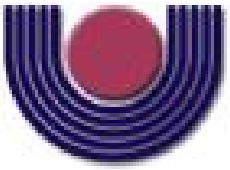


Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Navegação Autônoma de Robôs Móveis Usando Localização e Mapeamento Simultâneos

Alexander Hugo Tártari

CASCAVEL
2012

ALEXANDER HUGO TÁRTARI

NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS USANDO LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Josué Pereira de Castro

CASCAVEL
2012

ALEXANDER HUGO TÁRTARI

NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS USANDO LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Josué Pereira de Castro
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Adriana Postal
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Anibal Mantovani Diniz
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Pedro Luiz de Paula Filho
Colegiado de Ciência da Computação, UTFPR -
Medianeira

Cascavel, 10 de dezembro de 2012

DEDICATÓRIA

Dedico este trabalho a meus pais, Almir e Marli, e à minha irmã, Maiara, os quais estiveram sempre ao meu lado e me apoiaram em todos os momentos da minha vida.

AGRADECIMENTOS

Aos meus pais, Almir Tártari e Marli Salette dos Santos Tártari, por todo amor, atenção e cuidado que sempre recebi, e pela preocupação com meu ensino.

À minha irmã, Maiara Tártari, que sempre me deu força e apoio.

Ao meu orientador, Prof. Josué Pereira de Castro, pela atenção e ajuda no desenvolvimento deste trabalho, assim como por todo conhecimento passado como professor.

À Prof. Adriana Postal, por sua atenção como professora, co-orientadora e coordenadora da disciplina de TCC.

A todos os professores que me passaram conhecimento durante toda a minha vida.

A todos os meus amigos, os quais sempre tiveram muita importância em minha vida.

E finalmente a todas as pessoas que tentam todos os dias fazer do mundo um lugar melhor para se viver.

Lista de Figuras

Lista de Abreviaturas e Siglas

EKF	*Extended Kalman Filter*
KF	*Kalman Filter*
RANSAC	*Random Sample Consensus*
SLAM	*Simultaneous Localization and Mapping*
SMC	*Sequential Monte-Carlo*
USB	*Universal Serial Bus*

Sumário

Resumo

Este trabalho tem como objetivo realizar uma pesquisa exploratória sobre o problema da localização e mapeamento simultâneos, e implementar uma solução para o mesmo. O problema da realização simultânea da localização e do mapeamento é chamado de *Simultaneous Localization and Mapping* (SLAM). A solução aqui apresentada é um método que usa leituras dos sensores internos e pontos de referência (*landmarks*), obtidos através do sensor de distância, para obter precisamente o mapeamento do ambiente e a localização do robô. A cada iteração do método, o robô movimenta-se e refaz as medições dos sensores. Estes resultados são então comparados aos resultados esperados, e com estas informações é realizado um refinamento da localização e do mapeamento. Foi construído um protótipo de robô que conta com um sensor de distância, quatro rodas com sensores de movimentação (*encoders*), um computador de arquitetura x86 para processamento das informações, e uma placa Arduino, que faz as leituras dos sensores e as transmite ao computador através de conexão serial (USB). Para a detecção de *landmarks*, foi implementado o algoritmo de detecção de retas *Random Sample Consensus* (RANSAC). Como foram considerados apenas ambientes estruturados, compostos exclusivamente por paredes e terrenos planos, é possível detectar a posição das paredes, e assim usá-las como *landmarks*.

Palavras-chave: robótica, robótica móvel, mapeamento, localização, inteligência artificial.

Capítulo 1

Introdução

A robótica móvel é uma área da robótica que tem como objetivo o estudo de robôs móveis, dispositivos automáticos capazes de movimentar-se e interagir com o ambiente através de atuadores e sensores. Navegação autônoma é uma área da robótica móvel que estuda técnicas para que robôs possam movimentar-se de forma segura em um determinado ambiente. Estas técnicas dependem do tipo de agente, do ambiente, assim como dos seus requisitos e objetivos. Neste trabalho será apresentada uma solução para o problema de localização e mapeamento simultâneos, também chamado de *Simultaneous Localization and Mapping* (SLAM) [1]. A técnica a ser utilizada faz o mapeamento de ambientes com paredes sólidas, enquanto é mantido um registro da localização do agente (o robô) dentro do mapa. Para a aplicação prática desta técnica, foi construído um robô com atuadores para a movimentação, e sensores de movimentação e distância.

1.1 Objetivos do Trabalho

O objetivo deste trabalho é realizar uma pesquisa exploratória sobre o problema do SLAM, e implementar uma solução para o mesmo. Foi implementado um sistema que constrói um mapa bidimensional reutilizável do ambiente navegado, assim como calcular o posicionamento do robô em relação ao mapa. O ambiente deve ser composto exclusivamente de paredes e terrenos planos. Os objetivos específicos são:

- Construir um protótipo de robô com recursos necessários para a aplicação do método a ser implementado;
- Escolher e aplicar filtros para a normalização de leituras dos sensores;

- Detectar e reconhecer *landmarks*[1] com precisão;

- Realizar a movimentação do robô, utilizando as informações das leituras dos sensores internos de movimentação e uma estratégia de movimentação;

- Comparar leituras em diferentes localizações do ambiente, para refinar o mapeamento e a localização;

- Construir um mapa métrico de duas dimensões para representar o ambiente.

1.2 Justificativas

Esta pesquisa será útil como base para futuros estudos sobre navegação autônoma e reconhecimento de ambientes. As informações contidas neste trabalho permitem uma fácil replicação, permitindo assim que trabalhos futuros sejam efetuados com mais efetividade. Como são apresentados todos os passos do estudo da localização e mapeamento, desde a construção do hardware até as técnicas matemáticas utilizadas, o trabalho tem um grande potencial de ajuda para iniciantes na área, e pode ser usado como ponto de partida para a implementação de sistemas mais complexos.

1.3 Organização do trabalho

No capítulo 2 é apresentado o problema do mapeamento e localização simultâneos, SLAM, assim como o estado da arte das soluções do problema, e aplicações do mesmo. No capítulo 3 é apresentado o modo de obtenção e reconhecimento dos padrões de referência para o método implementado. O capítulo 4 apresenta a fundamentação teórica do filtro de Kalman, e sua aplicação no filtro de Kalman. No capítulo 5 é apresentado o hardware do robô e uma visão geral do seu funcionamento, e também os métodos utilizados para a obtenção dos resultados. No capítulo 6 são apresentados os resultados dos testes efetuados. O capítulo 7 traz as considerações finais e as possibilidades de trabalhos futuros.

[1] Pontos de referência. Para mais detalhes, ver capítulo 3.

Capítulo 2

Mapeamento e Localização

2.1 O Problema SLAM

O Problema SLAM (*Simultaneous Localization and Mapping* - Mapeamento e Localização Simultâneos) é um dos desafios fundamentais da robótica móvel [2], e consiste em efetuar de forma paralela o mapeamento de um ambiente desconhecido e a localização do robô. Esta execução simultânea, apesar de aumentar a complexidade, proporciona mais precisão à navegação, uma vez que o mapeamento e a localização são diretamente ligados. Para que o mapeamento e a localização sejam obtidos, o robô deve ser dotado de atuadores para a movimentação ao longo do ambiente, e sensores para a detecção de padrões de referências (*landmarks*) e odometria. O robô inicia em uma posição desconhecida, em um ambiente também desconhecido. São então procuradas *landmarks* em diversos locais do ambiente, e a cada movimentação os dados de odometria são comparados às posições esperadas das *landmarks*, para que o mapeamento e a localização sejam refinados. A figura 2.1 apresenta um diagrama do processo do SLAM, onde cada ciclo representa uma iteração do processo. Primeiramente são efetuadas as leituras dos sensores, das quais são extraídas *landmarks*, que são utilizadas para a atualização da localização e mapeamento do robô. Por fim, é efetuada a movimentação do mesmo.

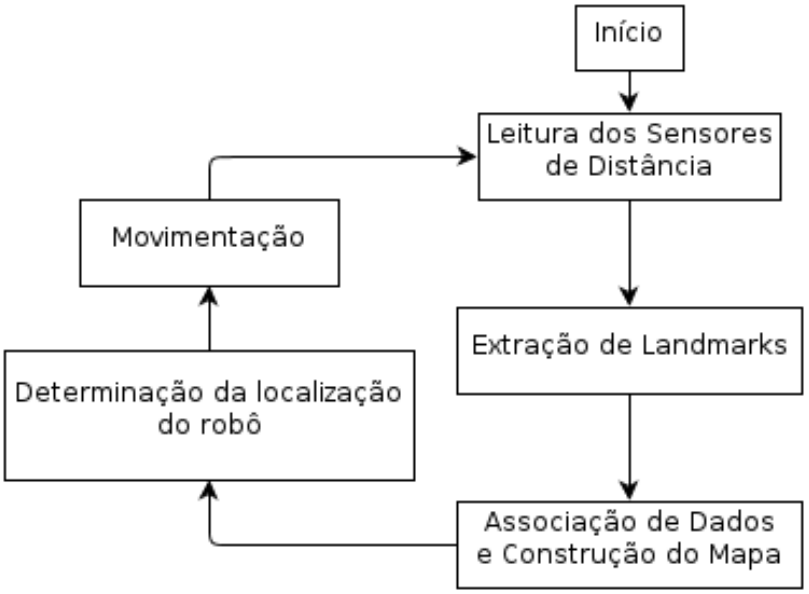


Figura 2.1: O processo do SLAM

A figura 2.2, retirada de Durrant-Whyte e Baley [1], mostra o problema essencial do SLAM. Os elementos com preenchimento em branco representam as posições reais do robô e das *landmarks*. Os elementos com preenchimento em cinza representam posições estimadas dos mesmos, obtidas pelo sensoriamento, que é suscetível a erros. O objetivo do SLAM é obter posições aproximadas que estejam mais próximas possível das posições reais.

