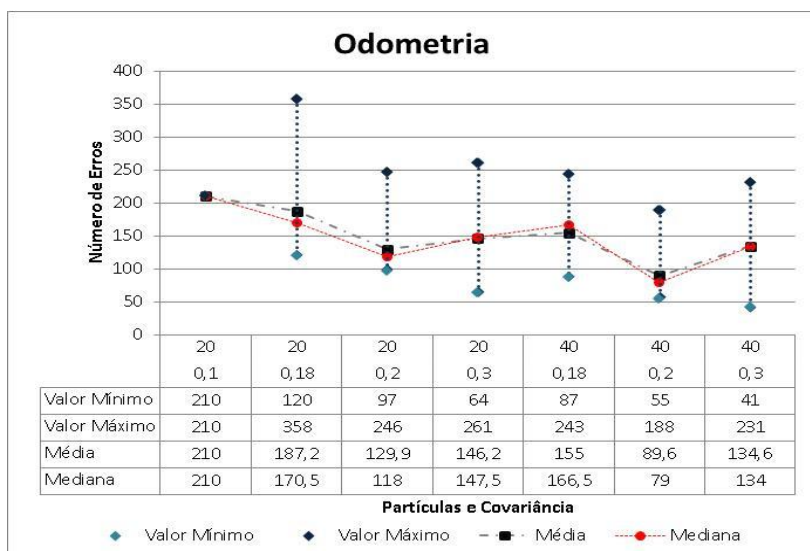


Figura 40 - Número de erros na odometria.

6.4.2 Caso 2

Como a trajetória está próxima ao ideal com a realização da fusão de sensores, se houver grande variância no modelo de transição do robô, as partículas geradas se distanciarão do caminho correto, ocasionando erros maiores no mapeamento.

Notou-se que com a variância de 0,15 ou menor, a diversidade das partículas é prejudicada, neste caso houve um resultado melhor, pois o trajeto construído pela fusão está próximo do real (o que não acontece no primeiro caso).

Percebeu-se que com um valor de 0,18 é possível garantir uma diversificação mínima das partículas, sendo este valor adotado como padrão para as próximas simulações.

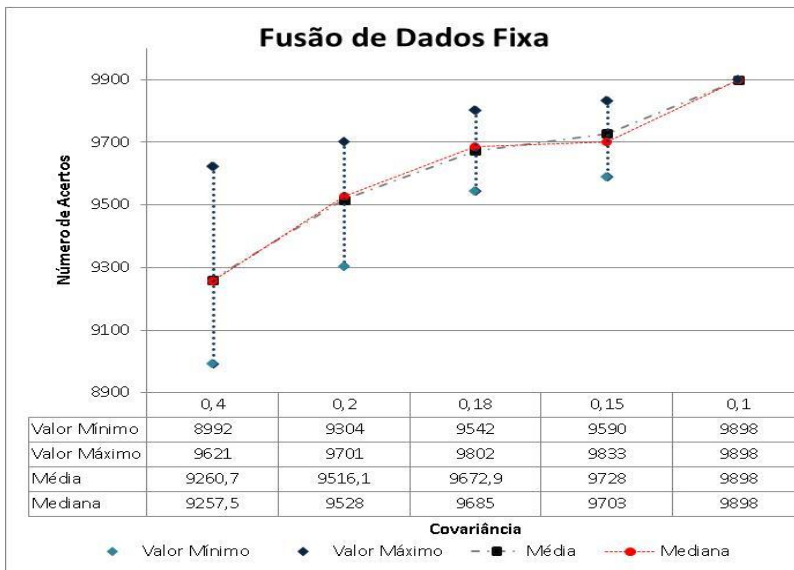


Figura 41 - Número de acertos na fusão de dados fixa.

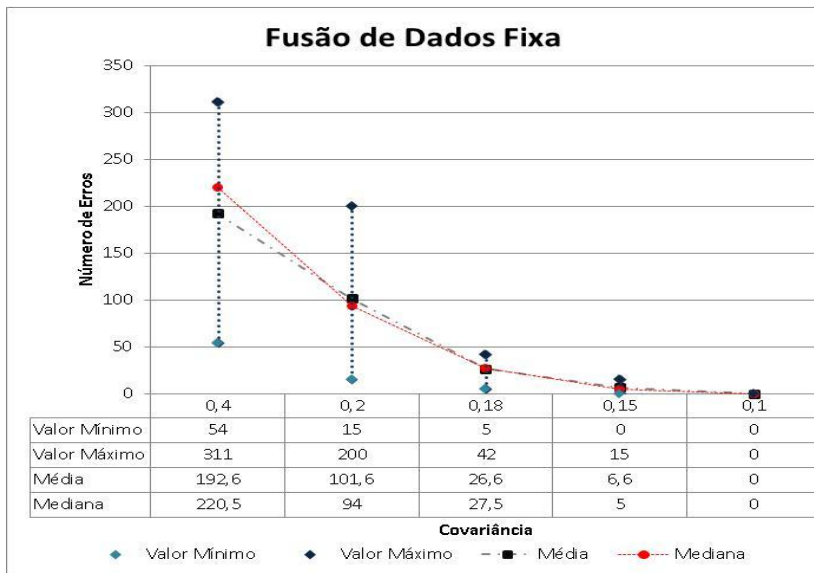


Figura 42 – Número de erros na fusão de dados fixa.

6.4.3 Caso 3

Os resultados obtidos com a fusão adaptativa não foram satisfatórios em um primeiro momento. Isto porque, quando os valores de P eram muito reduzidos, não acontecia a diversificação das partículas no SLAM, prejudicando o seu desempenho.

Para resolver este problema, foi estabelecido um valor mínimo para a covariância, sendo que qualquer valor de P que seja menor que 0,18, é substituído por este valor.

Assim, neste caso alterna-se o número de partículas, já que este se tornou o único parâmetro modificável.

Foi possível perceber como anteriormente, que com um número reduzido de partículas diferentes a cada interação, a diversificação de seus valores durante a simulação é limitada. Ou seja, quanto maior o número de partículas, mais opções de poses são geradas, possibilitando a correção do caminho através da sobrevivência das partículas com melhores valores.

Portanto, haverá uma grande variação no resultado final entre as simulações com poucas partículas, cujo valor, dependerá diretamente da probabilidade de ser amostrado um caminho correto.

Outra constatação está na média dos resultados da fusão adaptativa para 20 partículas, que é superior ao do caso fixo, que é melhor exposto no próximo caso.

