VITOR CAMPANHOLO GUIZILINI

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS COM AUXÍLIO VISUAL OMNIDIRECIONAL

SÃO PAULO
2008

VITOR CAMPANHOLO GUIZILINI

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS COM AUXÍLIO VISUAL OMNIDIRECIONAL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

SÃO PAULO
2008

VITOR CAMPANHOLO GUIZILINI

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS COM AUXÍLIO VISUAL OMNIDIRECIONAL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração:
Engenharia Mecatrônica e de Sistemas Mecânicos

Orientador:
Prof. Livre Docente Jun Okamoto Junior

SÃO PAULO
2008

DEDICATÓRIA

Dedico esse trabalho a todas as pessoas que fizeram eu me tornar o que sou hoje.

AGRADECIMENTOS

Agradeço à minha família pelo amor, pelo carinho e pelo apoio, sem meus pais e minha irmã eu jamais poderia ter chegado até aqui. Agradeço ao meu orientador e aos demais membros do LPA pelo companheirismo, pelo encorajamento e pelas discussões que permitiram o desenvolvimento desse trabalho e fizeram com que eu crescesse como pesquisador e como pessoa. Agradeço aos meus amigos pela confiança, pelas conversas e pelos bons momentos que me deram a força necessária para manter a cabeça erguida durante as horas difíceis. Agradeço aos aventureiros de Treitore pela diversão, pelas conquistas e por manterem a minha imaginação aberta para tudo que há de fantástico no mundo.

EPÍGRAFE

“All you have to decide is what to do with the time that is given to you.”

- Mithrandir

RESUMO

O problema da localização e mapeamento simultâneos, conhecido como problema do SLAM, é um dos maiores desafios que a robótica móvel autônoma enfrenta atualmente. Esse problema surge devido à dificuldade que um robô apresenta ao navegar por um ambiente desconhecido, construindo um mapa das regiões por onde já passou ao mesmo tempo em que se localiza dentro dele. O acúmulo de erros gerados pela imprecisão dos sensores utilizados para estimar os estados de localização e mapeamento impede que sejam obtidos resultados confiáveis após períodos de navegação suficientemente longos. Algoritmos de SLAM procuram eliminar esses erros resolvendo ambos os problemas simultaneamente, utilizando as informações de uma etapa para aumentar a precisão dos resultados alcançados na outra e vice-versa. Uma das maneiras de se alcançar isso se baseia no estabelecimento de marcos no ambiente que o robô pode utilizar como pontos de referência para se localizar conforme navega.

Esse trabalho apresenta uma solução para o problema do SLAM que faz uso de um sensor de visão omnidirecional para estabelecer esses marcos. O uso de sistemas de visão permite a extração de marcos naturais ao ambiente que podem ser correspondidos de maneira robusta sob diferentes pontos de vista. A visão omnidirecional amplia o campo de visão do robô e com isso aumenta a quantidade de marcos observados a cada instante. Ao ser detectado o marco é adicionado ao mapa que robô possui do ambiente e, ao ser reconhecido, o robô pode utilizar essa informação para refinar suas estimativas de localização e mapeamento, eliminando os erros acumulados e conseguindo mantê-las precisas mesmo após longos períodos de navegação. Essa solução foi testada em situações reais de navegação, e os resultados mostram uma melhora significativa nos resultados alcançados em relação àqueles obtidos com a utilização direta das informações coletadas.

Palavras-Chave: SLAM. Localização e Mapeamento Simultâneos. Navegação Autônoma de Robôs. Sistemas de Visão. Visão Omnidirecional.

ABSTRACT

The problem of simultaneous localization and mapping, known as the problem of SLAM, is one of the greatest obstacles that the field of autonomous robotics faces nowadays. This problem is related to a robot's ability to navigate through an unknown environment, constructing a map of the regions it has already visited at the same time as localizing itself on this map. The imprecision inherent to the sensors used to collect information generates errors that accumulate over time, not allowing for a precise estimation of localization and mapping when used directly. SLAM algorithms try to eliminate these errors by taking advantage of their mutual dependence and solving both problems simultaneously, using the results of one step to refine the estimatives of the other. One possible way to achieve this is the establishment of landmarks in the environment that the robot can use as points of reference to localize itself while it navigates.