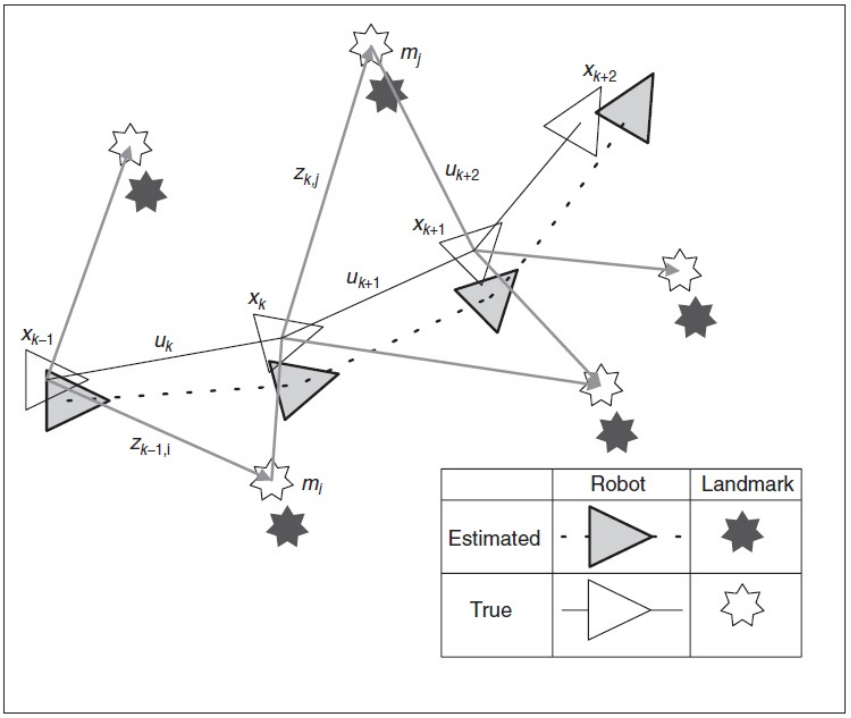


Figura 2.2: Exemplo do processo do SLAM, onde: k: tempo de execução; x_k: posição real do robô no tempo k; u_k: vetor de movimentação do robô no tempo k; m_i: i-ésima *landmark*; z_k: leitura da distância entre o robô e o *landmark*;

2.1.1 Mapeamento

Mapeamento consiste na representação gráfica ou lógica de um ambiente, e pode ser dividido em mapeamento métrico e mapeamento topológico [3].

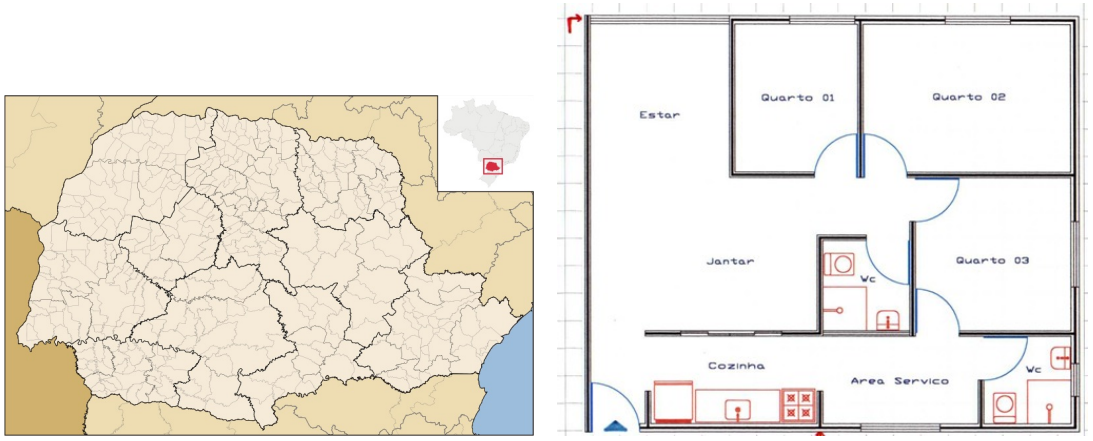
O mapeamento métrico é o método de representação mais comum, e consiste na representação métrica do ambiente, onde cada objeto é representado nas devidas proporções e distâncias, geralmente em escalas diferentes à escala real. É altamente suscetível a ruídos e necessita de alta precisão.

As figura 2.3(a) e 2.3(b) mostram exemplos de mapas métricos, que representam ambientes através de suas medidas físicas, em uma escala menor.

O mapeamento topológico representa o ambiente através de relações entre vários objetos e áreas. É geralmente representado por grafos.

As figuras 2.3(c) e 2.3(d) mostram exemplos de mapeamento topológico. A figura 2.3(c) mostra um grafo, onde podemos considerar cada vértice como uma posição dentro de um mapa,

e os valores das arestas como a distância entre os nós. A figura 2.3(d) apresenta o mapa do transporte metropolitano de São Paulo, onde não são respeitadas as medidas físicas, pois o objetivo da figura é informar as relações entre as diversas estações, independente da posição física da estação.



(a) Mapa do Paraná

(b) Planta de uma casa

(c) Grafo

(d) Mapa do Metrô de São Paulo

Figura 2.3: Exemplos de mapeamento. As figuras (a) e (b) apresentam mapas métricos. As figuras (c) e (d) apresentam mapas topológicos.

2.1.2 Localização

Localização é o registro da posição do robô em relação a um mapa ou pontos de referência. Em um mapa métrico, a localização é representada pela distância entre o robô e os elementos do mapa. Um exemplo de um sistema que utiliza localização em mapeamento métrico é o GPS (*Global Positioning System* - Sistema de Posicionamento Global), que tem um banco de

dados contendo diversos mapas métricos, e tem como função principal localizar-se dentro destes mapas através de sinais de satélites.

2.2 Algoritmos de Solução para o Problema SLAM

Nas próximas subseções veremos as soluções mais utilizadas para o problema SLAM, adaptado de Aulinas et al [2].

2.2.1 Métodos Baseados em Filtros de Partícula

Também chamado de método de Monte-Carlo Sequencial (SMC - *Sequential Monte-Carlo*), este filtro é um método estatístico para obter aproximações numéricas de funções complexas. É capaz de operar sobre conjuntos altamente não lineares, porém exige alto poder de processamento, o que o torna inviável para aplicações de mapeamento em tempo real. Apresenta bons resultados quando destinado apenas à localização, devido à menor quantidade de informações processadas.

2.2.2 Métodos Baseados em Maximização de Expectativas

O método de Maximização de Expectativas é um algoritmo que utiliza estatísticas para estimar as expectativas de leituras futuras. É um método iterativo e não incremental. O algoritmo utiliza os dados de todas as leituras, e estima os possíveis futuros estados. Com estes dados, o sistema analisa os dados conseguidos pelos sensores, e tenta adivinhar qual o estado mais provável. Este processo é executado várias vezes em cada iteração, podendo assim fazer uma filtragem por média ou frequência.

Tem bons resultados quando aplicado ao mapeamento, mas não quando aplicado à localização.

2.2.3 Métodos Baseados no Filtro de Kalman Estendido

O filtro de Kalman (*Kalman Filter* - KF) [4] é um método matemático criado por Rudolf Kalman. Este filtro é utilizado para a remoção de ruídos de conjuntos lineares de medições. Para que seja possível aplicar o filtro de Kalman em sistemas não-lineares, como leituras dos

sensores do robô, o filtro precisa ser modificado. O Filtro de Kalman Estendido (EKF - *Extended Kalman Filter*) é uma variação do Filtro de Kalman que pode ser usado neste tipo de situação. Algoritmos de solução para SLAM usando filtro de Kalman Estendido (EKF SLAM - *Extended Kalman Filter* SLAM) são os mais comuns devido à sua eficácia e simplicidade, e por funcionarem bem tanto para o mapeamento, quanto para a localização. Porém este tipo de solução exige um alto poder computacional em casos onde há uma grande quantidade de *landmarks* detectadas. Este método será explicado com mais profundidade no capítulo 4.

2.3 Estado da Arte

Solà [5] apresenta um guia do funcionamento do SLAM baseado no filtro de Kalman estendido, apresentando exemplos práticos feitos no Matlab. Uma solução para o problema baseada no filtro de Kalman estendido utilizando sensor de ultra-som é apresentada em Bigheti [6]. Werneck e Costa [7] e Santana [8], apresentam meios mais sofisticados de obtenção e processamento dos dados para a execução do SLAM, feitas com o auxílio de uma câmera para a captura de imagens. Fallon et al [9] apresenta uma aplicação de SLAM onde os agentes são pessoas com sensores portáteis.

2.4 Aplicações de Mapeamento e Localização

Mapeamento e Localização são úteis para situações que necessitam de uma navegação segura, onde o robô precisa saber exatamente onde está dentro do ambiente, e também para situações onde é necessário obter um mapa do ambiente para uso futuro. Como exemplo de utilização prática deste tipo de técnica, temos os veículos exploradores de terreno, que precisam mapear um ambiente desconhecido, e ao mesmo tempo localizar-se e navegar de forma segura, podendo calcular uma rota de retorno, e utilizar o mapa posteriormente [10]. A aplicação em veículos terrestres é bastante ampla, permitindo assim o desenvolvimento de veículos autônomos, como apresentado em [11], [12] e [13].Esta técnica pode também ser usada em veículos aéreos [14], aquáticos e submarinos. Pessoas também podem fazer o papel de agente, com sensores portáteis, como pode ser visto em Fallon et al [9].

Capítulo 3

Landmarks

Landmarks são padrões de referência utilizados no processo de mapeamento do ambiente e localização do agente. Estes padrões devem ser facilmente observáveis, distinguíveis entre si, estacionários, e abundantes no ambiente [15].

3.1 Tipos de *landmarks*

Existem dois tipos básicos de landmarks:

- *Landmarks* do tipo R: retas que representam paredes, e são representadas atráves de dois pontos, os quais encontram-se nas extremidades da *landmark*.
- *Landmarks* do tipo P: pontos de interseção entre as *landmarks* do tipo R.

O uso destes dois tipos diferentes de *landmark* é necessário porque os métodos matemáticos usados para refinar o processo não conseguem lidar com *landmarks* formadas por segmentos de reta. Porém *landmarks* formadas por interseções podem ser escassas no tipo de ambiente em que se está trabalhando, portanto as *landmarks* do tipo R são utilizadas como auxílio. Neste capítulo será explicado o método utilizado para a extração, reconhecimento e validação destas *landmarks*.

3.2 Extração de *Landmarks*

Para que a detecção de *landmarks* seja possível, é necessário encontrar padrões de retas nas leituras do sensor de distância. Para isto foi usado o algoritmo RANSAC (*Random Sample Consensus* - Consenso de Amostragem Aleatória). Este algoritmo detecta retas em um conjunto de

pontos. O algoritmo 1 mostra o pseudo-código do algoritmo RANSAC, adaptado de Riisgaard e Blas [15].