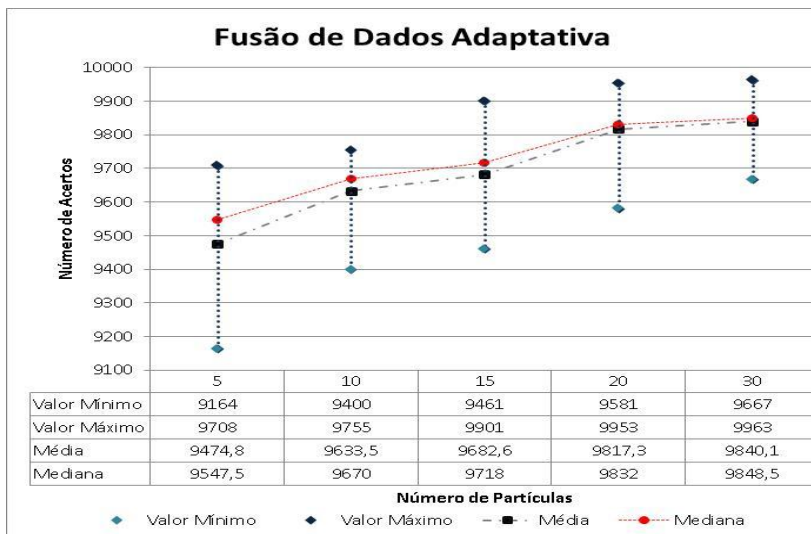


Figura 43 - Número de acertos na fusão de dados adaptativa.

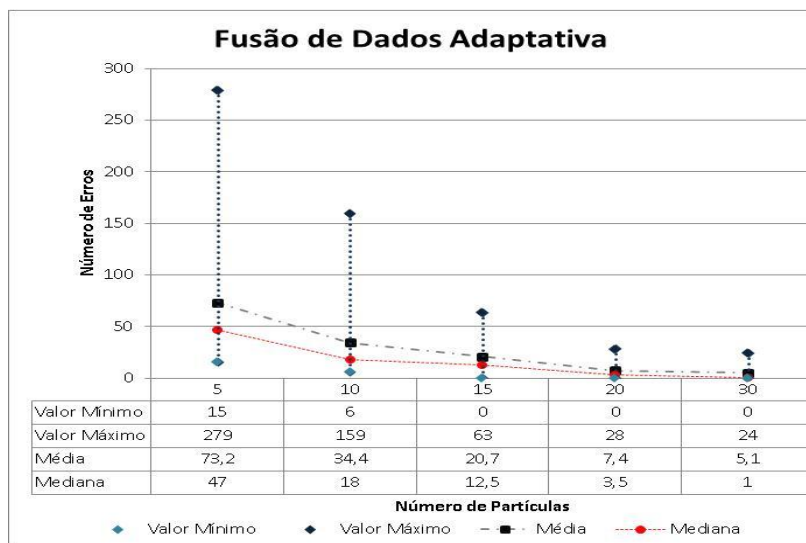


Figura 44 – Número de erros na fusão de dados adaptativa.

6.4.4 Caso 4

Foram feitas as simulações de todos os métodos para o caso de 20 partículas e variância fixa ou limitada a 0,18. Como pode ser observado o modelo que considera apenas odometria tem um desempenho bem inferior aos demais.

A fusão com variância flexível possui uma taxa de erro semelhante ao de valor fixo, mas como é adaptável, consegue atingir valores mais próximos do ideal. Isto se deve ao fato de que as variações na fusão dão pistas ao algoritmo de SLAM sobre o quão errado estão os valores, possibilitando uma melhor correção.

O trajeto real também foi simulado com variância muito pequena de 0,1 para indicar e ser referência de qual a melhor possibilidade que existe para este conjunto de simulações.

Foi simulado também com variância fixa de 0,18, indicando que mesmo com trajeto ideal, o resultado nunca será perfeito se a covariância for fixada com um valor mais alto e que apenas com valores pequenos de covariância é possível chegar ao resultado ideal (ao custo da não diversificação das partículas e mau desempenho para trajetórias ruins).

Isto se explica pelo fato de que o trajeto real está totalmente certo e qualquer amostragem com variância, amostrará um caminho errado.

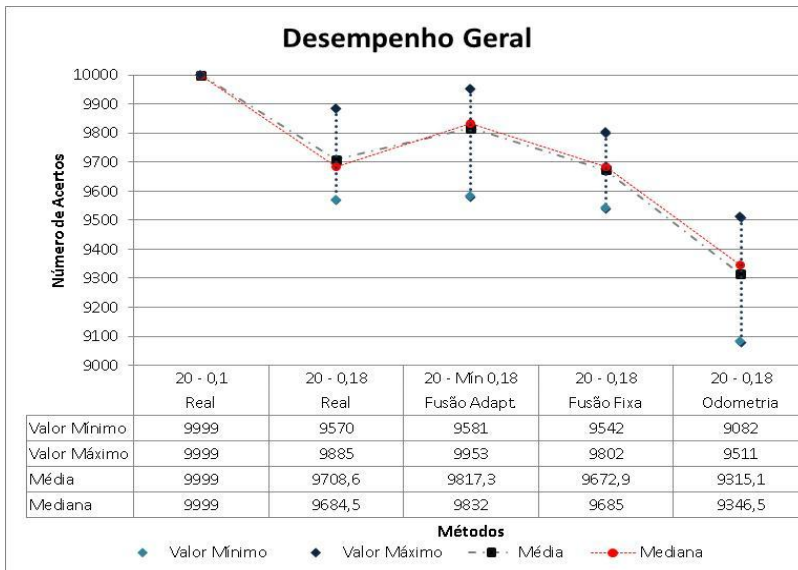


Figura 45 - Número de acertos entre os métodos.

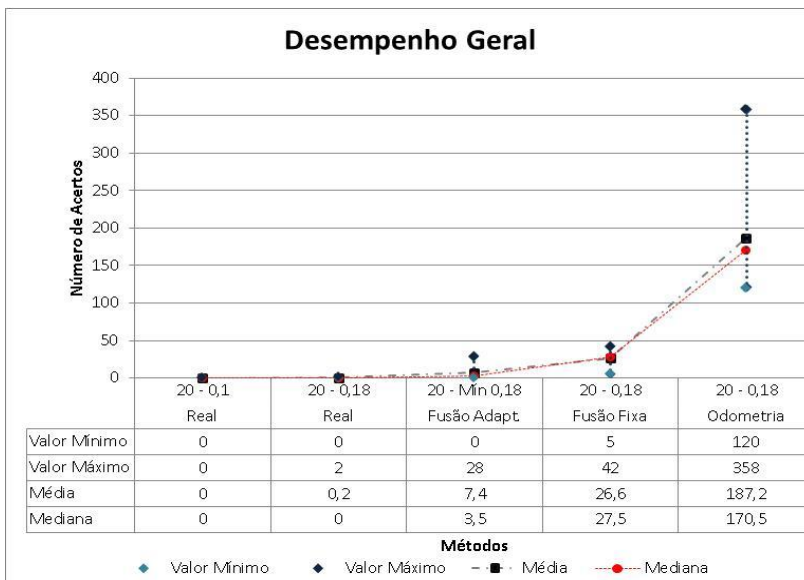


Figura 46 - Número de erros na fusão de dados adaptativa.