This work presents a solution to the problem of SLAM using an omnidirectional vision system to detect these landmarks. The choice of visual sensors allows for the extraction of natural landmarks and robust matching under different points of view, as the robot moves through the environment. The omnidirectional vision amplifies the field of vision of the robot, increasing the number of landmarks observed at each instant. The detected landmarks are added to the map, and when they are later recognized they generate information that the robot can use to refine its estimatives of localization and mapping, eliminating accumulated errors and keeping them precise even after long periods of navigation. This solution has been tested in real navigational situations and the results show a substantial improvement in the results compared to those obtained through the direct use of the information collected.

Keywords: SLAM. Simultaneous Localization and Mapping. Autonomous Robot Navigation. Visual Sensors. Omnidirectional Vision.

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

$\boldsymbol{x_t}$	Vetor de localização (x_t, y_t, θ_t) do robô no instante t
$\boldsymbol{x_{0:t}}$	Conjunto de localizações $\boldsymbol{x_t}$ do robô desde o instante inicial da navegação
x_t	Posição do robô no eixo x no plano de navegação
y_t	Posição do robô no eixo y no plano de navegação
θ_t	Orientação do robô (eixo z) no plano de navegação
$\boldsymbol{x_t^p}$	Vetor $\boldsymbol{x_t}$ de acordo com a hipótese da partícula p no instante t
$\boldsymbol{x_s^p}$	Vetor $\boldsymbol{x_t^p}$ obtido em um instante $s < t$ de navegação
$\boldsymbol{\overline{x}_t^p}$	Média da distribuição de probabilidades de $\boldsymbol{x_t^p}$
$\boldsymbol{\Sigma_t^p}$	Matriz de covariância referente à $\boldsymbol{x_t^p}$
$\boldsymbol{\theta_n^t}$	Vetor de posição (x_n^t, y_n^t, z_n^t) do marco n no instante t
x_n^t	Posição do marco n no eixo x do plano de navegação
y_n^t	Posição do marco n no eixo y do plano de navegação
z_n^t	Posição do marco n no eixo z do plano de navegação
$\boldsymbol{\theta_{n,p}^t}$	Vetor $\boldsymbol{\theta_n^t}$ de acordo com a hipótese da partícula p no instante t
$\boldsymbol{\Sigma_{n,p}^t}$	Matriz de covariância referente à $\boldsymbol{\theta_{n,p}^t}$
$\boldsymbol{Q}$	Matriz de erros do sensor
$\boldsymbol{R_{k,p}^t}$	Matriz de projeção da incerteza de $\boldsymbol{z_{k,p}^t}$ nos eixos cartesianos
$\boldsymbol{\delta_t^p}$	Vetor de deslocamento calculado entre $\boldsymbol{x_{t-1}^p}$ e $\boldsymbol{x_t^p}$
$\boldsymbol{\overline{\delta}_t^p}$	Média da distribuição de probabilidades de $\boldsymbol{\delta_t^p}$
$\boldsymbol{\sigma_{p,t}^{\delta}}$	Vetor de desvios-padrão calculado para o vetor de deslocamentos $\boldsymbol{\delta_t^p}$
$\boldsymbol{S_t}$	Conjunto de partículas em um instante t
w_t^p	Peso da partícula p no instante t
$\boldsymbol{Z_t}$	Conjunto de K informações obtidas pelo sensor no instante t
$\boldsymbol{Z_{0:t}}$	Conjunto de observações realizadas desde o instante inicial de navegação
$\boldsymbol{z_k^t}$	Observação incompleta $(\theta_k^t, \varphi_k^t, \psi_k^t)$ número k do conjunto $\boldsymbol{Z_t}$
$\boldsymbol{z_{n,k}^{p,t}}$	Observação completa $\left(d_{n,k}^{p,t}, \theta_k^t, \varphi_k^t, \psi_k^t\right)$ número k do conjunto $\boldsymbol{Z_t}$
$d_{n,k}^{p,t}$	Distância entre $\boldsymbol{x_t^p}$ e $\boldsymbol{\theta_{n,p}^t}$
θ_k^t	Orientação (eixo z) do marco observado por $\boldsymbol{z_k^t}$
φ_k^t	Inclinação (eixo xy) do marco observado por $\boldsymbol{z_k^t}$