Este algoritmo pode ser ajustado pelos seguintes parâmetros:

N: Número máximo de tentativas;

S: Número de leituras selecionadas para computar a linha inicial;

D: Graus de distância do intervalo de escolha das S leituras em relação à primeira leitura selecionada;

X: Distância máxima em que a leitura deve estar da linha computada para que a leitura seja considerada parte da linha;

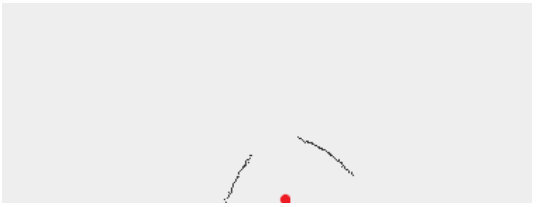
C: Número de pontos que devem estar em uma linha para que esta seja considerada uma *landmark* (Consenso).

A cada iteração, este algoritmo é usado para detectar as *landmarks* do tipo R presentes nas leituras, e então é feita uma segmentação na reta obtida para que sejam respeitadas as dimensões das leituras. Uma vez que estas *landmarks* são detectadas, é verificado se alguma destas *landmarks* têm características semelhantes a alguma outra *landmark* já conhecida, denotando assim que são na verdade a mesma *landmark*, e devem portanto ser unidas para formar uma só. Caso isso ocorra, após o processo de fusão das *landmarks*, o algoritmo RANSAC é executado novamente para esta *landmark*, garantindo assim uma representação mais precisa da *landmark*.

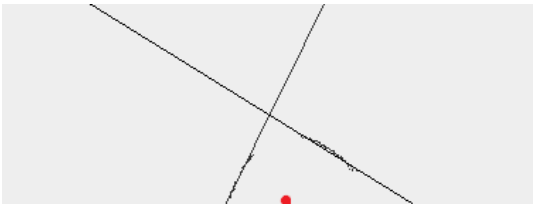
A figura 3.1 apresenta um teste do algoritmo 1. Na figura 3.1(a) podemos ver as leituras do sensor de distância, na figura 3.1(b) vemos a aplicação do algoritmo RANSAC nas leituras, e a figura 3.1(c) mostra a segmentação da reta para respeitar as dimensões das leituras. A figura 3.2 apresenta o ambiente onde o teste foi efetuado.

Algoritmo 1 Algoritmo RANSAC

```
Vetor RANSAC (Vetor Leituras){
    Vetor LinhasExtraídas;
    Enquanto ((tamanho do vetor Leituras > consenso) e (N>0)){

         Decrementa N;

         Selecione uma leitura aleatória do vetor Leituras;

         Selecione S leituras do vetor Leituras num intervalo
         de D graus da leitura escolhida no passo anterior;

         Utilizando o método dos mínimos quadrados, encontre
         a linha com melhor ajuste entre os pontos
         selecionados;

         Se (o número de leituras que estão a X centímetros
             de distância da linha encontrada for maior que
             o consenso C){

            Calcule uma nova linha com o método dos mínimos
            quadrados baseada em todas as leituras que estão
            a X centímetros de distância da linha encontrada
            anteriormente e adicione esta linha ao vetor
            LinhasExtraídas;

            Remova do vetor Leituras os pontos usados para
            calcular a nova linha;
         }
   }
   retorne LinhasExtraídas;
}
```



(a) leituras do sensor de distância



(b) paredes detectadas pelo RANSAC

(c) *Landmarks* detectados

Figura 3.1: Detecção de *landmarks* do tipo R através do método RANSAC

Figura 3.2: Posição do robô em relação às paredes durante o teste

Após a verificação das *landmarks* do tipo R, são detectadas interseções entre todas estas

landmarks, as quais serão consideradas *landmarks* do tipo P. Devido à dificuldade do sensor em detectar interseções entre paredes, foi adicionada uma margem de erro de 10 centímetros nas extremidades das *landmarks* do tipo R, como é exemplificado na figura 3.3.

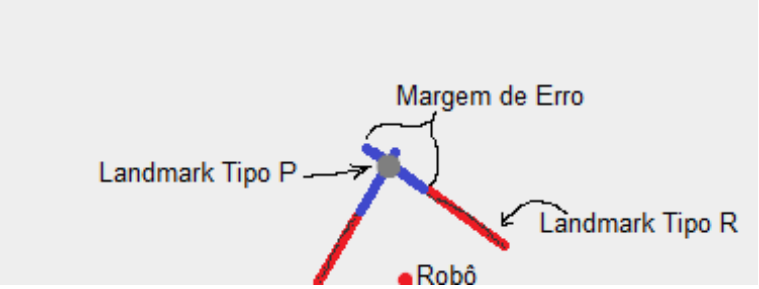


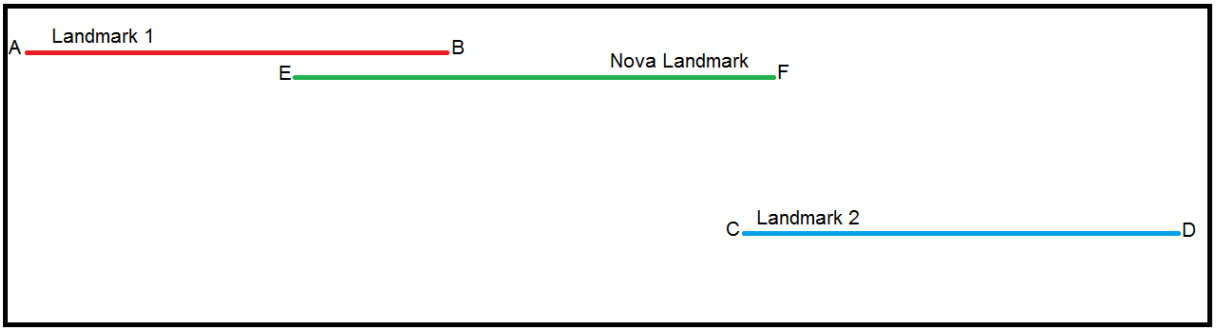
Figura 3.3: Extração de *landmarks* do tipo P

3.3 Reconhecimento e Associação de *Landmarks*

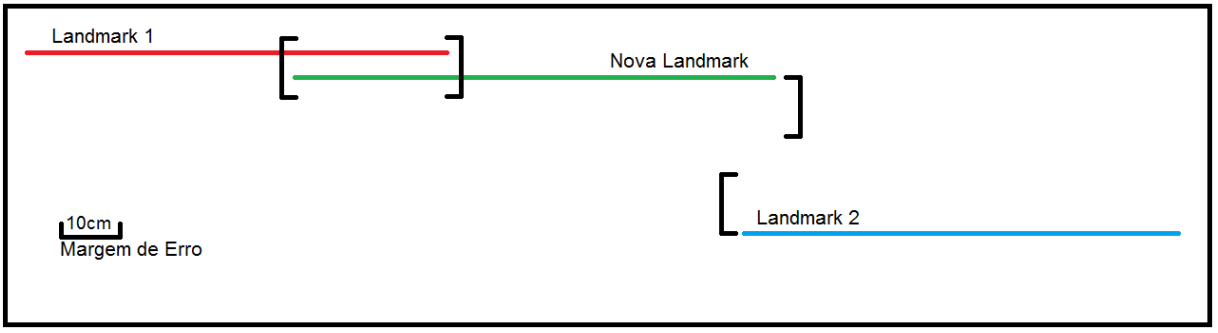
Após a extração das *landmarks*, é preciso verificar se esta *landmark* já foi extraída em uma outra iteração, e então associar as informações para aumentar a precisão. Para *landmarks* do tipo R, é necessário verificar, respeitando uma margem de erro, se algum segmento de reta da *landmark* detectada, que contenha ao menos uma das extremidades da mesma, está em posição similar a algum segmento de reta de alguma *landmark* já conhecida, que também deve conter ao menos uma das extremidades da mesma. Caso duas *landmarks* sejam reconhecidas como partes de uma única *landmark*, é feita uma associação entre as duas, onde suas extremidades mais distantes serão as extremidades de uma nova *landmark*. Para o reconhecimento de *landmarks* do tipo P, basta verificar semelhanças entre as *landmarks* do tipo R que geram as interseções. Como *landmarks* do tipo P são compostas apenas por pontos, a associação é feita mantendo-se apenas uma das *landmarks* a serem associadas.

A figura 3.4 mostra um exemplo do processo de associação de *landmarks* do tipo R. A figura 3.4(a) apresenta um cenário onde já existem duas *landmarks*, Landmark 1 e Landmark 2, e uma nova *landmark* é encontrada. A figura 3.4(b) mostra a verificação de similaridade entre os *landmarks*, levando em conta uma margem de erro de 10 centrímetros. Como Landmark 1 e a nova *landmark* são consideradas partes de uma mesma *landmark*, na figura 3.4(c) é apresentada a associação das duas *landmarks*, que recebe o nome da mais antiga. Essa associação é feita através da utilização dos extremos mais distantes entre as duas *landmarks*, neste caso, o ponto

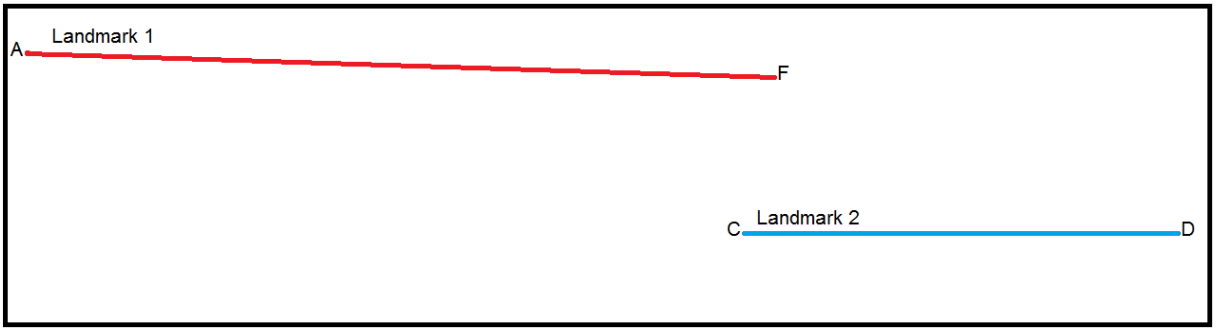
A e o ponto F.