7 CONCLUSÃO

A proposta deste trabalho foi a de mostrar que algoritmos de SLAM para robótica móvel terrestre podem ter um melhor desempenho se combinados com uma técnica de fusão de sensores, principalmente se houver uma maior integração entre o sistema de fusão de dados e o algoritmo de SLAM, reduzindo também, o custo computacional.

Essa melhoria pode ter um papel fundamental, em especial em robôs de baixo custo, onde nem sempre a precisão dos sensores é garantida e a capacidade de processamento é limitada.

Neste trabalho foram revistas as etapas de planejamento de uma navegação completa, revisando e escolhendo o robô, os sensores, a representação do robô, a sua navegação local e o SLAM propriamente dito.

Foi construído um SLAM baseado no algoritmo de Thrun (2006), incorporando a técnica de fusão de dados na função de transição de estados do filtro de partículas.

Com o algoritmo foram realizados casos para avaliar a influência da alteração de parâmetros nos mapas gerados, sendo possível verificar melhorias em todos os algoritmos quando há aumento do número de partículas.

Outro aspecto importante foi que a escolha da covariância certa tem papel fundamental neste algoritmo, sendo que se o valor for muito pequeno as partículas não se diversificam o suficiente e se for muito grande o valor, ocorre o inverso, as partículas divergem muito e causam um mau desempenho.

A principal contribuição deste trabalho foi a proposta de uma maior integração entre os algoritmos, utilizando a matriz de covariância da fusão de dados. Isto possibilita o SLAM ter uma melhor suposição sobre o erro da trajetória e não realizar uma transição maior do que a necessária.

Segundo Thrun (2012), o FastSLAM é a base do algoritmo utilizado no carro automatizado da Google, sendo alterado apenas como são feitas as funções de mapeamento e de atribuição de pesos, sendo que o autor também utiliza uma fusão, mas entre odometria, IMU e GPS. Apesar de utilizar a fusão, o autor não descreve como é realizada e não há publicações do mesmo sobre este assunto em especial, sendo que foi considerado que a fusão realizada é a padrão.

Ao final deste trabalho, através das primeiras simulações foi possível constatar uma boa diferença entre os mapas utilizando as

trajetórias geradas pela odometria e a trajetória utilizando fusão de sensores. O impacto no SLAM é considerável, havendo necessidade de um número grande de partículas para tentar compensar e minimizar o efeito de uma má predição.

A fusão de sensores traz uma grande vantagem, pois além de melhorar o resultado, podendo ser implementada em baixo nível, ou seja, próximo ao sensor. Isso possibilita uma redução considerável de processamento, pois o controle do robô se detém apenas com a questão de SLAM.

Com este trabalho espera-se contribuir para que futuros projetos de robôs móveis de baixo custo sempre incluam no seu design, o projeto de uma pequena unidade de fusão de sensores e que principalmente, considere a inclusão de um pequeno sensor IMU.

Como sugestão para futuros trabalhos, recomenda-se a aplicação de fusão de dados em outras técnicas de SLAM, como o filtro de partículas Rao-Blackwellizado.

Outra recomendação é a alteração das funções deste algoritmo para torna-lo mais eficiente, como por exemplo, a substituição de marcadores visuais por reconhecimento de padrões (como paredes).

A re-implementação e aplicação deste algoritmos em softwares como o Player/Stage para controle de robôs reais ou simulações bem próximas da realidade também podem ser de grande utilidade, apesar da alta complexidade de se implementar o algoritmo sem algumas bibliotecas básicas (como o de operações com matrizes).

REFERÊNCIAS

ABDEL-HAMID, Walid. **Accuracy Enhancement of Integrated MEMS-IMU/GPS Systems for Land Vehicular Navigation Applications**. UCGE Reports 20207, Universidade de Calgary, 2005.

AIUBE, F. A. L. e BAIDYA, T. K. N. **Modelagem dos Preços Futuros de Commodities: Abordagem pelo Filtro de Partículas**. Tese de doutorado, PUC-Rio, 2005.

AMIGONI, F. e A. GALLO. **A Multi-Objective Exploration Strategy for Mobile Robots**. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 3861–3866.

BIANCHI, Reinaldo. **Introdução à Robótica**, 2010. Slides de Aula. Faculdade de Engenharia Industrial – FEI, disponível em: <<http://fei.edu.br/~rbianchi/robotica/>>, acesso em: maio de 2012.

BORENSTEIN, H., EVERETT, R., FENG, L., e WEH, D. **Mobile Robot Positioning & Sensors and Techniques**, in *J. Robot. Syst.*, vol. 14, no. 4, pp. 231–249, 1997.

BORENSTEIN, J., EVERETT, H. R., FENG, L. **Where am I? Sensors and Methods for Mobile Robot Positioning**. 1996. Disponível em: <<http://www-personal.umich.edu/~johannb/position.htm>>, acesso em: maio de 2012.

BORGES, L. P.; DORES, R. C. **Automação predial sem fio utilizando bacnet/zigbee com foco em economia de energia**. Trabalho de conclusão de curso, graduação em Engenharia de Controle e Automação, UNB, Brasília, 2010.

BURGARD, W., CREMERS, A. B., FOX, D., HAHNEL, D., LAKEMEYER, G., SCHULZ, D., STEINER, W., e THRUN, S. **Experiences with an interactive museum tour-guide robot**. *AIJ*, 114(1-2), 3-55, 1990.

BURGARD, W.; MOORS, M. e STACHNISS, C. **Coordinated Multi-Robot Exploration**. *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

CHOSSET, Howie; LYNCH, Kevin; THRUN, Sebastian; et al. **Principles of Robot Motion: Theory, Algorithms, and Implementations**, 2005. Cap. 3, 4, 5 e 6.

DEAN, T., BASYE, K., CHEKALUK, R., e HYUN, S. **Coping with uncertainty in a control system for navigation and exploration**. In *AAAI-90*, Vol. 2, pp. 1010-1015, 1990.

DISSANAYAKE, G.; WILLIAMS, S. B.; DURRANT-WHYTE, H. e BAILEY, T. **Map Management for Efficient Simultaneous Localization and Mapping (SLAM)**. *Journal of Autonomous Robots*, 2002, Springer Netherlands, Vol. 12, Issue 3, p.267-286.

DOUCET, Arnaud; FREITAS, Nando de; MURPHY, Kevin; RUSSEL, Stuart. **A Simple Tutorial on RBPFs for DBNs**, 2000. Tutorial fornecido com o artigo na UAI2000.

DOUXCHAMPS, Damien. **Small List of IMU**, maio de 2012, disponível em <<http://damien.douxchamps.net/research/imu/>>.

FIGUEIREDO, L. J.; GAFANIZ, A. R.; LOPES, G. S. e PEREIRA, R. **Aplicações de Acelerómetros**. Instrumentação e Aquisição de Sinais, monografia, Lisboa, Portugal, 2007.