$\boldsymbol{\psi_k^t}$	Vetor descritor de uma observação $\boldsymbol{z_k^t}$
$\boldsymbol{\theta_{n,k}^{p,t}}$	Vetor $\boldsymbol{\theta_n^t}$ calculado a partir da observação $\boldsymbol{z_{n,k}^{p,t}}$ e $\boldsymbol{x_t^p}$
$h(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t})$	Função que relaciona a localização do robô x_t^p com a do marco $\theta_{n,p}^t$
$\boldsymbol{H_{n,p}^t}$	Jacobiano de $h(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t})$
$\boldsymbol{y_{k,n}^{p,t}}$	Vetor de diferenças entre $z_{n,k}^{p,t}$ e $h(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t})$
$\boldsymbol{S_{n,p}^t}$	Matriz de covariância da inovação na posição do marco $\boldsymbol{\theta_{n,p}^t}$
$\boldsymbol{K_{n,p}^t}$	Ganho de Kalman
$\boldsymbol{u_t}$	Vetor de controles (v_t, ω_t) fornecido ao robô no instante t
$\boldsymbol{u_{0:t}}$	Vetors de controles fornecidos desde o instante inicial de navegação
v_t	Velocidade linear do robô no instante t
ω_t	Velocidade angular (eixo z) do robô no instante t
$g(\boldsymbol{x_t}, \boldsymbol{u_t})$	Função de transição que propaga $\boldsymbol{x_t}$ no tempo
$\boldsymbol{M^t}$	Conjunto de variáveis m que formam o mapa do robô no instante t
$\boldsymbol{m_{xy}^t}$	Variável $(\boldsymbol{\psi_{xy}^t})$ que modela uma região (x, y) do ambiente
$\boldsymbol{\psi_{xy}^t}$	Probabilidade da região descrita por $\boldsymbol{m_{xy}^t}$ estar ocupada
$\bar{\boldsymbol{m}}_{xy}^t$	Estado ocupado da variável $\boldsymbol{m_{xy}^t}$
$\boldsymbol{m_{n,p}^t}$	Variável $(\boldsymbol{\theta_{n,p}^t}, \boldsymbol{\Sigma_{n,p}^t}, \boldsymbol{\psi_n^t})$ que modela o marco n no mapeamento por características de acordo com a hipótese da partícula p
$\boldsymbol{\psi_n^t}$	Vetor descritor de $\boldsymbol{m_n^t}$
$I(i,j)$	Valor do pixel de coordenadas (x, y) na imagem
$G(i,j,\sigma)$	Máscara gaussiana de centro (x, y) e desvio padrão σ
$L(i,j,\sigma)$	Pixel $I(x, y)$ suavizado gaussianamente por $G(x, y, \sigma)$
$D(i,j,\sigma)$	Diferença de gaussianas no espaço de escalas
$\boldsymbol{f_m}$	Coordenadas (i_m, j_m, o_m, n_m) de um candidato a característica
$\boldsymbol{c_m}$	Coordenadas (i_m, j_m) de um candidato a característica na imagem
i_m	Coordenada x do candidato a característica $\boldsymbol{f_m}$
j_m	Coordenada y do candidato a característica $\boldsymbol{f_m}$
o_m	Índice da oitava onde foi detectado o candidato a característica $\boldsymbol{f_m}$
n_m	Índice da escala dentro da oitava o_m do candidato a característica $\boldsymbol{f_m}$
$\boldsymbol{H_m}$	Matriz Hessiana do candidato a característica $\boldsymbol{f_m}$
$\theta(i,j)$	Orientação de uma característica $\boldsymbol{f_m}$
$m(i,j)$	Magnitude de uma característica $\boldsymbol{f_m}$
$d(\boldsymbol{f_m}, \boldsymbol{f_n})$	Distância euclidiana entre duas características