(a) Passo 1: Detecção de Novas *Landmarks*



(b) Passo 2: Verificação de Similaridade entre *Landmarks*



(c) Passo 3: Associação das *Landmarks* com Comportamentos Similares

Figura 3.4: Associação de *Landmarks* do tipo R

Capítulo 4

SLAM com Filtro de Kalman

4.1 Filtro de Kalman

Criado em 1960 por Rudolf Kalman [4], o filtro de Kalman é um método matemático recursivo que produz estimativas dos valores reais de medições realizadas ao longo de um período, as quais podem conter ruídos. Este filtro segue três estágios: predição, observação e atualização, que são apresentados na figura 4.1.

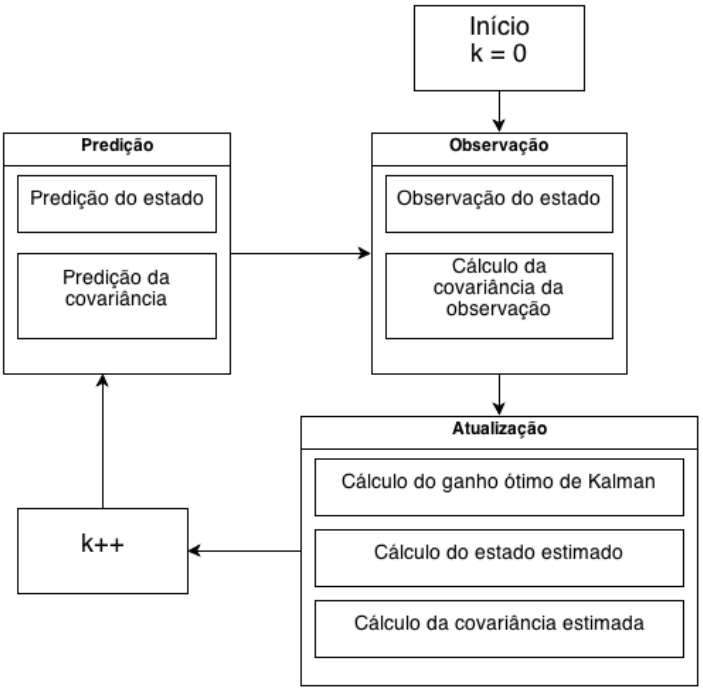


Figura 4.1: Sequência de blocos do funcionamento do fluxo dos dados no Filtro de Kalman.

De acordo com Kalman [4], o modelo para o filtro de Kalman assume que o estado real no

tempo k, representado por $\boldsymbol{x}_k$, é obtido através do estado no tempo (k-1), de acordo com:

$$\boldsymbol{x}_k = \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k \tag{4.1}$$

Onde:

$\boldsymbol{F}_k$**:** modelo de transição de estados;

$\boldsymbol{u}_k$**:** vetor de entradas de controle;

$\boldsymbol{B}_k$**:** modelo das entradas de controle;

$\boldsymbol{w}_k$**:** ruído do processo, assumido como sendo amostrado de uma distribuição normal multivariada de média zero e sua covariância $\boldsymbol{Q}_k$;

O estado do filtro é representado por:

$\hat{\boldsymbol{x}}_{m|n}$**:** a estimativa de $\boldsymbol{x}$ no tempo n, dadas as observações até o tempo m;

$\boldsymbol{P}_{m|n}$**:** a matriz de covariância do erro no tempo n, dadas as observações até o tempo m;

4.1.1 Predição

Este passo faz a predição do estado atual, e de sua covariância, utilizando informações do estado anterior.

Predição do estado:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k \boldsymbol{u}_k \tag{4.2}$$

Predição da covariância:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{4.3}$$

4.1.2 Observação

Esta etapa é responsável pela observação dos dados disponíveis do estado atual.

Observação do estado:

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1} \tag{4.4}$$

Onde:

$\boldsymbol{H}_k$**:** modelo de observação;

$\boldsymbol{v}_k$**:** ruído branco gaussiano de média zero;

$\boldsymbol{z}_k$**:** observação do estado real $\boldsymbol{x} + k$;

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{4.5}$$

Covariância da observação:

$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{4.6}$$

Onde:

$\boldsymbol{R}_k$**:** covariância do ruído da observação;

4.1.3 Atualização

Neste passo, utilizam-se as informações da predição e observação, em conjunto com um fator de ganho, para calcular o estado estimado e sua covariância.

Ganho ótimo de Kalman:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \tag{4.7}$$

Estado estimado:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k \tag{4.8}$$

Covariância estimada:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1} \tag{4.9}$$

4.2 Filtro de Kalman Estendido

O filtro de Kalman estendido é uma variação do filtro de Kalman que permite utilizar sistemas não lineares. Para isso, os modelos de predição e observação devem ser funções diferenciais, permitindo que os mesmos sejam não-lineares. Isto é importante para o SLAM pois o sensoriamento do ambiente retorna dados altamente não-lineares.

A seguir são apresentadas as modificações necessárias dos processos de predição e observação, adaptadas de Terejanu [16] para que os mesmos operem sobre dados não-lineares.

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{4.10}$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{4.11}$$

4.2.1 Predição

Predição do estado:

$$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}) \tag{4.12}$$

Predição da covariância:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \tag{4.13}$$

4.2.2 Observação

Observação do estado:

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1}) \tag{4.14}$$

Covariância da observação:

$$\boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{4.15}$$

4.3 EKF SLAM

O EKF SLAM utiliza o filtro de Kalman estendido para fazer um refinamento da localização do robô. Os passos para a execução do EKF SLAM são:

Leitura do sensor de distância: No começo da iteração, são efetuadas as leituras do sensor de distância, que retornam as distâncias dos obstáculos encontrados num ângulo de 180 graus.

Detecção, associação e reconhecimento de *landmarks*: O método implementado utiliza paredes planas como *landmarks*. A cada iteração, há uma procura das *landmarks* do ambiente. Após encontradas, é verificado se alguma destas *landmarks* tem características semelhante a uma *landmark* já existente. Caso não haja, esta *landmark* é adicionada ao

conjunto de *landmarks*, e ao mapeamento. Caso haja, efetua-se uma comparação entre a *landmark* encontrada e a *landmark* existente, e usando essas informações é possível refinar a posição e forma da *landmark* no mapa e a localização do robô.

Atualização do mapeamento e localização: Utilizando as informações de odometria e leituras dos sensores, aplica-se o filtro de Kalman estendido para atualizar o mapeamento e a localização através do estado estimado.

Movimentação: Ao fim da iteração, o robô movimenta-se de acordo com uma estratégia de movimentação, e um registro da odometria é armazenado para a utilização posterior.

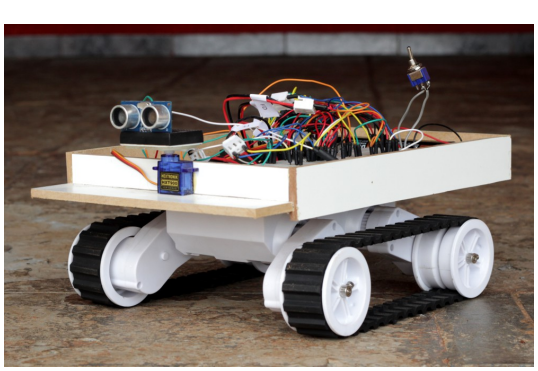
Devido à alta complexidade do filtro, e ao pouco tempo disponível no cronograma, não foi possível aplicar o EKF SLAM ao sistema desenvolvido. Apesar da redução na precisão, o mapeamento e a localização ainda são possíveis sem a utilização do filtro.

Capítulo 5

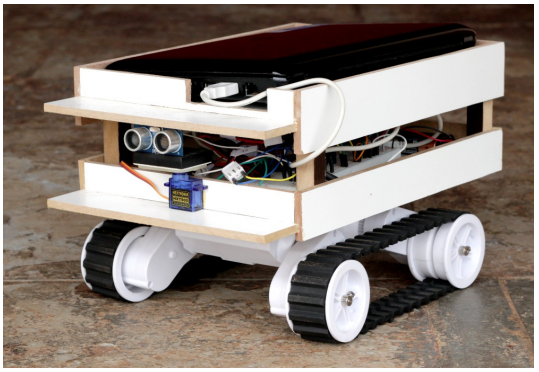
Materiais e Métodos

5.1 O Protótipo de Robô

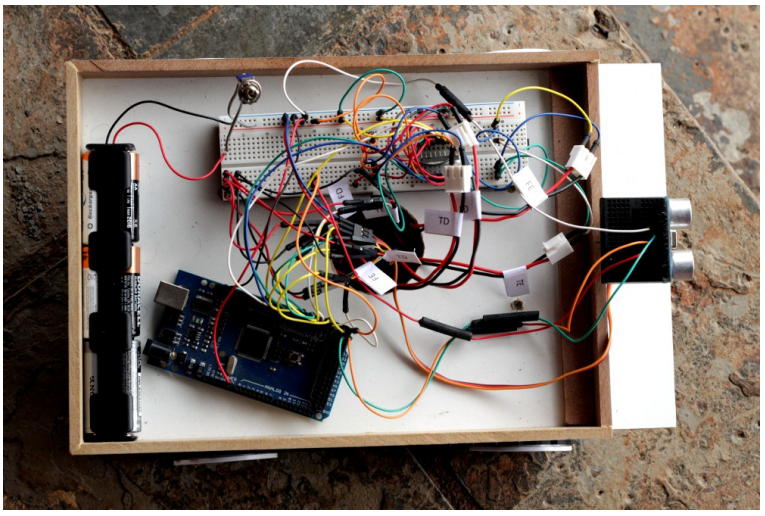
Para possibilitar a implementação da técnica e a realização dos testes, foi construído um protótipo de robô, apresentado na figura 5.1, que conta com um sensor de distância (ultrassônico), quatro rodas com sensores de movimentação (retroalimentação), um computador de arquitetura x86 para processamento das informações, e uma placa Arduino [17], que faz as leituras dos sensores e as transmite ao computador através de conexão serial (USB). Um computador para processamento externo foi necessário devido à possibilidade de mapas com grandes dimensões, e a necessidade de retorno de informações ao usuário. As figuras 5.1(a), 5.1(b) e 5.1(c) mostram o robô construído.