GRISSETTI, Giorgio; et al. **A Comparison of SLAM Algorithms Based on a Graph of Relations**. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Outubro de 2009.

GRISSETTI, Giorgio; STACHNISS, Cyrill, e BURGARD, Wolfram. **Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filter**. *IEEE Transactions on Robotics*, Vol. 23, No. 1, fevereiro de 2007.

HALL, D. L. e LLINAS, J. **HandBook of MultiSensor DataFusion**. 2001, CRC Press, New York – USA.

HOLZ, Dirk; BASILICO, Nicola; AMIGONI, Francesco; Behnke, Sven. **A comparative evaluation of exploration strategies and heuristics to improve them**. In *proceedings of the European conference on mobile robotics (ECMR)*, Outubro de 2011, Suécia.

HOLZ, Dirk; BASILICO, Nicola; AMIGONI, Francesco; BEHNKE, Sven. **Evaluating the efficiency of frontier-based exploration strategies**. In *proceeding of joint 41th International Symposium on Robotics and 6th German Conference on robotics*, Munich, Junho de 2010 (ISR/ROBOTIK 2010).

KLEINBAUER, R. **“Kalman Filtering Implementation with Matlab”**. 2004, Universidade de Stuttgart, Institute of Geodesy.

LEÃO, Gustavo. **Probabilidade e Estatística**. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, apresentação de Slides, 2010. Disponível em: < <http://www.dca.ufrn.br/~guga/downloads/pcs/aulas/Aula4.pdf> >, acesso em março de 2012.

LINDSTEN, Frederik. **Rao-Blackwellised Particle Methods for Inference and Identification**. 2011, Tese de Doutorado. Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping – Sweden.

LUO, Ren C.; KAY, Michael G. **Multisensor Integration and Fusion for Intelligent Machines and Systems**. Norwood: Ablex Publishing Corporation, 1995, 688 p.

MENEGALDO, L. L. et al. **SIRUS: A mobile robot for Floating Production Storage and Offloading (FPSO) ship hull inspection**, 2008, IME – Brasil, IEEE.

MICRO STRAIN. **Manual 3DM-GX2**, 2011. Disponível em: <<http://www.microstrain.com/inertial/3DM-GX2>>, acessado em: dezembro de 2011.

MITCHELL, H. B. **Multi-Sensor Data Fusion: An Introduction**. 2007, Editora Springer.

MONTEMERLO, M., THRUN, S., e WEGBREIT, B. **FastSLAM: a factored solution to simultaneous localization and mapping problem**. 2002, In *AAAI-02*.

MORAVEC, H. P. e ELFES, A. **High Resolution maps from wide angle sonar**. In *ICRA-85*, pp. 116-121, 1985.

MOUTARLIER, P.e CHANTILA, R. **Stochastic multisensory data fusion for mobile robot localization and environment modeling.** In ISRR-89, 1989.

MURPHY, K. e RUSSELL, S. J. **Rao-blackwellised particle filtering for dynamics Bayesian networks.** In Doucet, A., de Freitas, N. and Gordon, N. J. (Eds.), Sequential Monte Carlo Methods in Practice. Springer-Verlag, 2001.

MUSTIÈRE, Frédéric; BOLIC, Miodrag; BOUCHARD, Martin. **Rao Blackwellized Particle Filters: examples of applications.** 2006, in Canadian Conference on Electrical and Computer Engineering, CCECE '06.

NASSER, Sameh. **Improving the Inertial Navigation System (INS) Error Model for INS and INS/DGPS Applications.** UCGE Reports Number 20183, Tese de Doutorado, Universidade de Calgary, Canadá, 2003.

NORVIG, Peter e RUSSEL, Stuart. **Artificial Intelligence: A Modern Approach.** Prentice Hall, 2ª Edição - 2003 e 3ª Edição - 2010.

NORVIG, Peter e THRUN, Sebastian. **Introduction to Artificial Intelligence.** 2011, Vídeo-curso online. Disponível em: <<https://www.ai-class.com/>>, acesso em: dezembro 2011.

NORVIG, Peter e THRUN, Sebastian. **Programing a Robotic Car.** 2012, Vídeo-curso online. Disponível em: <<https://www.udacity.com/>>, acesso em: maio de 2012.

ODAKURA, Valguima. **Tópicos em Robótica Móvel.** 2010, Slides de Aula. Universidade Federal da Grande Dourados – UFGD.

RAOL, J. R. **Multi-Sensor Data Fusion with Matlab.** 2009, CRC Press, National Aerospace Lab e MSRT, Bangalore, Índia, 568 p.

RIBEIRO, M. I. **Localização em Robótica Móvel: Odometria.** Apresentação de Slides, Instituto de Sistemas e Robótica. Lisboa, Portugal, 1999.

RODRIGO, Jaderson. **Navegação Autônoma de Robôs Móveis**. Curso de RMI, apresentação de slides. Disponível em: <http://wiki.icmc.usp.br/images/7/76/NavegacaoAutonoma_Robotica_RAFR.pdf>, acesso em: maio de 2012.

RUSSEL, Stuart e MURPHY, Kevin. **Rao-Blackwellized Particle Filter for Dynamic Bayesian Networks**, 2000. DOI 10.1.1.7.6287.

SALUSTIANO, R. E. e REIS, C. **A Aplicação de Técnicas de Fusão de Sensores no Monitoramento de Ambientes**. Universidade de Campinas, dissertação de mestrado, 2006.

SIEGWART, R.; NOURBAKHS, ILLAH, R. **Introduction to Autonomous Mobile Robots**. 2004, The MIT Press, Cambridge, England.

SIM, R. e ROY, N. **Global A-Optimal Robot Exploration in SLAM**. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Abril 2005, pp. 661–666.

SMITH, R. C. e CHEESEMAN, P. **On the representation and estimation of spatial uncertainty**, 1986, *Int J. Robotics*, 5(4), 56-58.

STACHNISS, Cyril; Grisetti, Giorgio; Wolfram, Burgard. **Recovering Particle Diversity in a Rao-Blackwellized Particle Filter for SLAM After Actively Closing Loops**. *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, p. 667-672, Barcelona, Espanha, 2005.

STACHNISS, Cyril; **Building Accurate Maps Using RBPF**. Universidade de Freiburg – Alemanha. Apresentação de Slides. Disponível em: <<http://webdiis.unizar.es/~neira/5007439/stachniss-rbpfmapping.pdf>>, acesso em: maio 2012.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics**. The MIT Press, Cambridge, England, 2006. 647 p.

THRUN, S. **Stanley, the robot that won the DARPA Challenge**. *J. Field Robotics*, 23(9), 661-692, 2006.

VECTORNAV. **VN-100 IMU/AHRS**. 2012. Disponível em: <<http://www.vectornav.com>>, acesso em: maio 2012.