SUMÁRIO

1. Introdução

O problema da localização e mapeamento simultâneos é um dos problemas fundamentais a ser resolvido em tarefas de navegação autônoma de robôs (Thorpe, Durrant-Whyte; 2001), e por isso vem sendo alvo de extensa pesquisa, como mostrado em (Csorba; 1997) (Bailey; 2002) (Montemerlo; 2003) (Thrun, Fox, Burgard; 2005). Esse problema pode ser descrito como a dificuldade que um robô possui de, partindo de uma posição desconhecida em um ambiente desconhecido, construir incrementalmente um mapa dos locais por onde já passou ao mesmo tempo em que se localiza dentro desse mesmo mapa (Dissanayake et al.; 2001). Uma solução para esse problema permitiria o surgimento de robôs verdadeiramente autônomos, capazes de navegar de maneira segura por ambientes desconhecidos e cumprir objetivos sem a necessidade de auxílio externo de espécie alguma.

No início da navegação o robô não tem nenhum conhecimento referente aos seus estados de localização e mapeamento e deve estimá-los iterativamente conforme se movimenta pelo ambiente. Para isso ele conta com um conjunto de sensores embarcados que se movimentam juntamente com ele, fornecendo informações que são processadas de acordo com modelos pré-estabelecidos e transformadas nessas estimativas. Sistemas de odometria lidam com o problema da localização de maneira incremental, a partir da integração das informações de velocidade fornecidas pelos sinais de controle enviados ao robô e do seu estado anterior de localização. Sensores de medidas (ex: sonares, laser e câmeras) interagem com as estruturas ao redor do robô e fornecem informações referentes à posição delas no ambiente e as características que possuem, permitindo a construção de um mapa.

Contudo, qualquer informação obtida a partir desses sensores será inevitavelmente imprecisa, devido a limitações físicas e a fenômenos que não podem ser previstos e, portanto, modelados computacionalmente. Dessa forma, as estimativas de localização e mapeamento obtidas pelo robô a partir dessas informações possuirão erros que estabelecem uma diferença entre a realidade como ela é e aquela percebida pelo robô. Se esses erros não forem tratados eles tendem a se acumular (Yamauchi, Schultz, Adams; 1998), tornando as estimativas cada vez mais imprecisas e distantes da realidade. Eventualmente chegará o momento em que a imprecisão será tão grande que já não fará sentido utilizar essas estimativas, invalidando o próprio processo utilizado para obtê-las.

Essa dificuldade em se conciliar as etapas de localização e mapeamento de maneira a conseguir resultados consistentes com a realidade mesmo após longos períodos de navegação é conhecida como o problema do SLAM ("Simultaneous Localization and Mapping"), termo proposto em 1991 por Leonard e Durrant-Whyte. Soluções probabilísticas (Thrun, Fox, Burgard; 2005) lidam com esse problema através da minimização simultânea das incertezas envolvidas em cada uma das estimativas, reconhecendo que ambas são dependentes entre si (Araneda; 2004). Resultados precisos de localização são necessários para o estabelecimento de um mapa preciso e vice-versa, e dessa forma não faz sentido abordar apenas um dos problemas, é necessário resolvê-los simultaneamente, eliminando sistematicamente os erros antes que eles se acumulem.