(a) Protótipo sem o suporte para o computador



(b) Protótipo com o suporte para o computador



(c) Vista superior do protótipo

Figura 5.1: Protótipo do robô

5.1.1 O Computador

Para o processamento das informações, foi utilizado um computador portátil que conta com um processador Atom N270 1.6GHz e 1Gb de memória ram DDR2. Este computador comunica-se com o Arduino através de comunicação serial (USB). Uma placa Arduino Mega 1280 [17] serve como intermediário entre os atuadores e sensores, e o computador.

5.1.2 A Plataforma Rover 5

O robô protótipo foi construído sobre uma plataforma Dagu Rover 5 [18], que conta com:

- 4 motores DC independentes;

- 4 *encoders* óticos independentes com resolução de 1000 pulsos a cada 3 giros;

- 4 rodas de plástico;

- Esteira de borracha;

- Suporte para 6 pilhas AA;

5.1.3 Movimentação dos Motores

Para movimentar os motores, foi usada uma ponte H L298N [19]. Devido ao uso das esteiras, a movimentação dos motores deve ser sincronizada entre os motores dianteiros e traseiros, havendo diferenças apenas entre as movimentações dos motores do lado esquerdo e lado direito. Sendo assim, os motores dianteiros e traseiros de cada lado foram ligados de forma paralela entre si.

5.1.4 Sensor de Distância

Para as medições de distância, foi usado o sensor ultrassônico HC-SR04 [20], um sensor de distância de baixo custo encontrado facilmente em lojas virtuais. O sensor emite um som de alta frequência, que pode então refletir em algum objeto, e retornar ao sensor. O tempo de retorno do som é usado então para estimar a distância entre o sensor e o objeto que fez o som refletir. Este sensor tem um alcance nominal de até 5 metros. Porém em testes práticos, nota-se que o sensor apresenta bons resultados para distâncias de até 1 metro. Observa-se ainda que este sensor pode ser substituído por um sensor infravermelho ou laser, que tem funcionamento parecido.

Leitura do Sensor de Distância

O sensor de distância encontra-se na parte frontal do robô, e conta com um servo-motor para que a leitura possa ser feita num ângulo de aproximadamente 180 graus. A cada iteração, são feitas 181 medições, sendo uma a cada grau, começando da posição onde o grau de rotação do servo-moto é 0. Após a medição, estes dados são repassados ao computador através da comunicação serial, que por sua vez faz a transformação das leituras para coordenadas do sistema

cartesiano, levando em conta a posição e direção do robô dentro do mapa. Estas leituras são então interpretadas em busca de *landmarks*.

5.2 Visão Geral do Sistema

Na figura 5.2 podemos ter uma visão geral de como o sistema todo está conectado. Pode-se observar que o Arduino serve como intermediário para a comunicação entre os sensores e atuadores e o computador. O computador faz todo o processamento dos dados recebidos pela comunicação serial, retorna as informações relevantes ao usuário, e comunica-se novamente com o Arduino para enviar as informações de movimentação. A figura 5.3 apresenta as ligações dos componentes que constituem o robô.

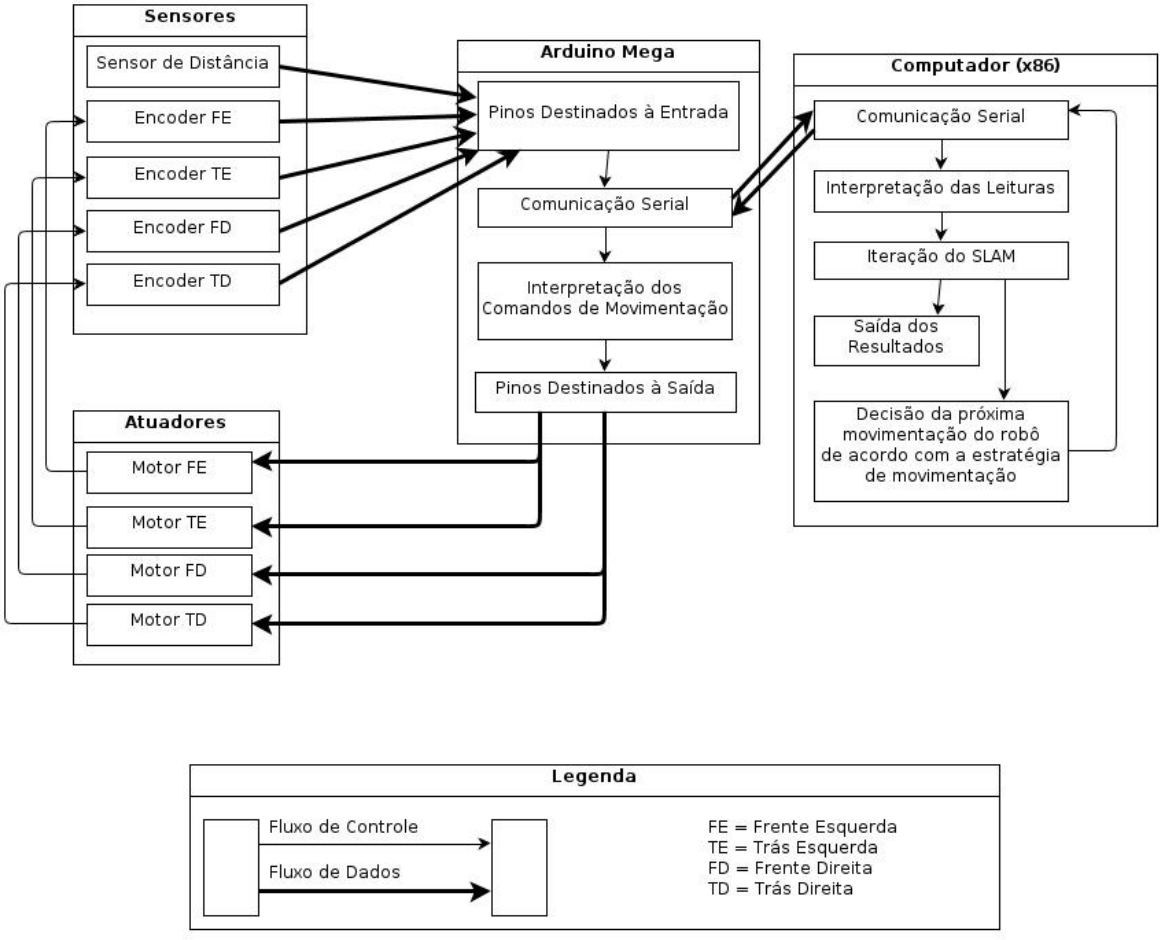


Figura 5.2: Diagrama de estrutura modular

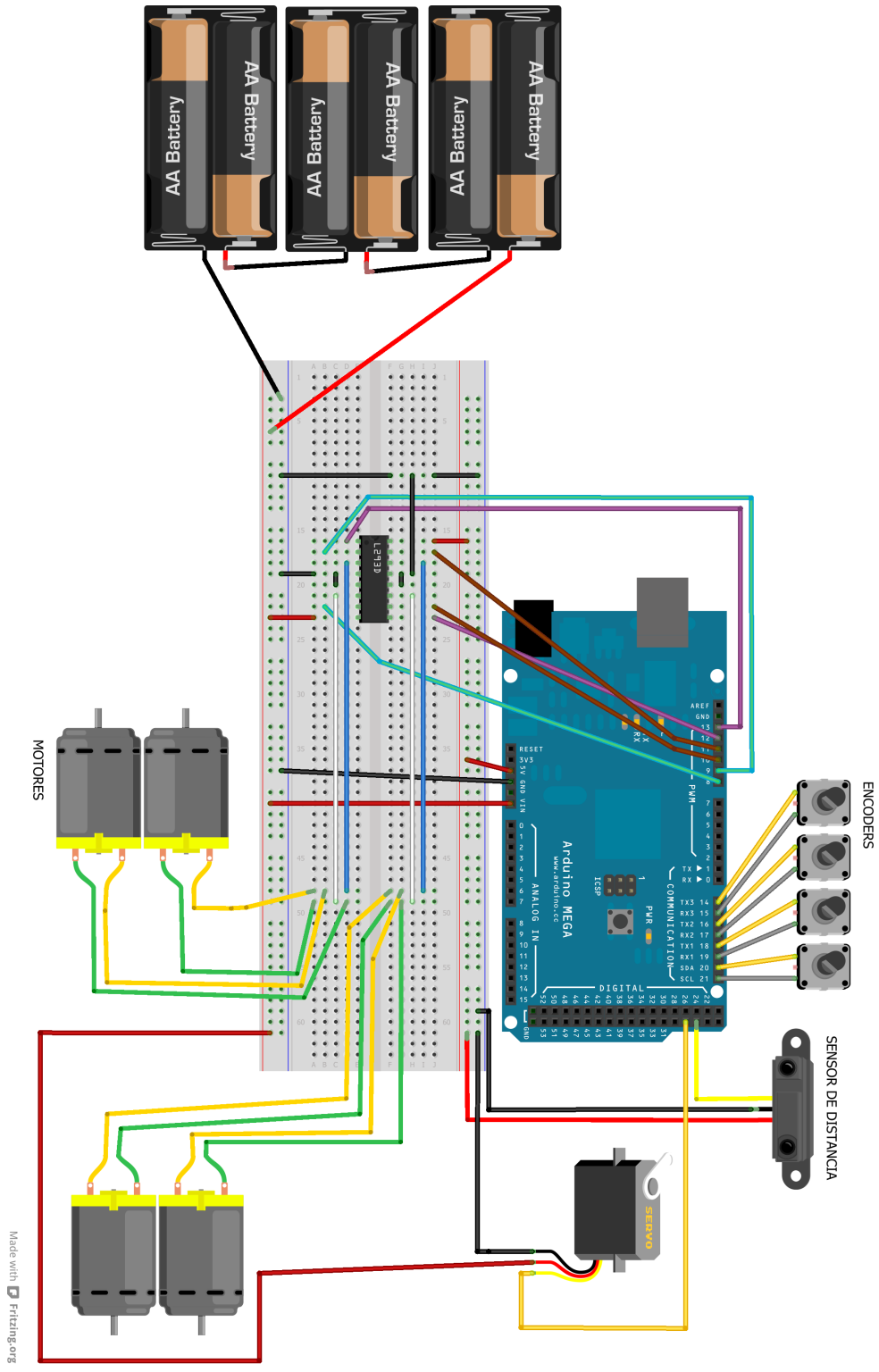


Figura 5.3: Ligações dos componentes do robô

5.3 O Ambiente

O escopo do trabalho envolverá apenas ambientes com paredes e terrenos planos. As paredes devem ser maiores que 20 centímetros de comprimento e 20 centímetros de altura, e devem estar em um plano perpendicular ao terreno.