WELCH, G. e Bishop, G. **An Introduction to the Kalman Filter**. 2006, Department of Computer Science, University of North Carolina.

WU, Ling; PUIG, Domenec; GARCIA, Miguel Angel. **Active path planning strategy for coordinated multi-robot exploration**. 2010. Disponível em: <<http://www.jopha.net/waf/index.php/waf/waf10/paper/viewFile/58/68>>, acesso em: maio de 2012.

YAMAUCHI, Brian. **Frontier-based exploration using multiples robots**. 1998, *In Proceedings of the Second International Conference on Autonomous Agents*.

YAMAUCHI, B. **A Frontier Based Approach for Autonomous Exploration**. *In Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997, pp. 146–151.

YAMAUCHI, Brian. **Frontier-Based Exploration**. Disponível em: <<http://www.robotfrontier.com/frontier/index.html>>, acesso em março de 2012.

APÊNDICE A – Probabilidade

Para construir o algoritmo do SLAM e da fusão de sensores são utilizados conceitos básicos de redes bayesianas e por consequência de probabilidade.

De uma maneira bem básica, entende-se por probabilidade a relação $P = x / n$, onde x é um número conhecido de eventos dentre n eventos quaisquer, com valor igual ou inferior a n . A probabilidade também pode ser definida como a medida de um conjunto de eventos que satisfaz três axiomas descritos abaixo.

- A medição de cada evento está entre 0 e 1. Ou seja, $0 \leq P(X = x_i) \leq 1$, onde X é uma variável aleatória representando um evento e x_i são os possíveis valores de X . Em geral, variáveis aleatórias são denotadas com letra maiúsculas e seus valores com letras minúsculas;
- O valor de todo o conjunto é 1. Ou seja, $\sum_{i=1}^n P(X = x_i) = 1$. Isso implica, por exemplo, que se a chance de um evento $P(A)$ acontecer for 0,70, a chance do evento não acontecer será $P(\neg A) = 1 - 0,70 = 0,30$.
- A probabilidade da união de eventos não relacionados é a soma da probabilidade dos eventos individuais. Ou seja, $P(X = x_1 | X = x_2) = P(X = x_1) + P(X = x_2)$.

Os eventos podem ser classificados em vários tipos:

- Eventos equiprováveis: os que possuem a mesma probabilidade de ocorrerem (ao exemplo da moeda lançada para o alto);
- Eventos condicionais: são aqueles em que a chance do segundo evento ocorrer depende da ocorrência do primeiro evento. É representado como $P(B|A)$, ou seja, probabilidade do evento B acontecer dado evento A.
- Eventos independentes: são aqueles que ocorrem de um modo totalmente independente, ou seja, $P(B|A) = P(B)$;
- Eventos mutuamente exclusivos: quando a realização de um exclui a realização do outro.

APÊNDICE B – Distribuição Gaussiana

A distribuição de Gauss ou Gaussiana é uma das mais importantes distribuições da estatística, conhecida também como distribuição normal. É caracterizada por dois parâmetros, a média μ e uma medida de dispersão que pode ser o desvio-padrão σ ou a variância σ^2 .

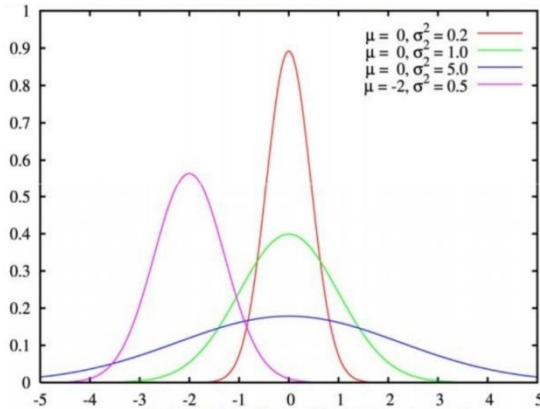


Figura 47 - Distribuição Gaussiana.

Fonte: Leão (2010)

A área sob a curva será sempre 1, sendo que a média dita o centro da distribuição e a variância o espalhamento da curva. A notação para variável X governada por uma distribuição Gaussiana é $X \sim N(\mu, \sigma^2)$.

A função densidade de probabilidade da variável aleatória com distribuição normal é dada por (34).

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (34)$$

A distribuição Gaussiana é muito utilizada na robótica, pois além de ser peça chave do filtro de Kalman utilizado na fusão de sensores, a mesma pode representar o movimento do robô com sua incerteza (ruídos), a leitura de um sensor com os ruídos ou qualquer outra forma de dados que estejam na forma de um valor aproximado e alguma incerteza (média e variância).

APÊNDICE C – Redes Bayesianas

Quando se trabalha com modelos probabilísticos é inevitável que se construa uma rede Bayesiana. As redes bayesianas são modelos gráficos para raciocínio baseado na incerteza, onde os nós representam as variáveis de um domínio e os arcos representam as conexões ou dependências entre elas. Um exemplo bem simplificado de rede Bayesiana aplicada à automação é mostrado Figura 48.

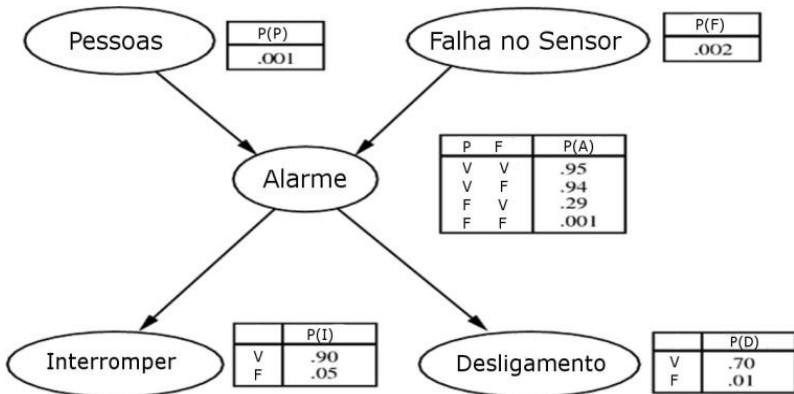


Figura 48 - Exemplo de Rede Bayesiana.

Fonte: Adaptado de Thrun (2011)

Neste exemplo, é descrita uma situação de uma área restrita que possui um alto grau de periculosidade. A rede bayesiana descreve a probabilidade de eventos ocorrerem. É possível notar na parte superior do grafo, a probabilidade de dois eventos simples e independentes ocorrerem, tendo o valor de 0,1% para o evento de detecção de uma pessoa por um sensor ($P(P)$) e o valor de 0,2% para o evento de alguma falha acionar o sensor acidentalmente ($P(F)$). Ou seja, está sendo considerado que a segurança é tão bem montada, que há chances maiores de o sensor apresentar falha do que uma pessoa entrar nesta área.