Uma abordagem possível na resolução do problema do SLAM, apresentada inicialmente por Smith e Cheeseman (1986) é o estabelecimento de marcos ("*landmarks*"), que são estruturas no ambiente que o robô pode utilizar como pontos de referência para se localizar durante a navegação (Leonard, Durrant-Whyte, Cox; 1992). Esses marcos são armazenados no mapa que o robô possui do ambiente, e sempre que um novo marco é detectado ele é incorporado a esse mapa. Quando esse marco é reconhecido em um instante posterior, o robô adquire novas informações não apenas relativas à posição da estrutura que ele representa no ambiente, mas também relativas à sua própria localização (Betke, Gurvits; 1997), e com isso consegue refinar ambas simultaneamente, tornando localização e mapeamento globalmente mais precisos.

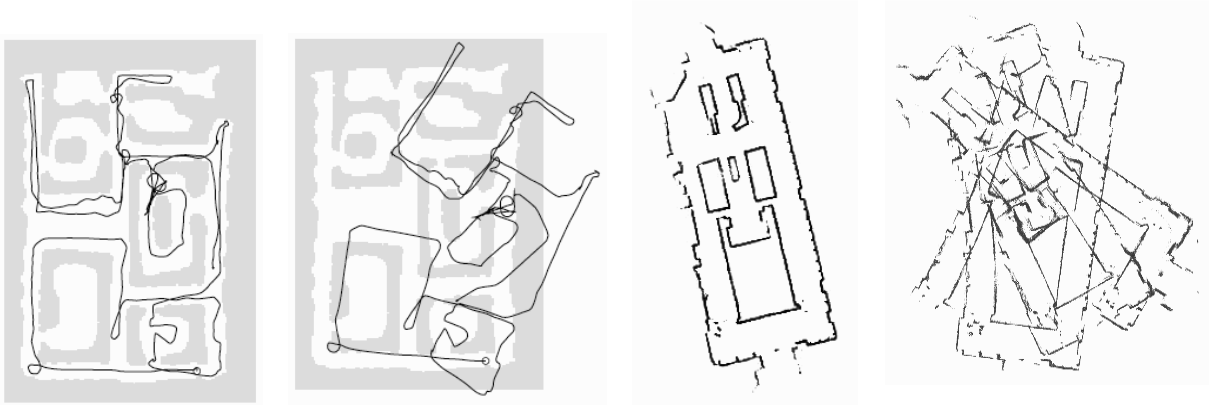
Em trabalhos anteriores (Guizilini et al.; 2007) foi discutida pelo autor uma solução para o problema do SLAM que utiliza sensores de distância esparsos para coletar informações do ambiente, que são insuficientes para o estabelecimento de marcos e resultam em imprecisões residuais de localização e mapeamento. O trabalho aqui apresentado tem como objetivo a solução do problema do SLAM através de um sensor visual para a detecção de marcos naturais ao ambiente, de maneira a melhorar o desempenho do SLAM e aumentar a qualidade das estimativas obtidas durante a navegação. Em especial, é utilizado um sistema de visão omnidirecional para ampliar o campo de visão do robô (Kim, Chung; 2003) e permitir a obtenção simultânea de uma maior quantidade de marcos. Para isso é utilizado um algoritmo de extração de características ("*features*") conhecido como SIFT (Lowe; 1999), que é aplicado diretamente sobre a imagem omnidirecional. As características fornecidas pelo SIFT possuem propriedades de invariância que facilitam a correspondência entre marcos sob diferentes pontos de vista.

A informação obtida pelo sistema de visão durante a navegação é incorporada às estimativas de localização e mapeamento que o robô possui de acordo com o FastSLAM (Montemerlo; 2003), algoritmo que fatora o problema do SLAM e com isso consegue lidar com grandes quantidades de marcos de maneira mais eficiente, além de ser menos sensível a eventuais correspondências erradas porque lida com múltiplos mapas simultaneamente. É apresentada também uma alteração no algoritmo tradicional de FastSLAM que proporciona um aumento nessa eficiência de armazenamento e manipulação dos marcos. A solução aqui proposta foi desenvolvida em ambientes virtuais, com o auxílio do simulador Player/Gazebo, e validada em situações reais de navegação com robôs Magellan Pro e Pioneer 3AT.