Estas restrições foram colocadas pelos seguintes motivos:

- Medidas físicas do protótipo;
- Limitações dos meios de sensoriamento. Um sensor de distância ultrassônico não tem precisão suficiente para detectar objetos de pequeno porte ou formas variadas;
- Maior possibilidade de refinamento. Paredes são geralmente extensas, garantindo assim um grande conjunto de dados;
- A simplificação do ambiente permitiu que sua implementação fosse realizada dentro do período previsto no cronograma inicial.

Este tipo de ambiente é muito comum em construções feitas pelo homem, e em problemas de navegação, como labirintos.

5.4 Estrutura do Mapa

Neste trabalho foi utilizado um mapa métrico para representar o mundo, onde será também efetuada a localização do robô. Este mapa será armazenado em uma matriz de valores de duas dimensões, onde cada elemento desta matriz representa uma área específica do ambiente, determinada pelo usuário. Nos testes realizados neste trabalho, cada elemento desta matriz representa 1 centímetro quadrado. Cada elemento da matriz tem um valor que representa o estado. Este estado pode ser: desconhecido, livre ou obstáculo. No começo do mapeamento, todos os elementos da matriz são setados como "desconhecido". À medida que o robô movimenta-se pelo ambiente, o sensoriamento trará resultados que modificarão o estado de alguns elementos. A figura 5.4 mostra uma representação gráfica de um mapa hipotético, onde os elementos em branco representam o estado "desconhecido", os elementos em preto representam o estado "obstáculo", os elementos em azul o estado "livre", e os elementos em vermelho o estado do robô.

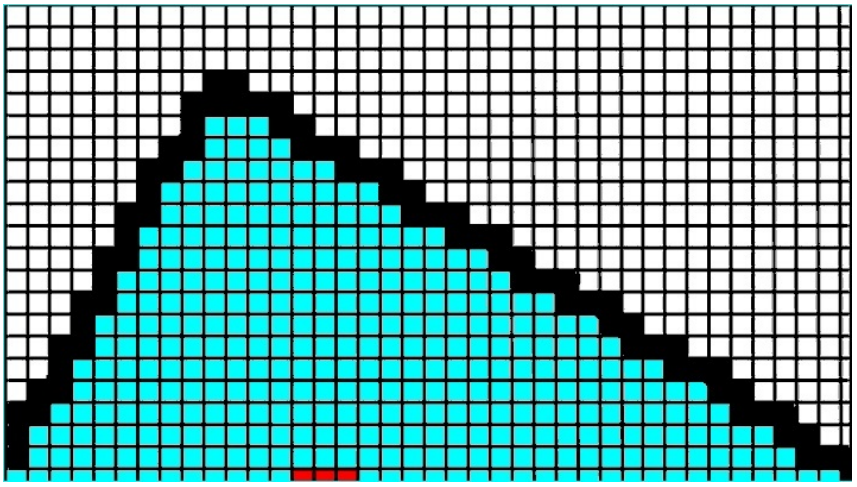


Figura 5.4: Exemplo da representação gráfica do mapa

Além dos estados dos elementos do mapa, são armazenadas as posições das *landmarks*, e as informações de movimentação do robô. Para armazenar *landmarks* do tipo R, somente os pontos extremos do segmento de reta que a forma são salvos, além de sua identificação. Para *landmarks* do tipo P, além do ponto em que está posicionada e a identificação, são armazenadas as informações sobre as *landmarks* do tipo R que a formam, para facilitar comparações de *landmarks*. Já o armazenamento das informações de movimentação do robô é feita com um vetor de estruturas, as quais armazenam o número da iteração, a posição inicial do robô, o ângulo de rotação em relação ao norte do mapa, e a distância percorrida a partir da posição inicial, caso o robô tenha movimentado-se.

5.5 Estratégia de movimentação

O robô inicia o processo em um ponto qualquer do ambiente. A direção para a qual a frente do robô está voltada será considerada o norte. A seguinte estratégia de movimentação será seguida:

1- Procura *landmarks*;

2- Caso não encontre *landmarks*, movimenta-se para frente até encontrar alguma *landmark*;

3- Movimente-se de forma paralela à *landmark* mais próxima, para o lado direito, até encontrar uma *landmark* adjacente, ou não encontre nenhuma *landmark*;

4- Caso encontre uma *landmark* adjacente, voltar ao passo 3;

5- Caso não encontre nenhuma *landmark*, gire 90 graus em sentido anti-horário, e volte ao passo 1;

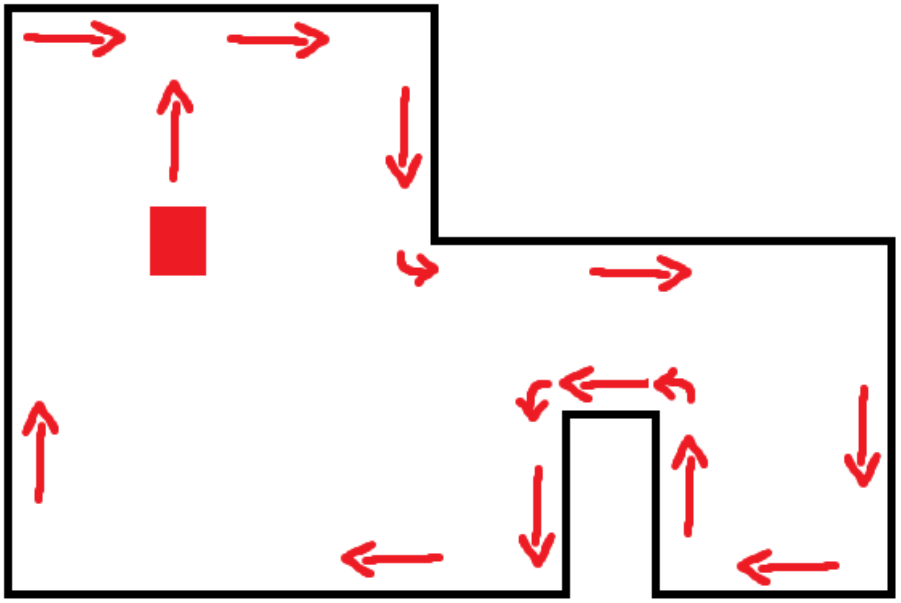


Figura 5.5: Movimentação de acordo com a estratégia de movimentação

Na figura 5.5 podemos ver a aplicação da estratégia em um ambiente com as restrições impostas anteriormente, onde o retângulo vermelho representa o robô.

Capítulo 6

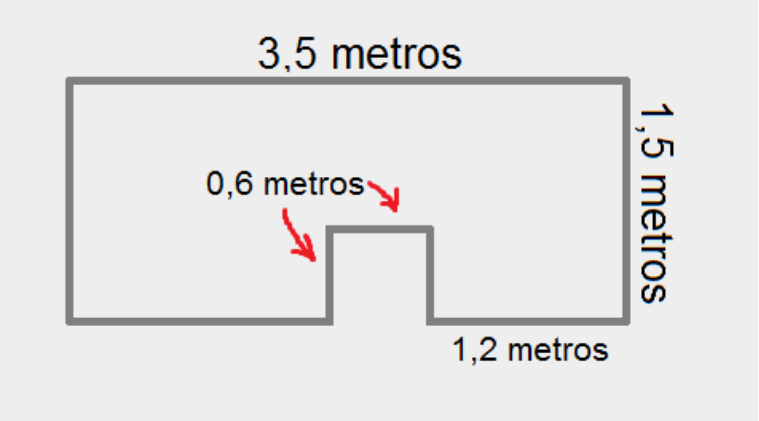
Testes

Neste capítulo serão mostrados os resultados dos testes efetuados com o robô, utilizando as técnicas de mapeamento e localização definidas neste trabalho.

6.1 Primeiro Teste

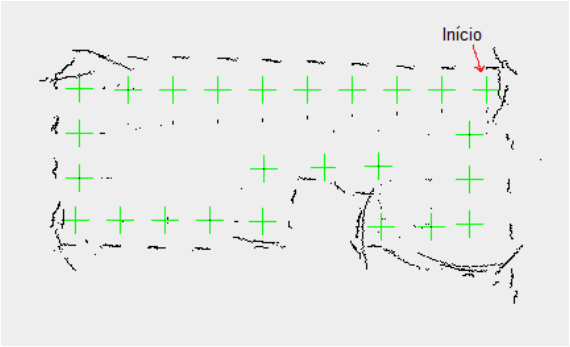
A figura 6.1 apresenta o ambiente no qual foi executado o primeiro teste, que é constituído de paredes de densidades variadas. A figura 6.2 mostra os resultados obtidos pelo robô neste ambiente. A figura 6.2(a) mostra as leituras dos sensores de distância, e a figura 6.2(b) mostra a aplicação do RANSAC nestas leituras. Nestas figuras os sinais verdes em forma de cruz representam a localização do robô, as marcas pretas são as leituras dos sensores, e as linhas vermelhas são os landmarks do tipo R detectados através do RANSAC. A figura 6.2(c) apresenta uma comparação entre o mapa real, e o mapa obtido. Nota-se que, apesar de manter dimensões e formas similares, o mapa gerado ainda é bastante impreciso. A figura 6.2(d) exibe o resultado final do mapeamento, onde as áreas em azul representam as áreas livres, enquanto as áreas em preto representam obstáculos, e as áreas em branco são desconhecidas.