Ao centro do exemplo há uma tabela de verdade, descrevendo qual a probabilidade do evento “Alarme” ($P(A)$) ocorrer dados os valores dos eventos anteriores (Presença de Pessoas e/ou Falha no Sensor).

Se o alarme ocorrer, há chances diferentes de haver um desligamento total do processo ou uma interrupção momentânea das máquinas da área em questão. É possível notar que todos os eventos

estão inter-relacionados, ou seja, se um evento inicial ocorrer é possível prever com certa probabilidade o que irá acontecer.

Para realizar estes cálculos são necessárias duas equações fundamentais, a equação da regra de Bayes (35), que calcula as chances de um evento A acontecer dado evento B, e a equação da probabilidade total (36), que calcula a probabilidade total, considerando todos os acontecimentos que levam a um evento B.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (35)$$

$$P(B) = \sum_a P(B|A = a)P(A = a) \quad (36)$$

O cálculo de valores através de uma rede de Bayes é apresentado através de um exemplo, no Quadro 7.

Há uma forma de simplificar os cálculos, utilizando um normalizador n, conforme mostrado nas equações (37), (38), (39) e (40), e exemplificado no Quadro 8.

$$P'(A|B) = P(B|A) \cdot P(A) \quad (37)$$

$$P'(\neg A|B) = P(B|\neg A) \cdot P(\neg A) \quad (38)$$

$$n = (P'(A|B) + P'(\neg A|B))^{-1} \quad (39)$$

$$P(A|B) = n \cdot P'(A|B) \quad (40)$$

Teste de Obstáculo 1

Considerando que a probabilidade de haver um obstáculo em uma área restrita é de $P(O) = 0,01$ e é possível determinar a presença ou não de obstáculos através de um sensor, que possui $P(+|O) = 0,9$, ou seja, confiabilidade de indicar corretamente (e positivamente) a presença de obstáculos de 90% e que tem a chance de um falso positivo acontecer de $P(+|-O) = 0,2$ (20%). Calcule a chance de haver um obstáculo, dado um resultado positivo no sensor $P(O|+)$.

$$P(O) = 0,01 \quad \text{Implica em: } P(-O) = 0,99$$

$$P(+|O) = 0,9 \quad \text{Implica em: } P(-|O) = 0,1$$

$$P(+|-O) = 0,2 \quad \text{Implica em: } P(-|-O) = 0,8$$

Probabilidades Conjuntas:

$$P(+, O) = P(+|O) \cdot P(O) = 0,9 \cdot 0,01 = 0,009$$

$$P(-, O) = P(-|O) \cdot P(O) = 0,1 \cdot 0,01 = 0,001$$

$$P(+, -O) = P(+|-O) \cdot P(-O) = 0,2 \cdot 0,99 = 0,198$$

$$P(-, -O) = P(-|-O) \cdot P(-O) = 0,8 \cdot 0,99 = 0,792$$

A probabilidade de um sinal positivo acontecer pode ter origem de uma medição correta e de um falso positivo (O e $-O$), portanto:

$$P(+) = \sum_a P(+|O = a)P(O = a) = 0,009 + 0,198 = 0,207$$

$$P(O|+) = \frac{P(+|O) \cdot P(O)}{P(+)} = \frac{0,009}{0,207} = 0,043 = 4,3\%$$

Ou seja, a probabilidade de que haja um objeto na área restrita é de apenas 4,3%.

Quadro 7 – Exemplo de cálculo na rede de Bayes.

Fonte: Adaptado de Thrun (2011)

Teste de Obstáculo 2

Considerando os mesmos dados do exemplo anterior, verifique a probabilidade de haver um obstáculo, dado dois resultados positivos $P(O|++)$, utilizando as fórmulas com normalizador.

	Priori	+	+	P' (multiplicação dos 3)
O	0,01	0,9	0,9	0,0081
-O	0,99	0,1	0,1	0,0396

$$n = (P'(O|++) + P'(-O|++))^{-1} = \frac{1}{0,081 + 0,0396} = 8,2919$$

Assim, temos que:

$$P(O|++) = n \cdot P'(O|++) = 0,0081 \cdot 8,2919 = 0,1698$$

$$P(-O|++) = n \cdot P'(-O|++) = 0,8301$$

A probabilidade de haver um obstáculo é de 16,98%.

Quadro 8 – Exemplo de cálculo utilizando o normalizador.

Fonte: Adaptado de Thrun (2011)

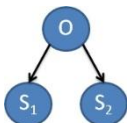


Figura 49 - Medição de uma variável não observável.

Estes exemplos demonstram a medição em uma rede Bayesiana de um estado escondido (não observável) “O” através de estados observáveis (os testes) (Figura 49). Pode-se afirmar que T1 e T2 são condicionalmente independentes, pois caso haja certeza sobre o valor de O, não haverá mais conexão entre T1 e T2.

Apesar de serem exemplos simples, estes podem ser expandidos para a robótica. O estado (pose) do robô não é diretamente mensurável, mas o que é possível saber são as leituras dos sensores que fornecem alguma informação sobre o movimento robô. Esse mesmo conceito pode ser aplicado ao mapeamento, dado que o sensor de distância, a cada varredura, confirma a presença ou não de um obstáculo com certa

probabilidade de erro e quanto maior o número de confirmações, maior certeza haverá sobre a presença ou não de obstáculos.

Estas redes exemplificadas são estáticas e dependem apenas do número de testes realizados, mas quando a rede Bayesiana varia com o tempo, a mesma recebe a denominação de rede Bayesiana Dinâmica.

Outra denominação importante surge quando na rede Bayesiana o estado atual depende apenas do estado anterior, recebendo o nome de rede ou cadeia de Markov (*Markov Chain*).

Assim, se a mesma assume ambas as propriedades, e ainda possui estados que só são possíveis de serem observados através de medições, a rede Bayesiana é denominada de modelo oculto de Markov (*HMM – Hidden Markov Model*).

A HMM pode ser utilizada para prever e analisar séries temporais (Thrun 2011), sendo aplicada principalmente nos campos da robótica, medicina, finanças, linguagem e muitas outras áreas. A HMM é o núcleo central de técnicas como Filtro de Kalman e Filtro de Partículas, que serão discutidos em capítulos posteriores.

Um exemplo do funcionamento do cálculo de uma HMM é feito no Quadro 9, utilizando o modelo da Figura 50. Neste, dada uma sequência de objetos de cores variadas, deseja-se classificá-los em Coloridos e Brancos (C e B, respectivamente). As medições são dadas por um sensor que emite um feixe de luz e mede a intensidade de retorno, sendo R para casos em que há Reflexão do raio pela superfície e A para os casos onde há absorção.

No caso mostrado pela figura, quando há reflexão há uma grande chance do objeto ser Branco. Ainda, o problema foi elaborado considerando que a probabilidade de haver objetos coloridos ou brancos sequenciais (0,6 e 0,8) é maior do que a probabilidade de alternância de cores (0,4 e 0,2).

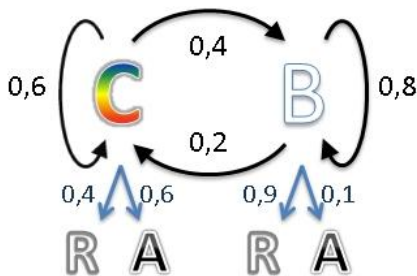


Figura 50 - Modelo de transições de uma HMM.

Fonte: Adaptado de Thrun (2011)

Exemplo 1 – Dado que houve uma Reflexão (R) e que não sei a cor do objeto anterior (Inicial), qual a probabilidade do objeto ser colorido? Ou seja, no instante inicial $P(C_0) = 0.5$ e $P(B_0) = 0.5$, e desejo saber o valor no próximo instante (próximo objeto) dado que houve R no primeiro instante, $P(C_1 | R_1)$.

A solução é dada resolvendo a equação da regra de Bayes:

$$P(C_1 | R_1) = \frac{P(R_1 | C_1) \cdot P(C_1)}{P(R_1)}$$

Para calcular a probabilidade de o objeto ser colorido, tenho de considerar a chance de o primeiro ter sido colorido e que a sequência se mantém, mais a possibilidade de que o objeto era Branco e alternou para Colorido (regra da probabilidade total):

$$P(C_1) = P(C_1 | C_0) \cdot P(C_0) + P(C_1 | B_0) \cdot P(B_0)$$

$$P(C_1) = 0,6 \cdot 0,5 + 0,2 \cdot 0,5 = 0,4$$

$$P(R_1) = P(R_1 | C_1) \cdot P(C_1) + P(R_1 | \neg C_1) \cdot P(\neg C_1)$$

$$P(R_1) = P(R_1 | C_1) \cdot P(C_1) + P(R_1 | B_1) \cdot (1 - 0,4)$$

$$P(R_1) = 0,4 \cdot 0,4 + 0,9 \cdot 0,6 = 0,7$$

$$P(C_1 | R_1) = \frac{0,4 \cdot 0,4}{0,7} \cong 0,229$$

Ou seja, há chance de 22,9% do objeto ser colorido.

Quadro 9 – Exemplo de HMM.

Fonte: Adaptado de Thrun (2011)

Exemplo 2 – Mesmo procedimento que o anterior, mas com a certeza de que o primeiro objeto era Colorido. Ou seja, $P(C_0)=1$ e $P(B_0)=0$.

$$P(C_1) = P(C_1|C_0) \cdot P(C_0) + P(C_1|B_0) \cdot P(B_0)$$

$$P(C_1) = 0,6 \cdot 1 + 0,2 \cdot 0 = 0,6$$

$$P(R_1) = P(R_1|C_1) \cdot P(C_1) + P(R_1|\neg C_1) \cdot P(\neg C_1)$$

$$P(R_1) = P(R_1|C_1) \cdot P(C_1) + P(R_1|B_1) \cdot (1 - 0,6)$$

$$P(R_1) = 0,4 \cdot 0,6 + 0,9 \cdot 0,4$$

$$P(R_1) = 0,6$$

$$P(C_1|R_1) = \frac{0,4 \cdot 0,6}{0,6} = 0,4$$

Ou seja, há chance de 40% do objeto ser colorido.

Quadro 10 – Segundo exemplo de HMM.

Fonte: Adaptado de Thrun (2011)

Exemplo 3 – Deseja-se saber a probabilidade do objeto ser Branco, dada a observação de uma reflexão e sabendo que o primeiro objeto era Branco, $P(C_0)=0$ e $P(B_0)=1$.

$$P(B_1) = P(B_1|B_0) \cdot P(B_0) + P(B_1|C_0) \cdot P(C_0)$$

$$P(B_1) = 0,8 \cdot 1 + 0,4 \cdot 0 = 0,8$$

$$P(R_1) = P(R_1|B_1) \cdot P(B_1) + P(R_1|\neg B_1) \cdot P(\neg B_1)$$

$$P(R_1) = P(R_1|B_1) \cdot P(B_1) + P(R_1|C_1) \cdot (1 - 0,8)$$

$$P(R_1) = 0,9 \cdot 0,8 + 0,4 \cdot 0,2$$

$$P(R_1) = 0,8$$

$$P(B_1|R_1) = \frac{0,9 \cdot 0,8}{0,8} = 0,9$$

Ou seja, há chance de 90% do objeto ser branco.

Quadro 11 – Terceiro exemplo de HMM.

Fonte: Adaptado de Thrun (2011)

Através dos exemplos, é possível entender melhor como é feito o procedimento de cálculo em uma HMM. Em cada situação podemos

observar o efeito nas escolhas dos valores das probabilidades. No primeiro exemplo, é observada uma Reflexão, o que indicaria uma boa chance do objeto ser Branco, com isto, a probabilidade do objeto ser colorido $P(C_1)$ cai de 40%, sem a leitura do sensor, para 22,9% com a nova informação ($P(C_1|R_1)$).

Adicionalmente, como não se sabe a cor do objeto inicial, esta informação não tem grande influência sobre o resultado. Isto fica melhor evidenciado no segundo exemplo, onde foi considerado que o primeiro objeto é colorido. Como há uma maior chance de haverem objetos de cores sequenciais do que alternância entre a cor dos mesmos, esta informação altera a probabilidade do objeto ser colorido de 40% do exemplo anterior para 60%, mas com a leitura dos sensores que favorece o Branco, diminui para 40% contra os 22,9% do primeiro exemplo.

No terceiro exemplo é possível notar que mesmo todos os fatores confirmando que o objeto é Branco e aumentando sua certeza para 90%, ainda há um espaço para erro de 10%. Essa é uma característica muito importante das redes bayesianas, pois dificilmente teremos certeza absoluta sobre algo e sim uma alta ou baixa probabilidade do evento acontecer.

Aplicado à robótica, é possível criar HMMs na localização, exploração e mapeamento. A Figura 51 é um exemplo aplicado à localização, onde X são os estados do robô (posições e velocidades, por exemplo), que dependem apenas do estado do instante anterior mais o sinal de controle u do instante atual. Estes então são corrigidos pelas medições dos sensores (Z) do instante atual, que possuem alguma informação sobre X .

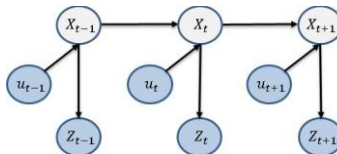


Figura 51 – Rede de Bayes aplicada à robótica.

Fonte: Thrun (2006)

No contexto deste trabalho, é importante saber que o filtro de Kalman é um caso especial de uma HMM, que em vez de utilizar probabilidades individuais, faz aproximações por distribuições Gaussianas nos seus cálculos.