(a) Ambiente onde foi realizado o primeiro teste

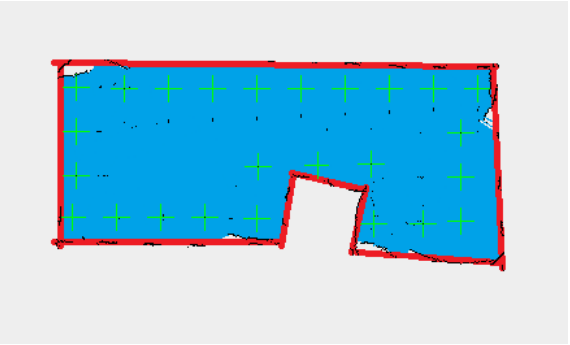


(b) Dimensões do ambiente do primeiro teste

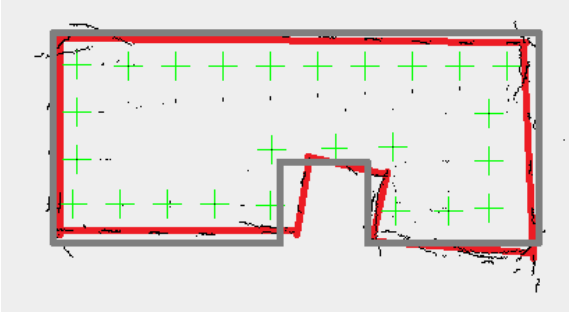
Figura 6.1: Ambiente do primeiro teste



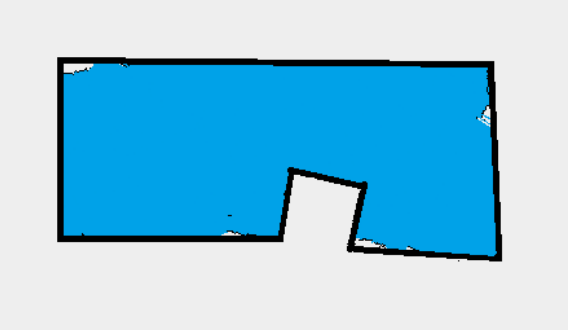
(a) Resultados obtidos das leituras dos sensores



(b) Landmarks e mapeamento obtidos a partir das leituras dos sensores



(c) Comparação entre o mapa obtido, e o mapa real

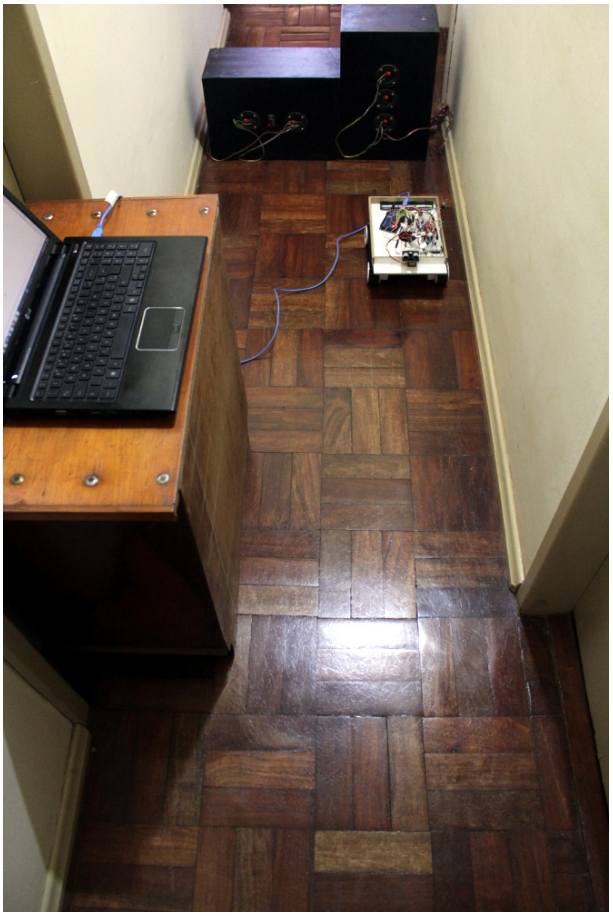


(d) Representação final do mapa

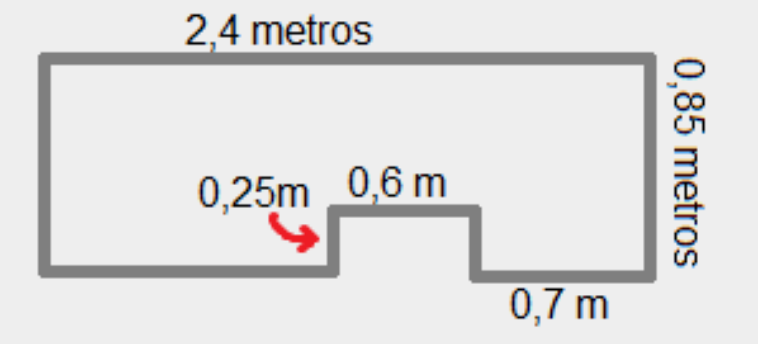
Figura 6.2: Resultados do primeiro teste

6.2 Segundo Teste

O segundo teste foi executado em um ambiente com paredes de alta densidade. A figura 6.3 apresenta o ambiente no qual foi executado o segundo teste. A figura 6.4 mostra os resultados obtidos pelo robô no ambiente, com representações análogas ao primeiro teste. A figura 6.4(a) mostra as leituras dos sensores de distância, e a figura 6.4(b) mostra a aplicação do RANSAC nestas leituras. A figura 6.4(c) apresenta uma comparação entre o mapa real, e o mapa obtido. A figura 6.4(d) exibe o resultado final do mapeamento, onde as áreas em azul representam as áreas livres, enquanto as áreas em preto representam obstáculos, e as áreas em branco são desconhecidas.

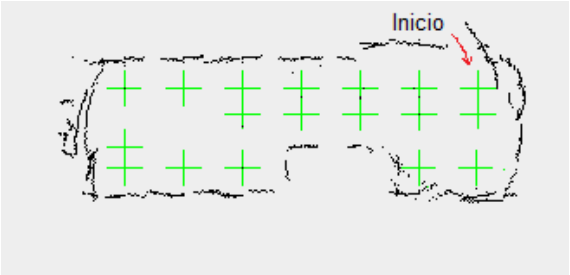


(a) Ambiente onde foi realizado o segundo teste

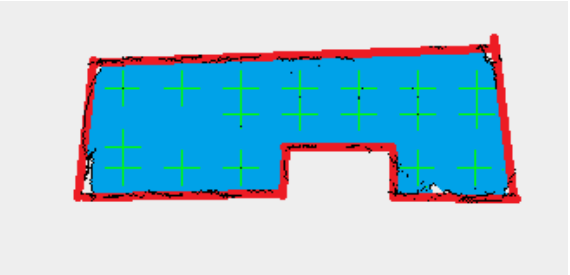


(b) Dimensões do ambiente do segundo teste

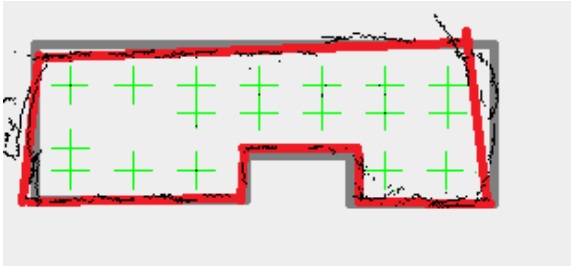
Figura 6.3: Ambiente do segundo teste



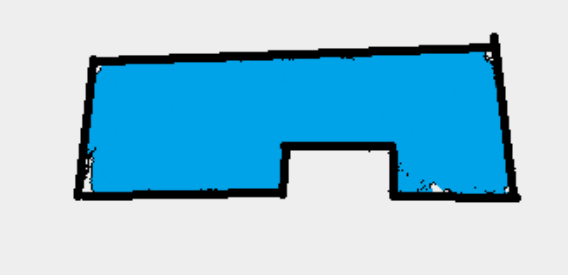
(a) Resultados obtidos das leituras dos sensores



(b) Landmarks e mapeamento obtidos a partir das leituras dos sensores



(c) Comparação entre o mapa obtido, e o mapa real



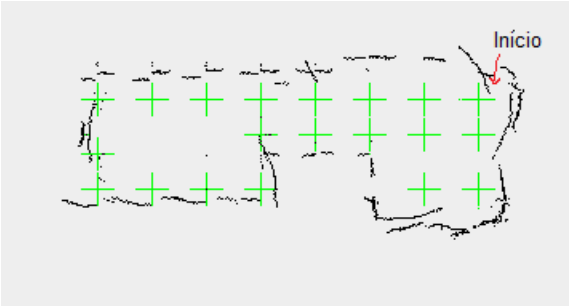
(d) Comparação entre o mapa obtido, e o mapa real

Figura 6.4: Resultados do segundo teste

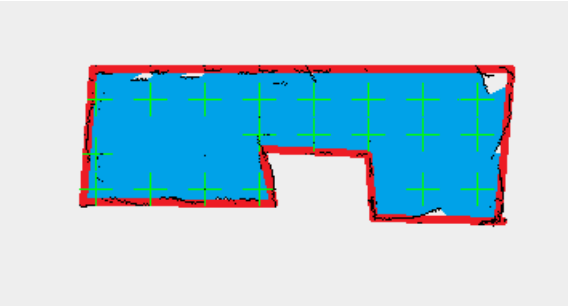
6.3 Terceiro Teste

O terceiro teste foi executado no mesmo ambiente do segundo teste, porém uma de suas paredes foi alterada, retirando a parede de madeira, que tem alta densidade (madeira), e o substituindo por papelão, que tem baixa densidade. Este teste foi feito para se obter uma comparação, e analisar se há uma grande influência do material na leitura dos sensores. A figura 6.5 apresenta o ambiente no qual foi executado o teste. A figura 6.6 mostra os resultados obtidos pelo robô no ambiente, com representações análogas ao primeiro teste. A figura 6.6(a) mostra as leituras dos sensores de distância, e a figura 6.6(b) mostra a aplicação do RANSAC nestas leituras. A figura 6.6(c) apresenta uma comparação entre o mapa real, e o mapa obtido. A figura 6.6(d) exibe o resultado final do mapeamento, onde as áreas em azul representam as áreas livres, enquanto as áreas em preto representam obstáculos, e as áreas em branco são desconhecidas.