APÊNDICE D – Simulação de Trajetórias

Baseando-se no algoritmo EKF, foi construída uma simulação prévia somente sobre o acompanhamento e previsão de uma trajetória através da fusão.

Para simular o sistema, foi utilizado o software Matlab. Em um primeiro momento, foi elaborada uma simulação com comportamento ideal do robô realizando uma trajetória pré-definida e armazenando os dados das leituras ideais do robô de todos os instantes.

Em uma segunda etapa, foi adicionado ruído a estas leituras (com intensidade de ± 1 para a os sensores da odometria e de $\pm 0,1$ para a IMU) e feitas novas simulações, desta vez tentando prever a trajetória do robô apenas com as informações dos sensores na fusão de dados.

Com isto, temos uma simulação inicial com 2 curvas, uma com a trajetória real da primeira simulação e uma segunda com a tentativa de prever o caminho utilizando somente a fusão entre a IMU e a odometria.

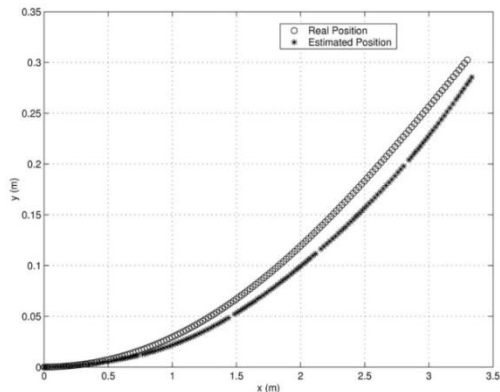


Figura 52 - Resultados obtidos com o robô real.

Fonte: Menegaldo (2008)

Nas simulações iniciais (Figura 53) a trajetória foi bem simples, sendo apenas uma linha reta. O deslocamento real está na cor azul e em vermelho o trajeto previsto pela fusão.

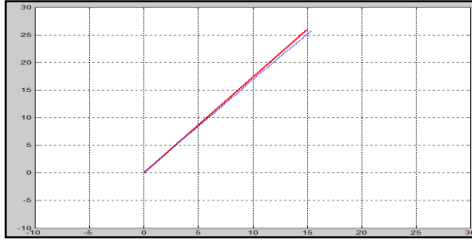


Figura 53 - Simulação do deslocamento no tempo.

Foi feita uma medição de erros durante cada simulação, entre o estado real e estimado, apenas com o intuito de acompanhar a evolução das medições. O erro que é gerado a todo instante vai acumulando, mas com intensidade muito menor do que foi gerado.

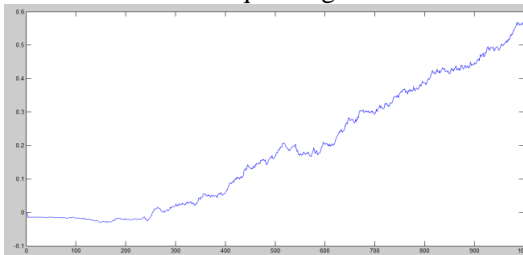


Figura 54 - Erro entre o estado estimado e o estado real no tempo.

É possível notar que após 1000 amostras um pequeno erro de 0,5, concluindo que nos casos em que são utilizados os dados da IMU para complementar a odometria, a trajetória consegue ser acompanhada.

Os outros dois casos simulados estão representados na Figura 55 e Figura 56. Em ambos os casos o filtro de Kalman se comportou adequadamente prevendo o comportamento do robô com boa precisão.

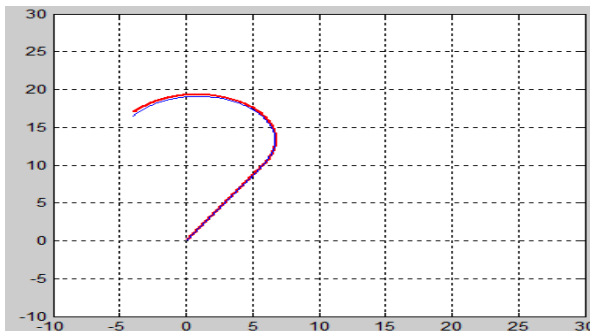


Figura 55 - Segunda trajetória simulada.

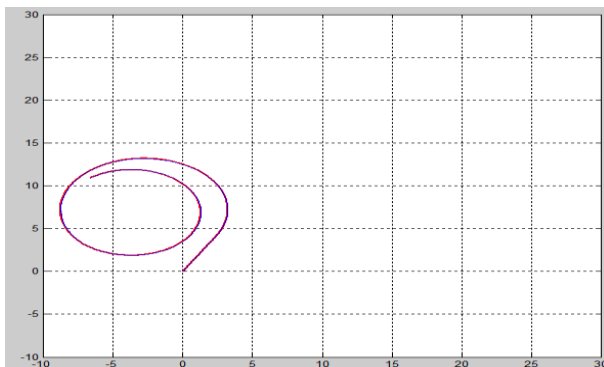


Figura 56 - Terceira trajetória simulada.

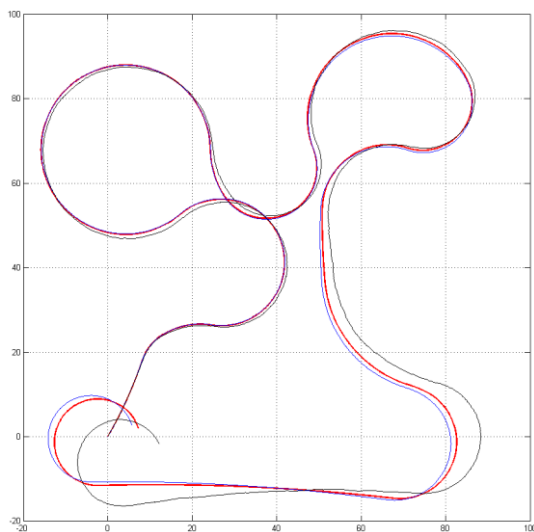


Figura 57 - Simulação de trajetórias.

Para comparar os resultados obtidos com a fusão com a odometria, foram refeitas as simulações para uma trajetória mais complexa apresentada na Figura 57.

Esta segue o mesmo princípio das anteriores, mas agora temos a trajetória real em azul, a trajetória vermelha sendo a da fusão e em preto, a da odometria com ruídos 10 vezes menores do que os simulados pela fusão.

Foi escolhido elaborar os ruídos desta maneira, pois a odometria não é robusta o suficiente e diverge muito da trajetória real. Desta forma, podemos avaliar o impacto no SLAM de uma predição de trajetória não tão ruim para a odometria e a que foi obtida com a fusão.

É possível notar que o comportamento do filtro é adequado, mesmo sem uma exata elaboração das matrizes Q , R e P . Percebe-se também que o erro é deveras pequeno se comparado à complexidade das trajetórias, mas para uma aplicação real, deve-se avaliar uma correta identificação dos parâmetros para melhor desempenho do sistema.