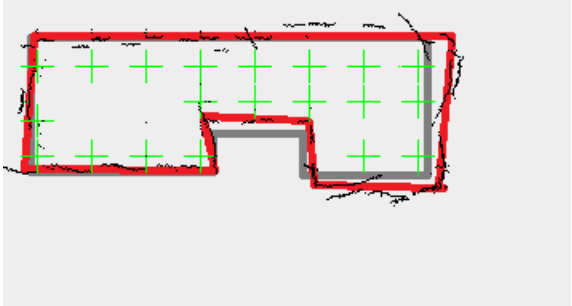
Figura 6.5: Ambiente onde foi realizado o terceiro teste



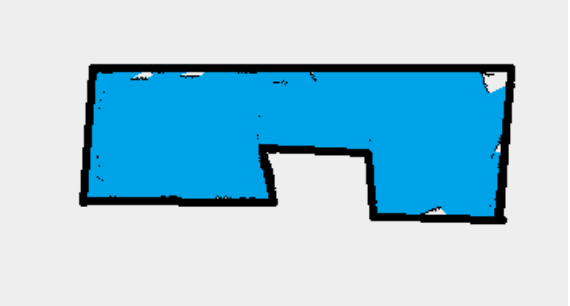
(a) Resultados obtidos das leituras dos sensores



(b) Landmarks e mapeamento obtidos a partir das leituras dos sensores



(c) Comparação entre o mapa obtido, e o mapa real



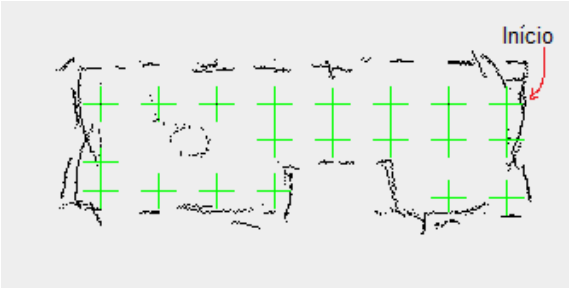
(d) Comparação entre o mapa obtido, e o mapa real

Figura 6.6: Resultados do terceiro teste

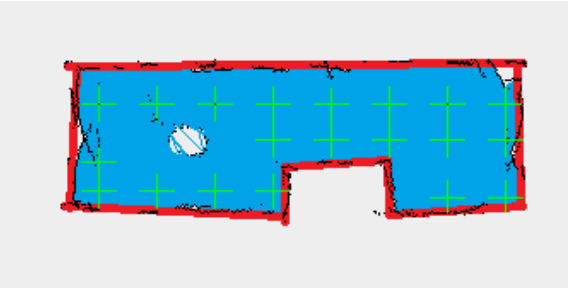
6.4 Quarto Teste

O quarto teste foi executado da mesma maneira do terceiro teste, porém foi adicionado um obstáculo de 11 centímetros de largura, e 11 centímetros de profundidade, em uma posição arbitrária do ambiente. Este teste foi feito para analisar como o robô se comportaria diante o obstáculo. A figura 6.7 apresenta o ambiente no qual foi executado o teste. A figura 6.8 mostra os resultados obtidos pelo robô no ambiente, com representações análogas ao primeiro teste. A figura 6.8(a) mostra as leituras dos sensores de distância, e a figura 6.8(b) mostra a aplicação do RANSAC nestas leituras. A figura 6.8(c) apresenta uma comparação entre o mapa real, e o mapa obtido. A figura 6.8(d) exibe o resultado final do mapeamento, onde as áreas em azul representam as áreas livres, enquanto as áreas em preto representam obstáculos, e as áreas em branco são desconhecidas.

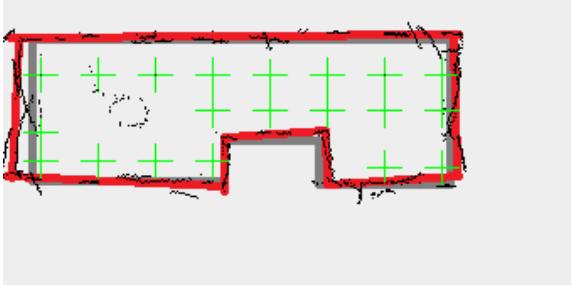
Figura 6.7: Ambiente onde foi realizado o quarto teste



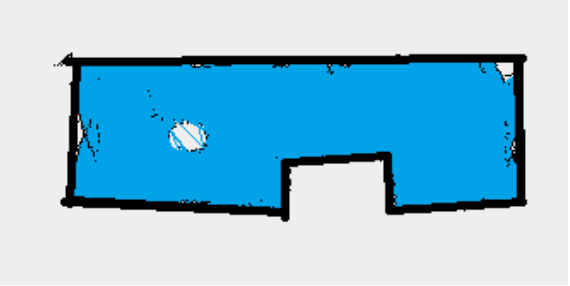
(a) Resultados obtidos das leituras dos sensores



(b) Landmarks e mapeamento obtidos a partir das leituras dos sensores



(c) Comparação entre o mapa obtido, e o mapa real



(d) Comparação entre o mapa obtido, e o mapa real

Figura 6.8: Resultados do quarto teste

Capítulo 7

Considerações finais

O objetivo deste trabalho foi efetuar um estudo sobre o mapeamento e a localização de robôs móveis, simultaneamente, também chamado de problema SLAM. Foram apresentadas diferentes métodos de solução do problema, dos quais a utilização do filtro de Kalman se destacou. Foram também apresentados os meios de extração de padrões de referência do ambiente para usar no processo do SLAM, conseguindo assim aumentar sua precisão e eficiência.

O filtro de Kalman foi certamente o ponto de maior dificuldade do trabalho. Este método é altamente complexo, e por este motivo sua aplicação não foi viável. O algoritmo RANSAC, apesar de ter fácil implementação, necessita de calibrações complexas e relativas ao ambiente, o que exige uma configuração da aplicação diferente para cada ambiente.

O robô foi construído para possibilitar a implementação prática de uma solução para o problema do SLAM, e teve uma boa performance, mesmo com um custo baixo e sensores relativamente imprecisos. Os resultados foram como o esperado, e mostraram que é possível executar, com o equipamento utilizado, o mapeamento do tipo de ambiente proposto, com armazenamento da localização do robô. Embora os resultados não sejam totalmente precisos, eles correspondem à realidade, apresentando erros de precisão devido à imprecisão dos sensores.

7.1 Trabalhos futuros

Para trabalhos futuros, os seguintes itens são de grande importância:

Aplicação de filtros: a aplicação de filtros como o filtro de Kalman estendido podem reduzir a imprecisão no processo do SLAM, trazendo assim resultados mais precisos.

Utilização de sensores mais precisos ou dinâmicos: a aplicação de sensores como lasers ou câmeras de vídeo permite mais precisão, velocidade e novas possibilidades as quais o sensor usado não permitia.

Elaboração de estratégias de movimentação mais complexas: Embora simples e utilizável na maioria dos ambientes do tipo proposto, a estratégia de movimentação usada neste trabalho apresenta limitações e falhas, como a possibilidade de movimentação em círculo ou de ignorar partes do ambiente.

Ampliação do escopo do ambiente: Mudanças no hardware permitirão ao robô explorar novos tipos de ambientes.

Referências Bibliográficas

[1] DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part 1. *Robotics Automation Magazine, IEEE*, v. 13, n. 2, p. 99–108, Junho 2006.

[2] AULINAS, J. et al. The slam problem: a survey. *Artificial Intelligence Research and Development - Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, p. 363–371, Outubro 2008.

[3] THRUN, S. Robotic mapping: A survey. In: LAKEMEYER, G.; NEBEL, B. (Ed.). *Exploring Artificial Intelligence in the New Millenium*. [S.l.]: Morgan Kaufmann, 2002.

[4] KALMAN, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, v. 82, n. Series D, p. 35–45, 1960.

[5] SOLà, J. *Simulataneous localization and mapping with the extended Kalman filter*. Agosto 2012. Http://www.joansola.eu/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM,Consultado na Internet em: 18/10/2012.

[6] BIGHETI, J. A. *Navegação de robôs em ambientes internos usando SLAM*. Dissertação (Mestrado) — UNESP - Universidade Estadual Paulista, São Paulo, 2011.

[7] WERNECK, N. L.; COSTA, A. H. R. *SLAM monocular com reconstrução de planos para ambientes internos*. Tese (Doutorado) — USP - Universidade de São Paulo, São Paulo, Agosto 2011.

[8] SANTANA, A. M. *Localização e Mapeamento Simultâneos de Ambientes Planos Usando Visão Monocular e Representação Híbrida do Ambiente*. Tese (Doutorado) — UFRN - Universidade Federal do Rio Grande do Norte, Natal - RN, 2011.

[9] FALLON, M. F. et al. *Sensor Fusion for Flexible Human-Portable Building-Scale Mapping*. Algarve, Portugal: [s.n.], Outubro 2012. IROS - Intelligent Robots and Systems.

[10] WEISS, S. et al. Intuitive 3d maps for mav terrain exploration and obstacle avoidance. *Journal of Intelligent and Robotic Systems*, v. 61, n. 1-4, p. 473–493, Novembro 2010.

[11] COUTO, L. N. *Sistema para localização robótica de veículos autônomos baseado em visão computacional por pontos de referência*. Dissertação (Mestrado) — USP - Universidade de São Paulo, São Paulo, 2012.

[12] SABBAGH, V. B. *Desenvolvimento de um Sistema de Controle para um Veículo Autônomo*. Dissertação (Projeto Final de Curso) — UFMG - Universidade Federal de Minas Gerais, Belo Horizonte - MG, 2009.

[13] BOAS, E. R. V. *Mapeamento e Localização Simultânea de Ambientes Dinâmicos Aplicados na Navegação de Veículo Autônomo Inteligente*. Dissertação (Mestrado) — UNIFEI - Universidade Federal de Itajubá, Itajubá - MG, 2011.

[14] PRAVITRA, C.; CHOWDHARY, G.; JOHNSON, E. *A Compact Exploration Strategy for Indoor Flight Vehicles*. Dezembro 2011. 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC).

[15] RIISGAARD, S.; BLAS, M. R. *SLAM for Dummies, A Tutorial Approach to Simultaneous Localization and Mapping*. Maio 2004. Http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf ,Consultado na Internet em: 10/07/2012.

[16] TEREJANU, G. A. *Extended Kalman Filter Tutorial*. Buffalo, NY, EUA: [s.n.]. Http://users.ices.utexas.edu/ terejanu/files/tutorialEKF.pdf ,Consultado na Internet em: 04/12/2012.

[17] MARGOLIS, M. *Arduino Cookbook*. 1. ed. Sebastopol, California, EUA: O'Reilly Media, Inc., 2011.

[18] SPARKFUN. *Rover 5*. Http://www.sparkfun.com/datasheets/Robotics/RoverConsultado na Internet em: 12/07/2012.

[19] STMICROELECTRONICS. *DUAL FULL-BRIDGE DRIVER*. 2000. Http://www.datasheetcatalog.org/datasheet/stmicroelectronics/1773.pdf, Consultado na Internet em: 12/07/2012.

[20] ITead Studio. *Ultrasonic ranging module: HC-SR04*. Março 2010. Http://iteadstudio.com/store/images/produce/Sensor/HCSR04/HC-SR04.pdf, Consultado na Internet em: 12/07/2012.