2. Localização e Mapeamento Simultâneos

O problema de localização em tarefas de navegação autônoma se relaciona com a capacidade que um robô deve possuir de conhecer a sua posição e orientação a cada instante em um sistema absoluto de coordenadas (Burgard, Fox, Thrun; 1997). Similarmente, o problema de mapeamento se relaciona com a sua capacidade de distribuir as estruturas que se encontram ao seu redor dentro desse mesmo sistema de coordenadas (Gutmann, Konolige; 1999). Nem sempre é possível fornecer ao robô um mapa confiável do ambiente, pois esse mapa pode se alterar durante a navegação ou até mesmo ser desconhecido, cabendo ao robô explorá-lo e gerar esse mapa para aplicações posteriores (Engelson, McDermott; 1992). Da mesma forma, nem sempre é possível fornecer uma localização precisa do robô a cada instante, pois sua posição inicial ou trajetória podem ser desconhecidas, e existem situações onde sistemas absolutos de posicionamento, como o GPS, não podem ser utilizados ou não fornecem resultados confiáveis (Negenborn; 2003).

Nesses casos, o robô deve obter estimativas de localização e de mapeamento através dos seus próprios sensores embarcados, cujas informações fornecidas serão inevitavelmente ruidosas, devido a limitações físicas e a uma grande quantidade de fenômenos que não podem ser modelados devido à sua complexidade ou imprevisibilidade. Esses erros fazem com que as estimativas obtidas pelo robô sejam imprecisas, limitando a sua aplicabilidade em algoritmos posteriores que dependam de resultados precisos de localização e de mapeamento.



(a) *(b)* *(c)* *(d)*

Figura 2.1 - Influência da incerteza na localização e mapeamento (Thrun, Fox, Burgard; 2005).
(a) Localização sem erros (b) Localização com erros
(c) Mapeamento sem erros (d) Mapeamento com erros

As primeiras tentativas de se lidar com esses erros procuraram fazê-lo através da construção de sensores cada vez mais precisos, que possuiriam um menor erro inerente e por isso proporcionariam um resultado final mais confiável. Mas o aumento da precisão fez com que os sensores utilizados ficassem cada vez mais caros, e logo essa abordagem deixou de ser atrativa, sendo necessários investimentos cada vez maiores para gerar aumentos de precisão cada vez menores. Adicionalmente, era óbvio que seria impossível eliminar completamente os erros inerentes aos sensores, devido a fatores que estão fora do controle dos projetistas. Em outras palavras, essa abordagem apenas ocultava o problema ao invés de eliminá-lo, e em uma navegação suficientemente prolongada esses erros eventualmente se tornariam grandes o suficiente para comprometer os resultados finais.

Sendo impossível eliminar esses erros a solução foi então lidar com eles, e para isso é necessário que eles sejam incorporados aos modelos utilizados, que até então apresentavam características determinísticas (Maybeck; 1979). Individualmente esses erros não podem ser previstos, mas quando considerados como um conjunto de fenômenos isolados percebeu-se que eles podem ser tratados estatisticamente, apresentando padrões que podem ser modelados. Essa visão impulsionou o surgimento da chamada robótica probabilística (Thrun, Fox, Burgard; 2005), onde os estados de localização e mapeamento do robô deixam de apresentar valores únicos e se tornam distribuições de probabilidades que indicam todos os estados possíveis dada a precisão das observações realizadas até aquele momento. A obtenção de estimativas precisas se torna então uma questão de minimizar as incertezas contidas em cada uma das variáveis de interesse, o que pode ser feito a partir do acúmulo de informações (Elfes; 1987) visando eliminar os erros aleatórios gerados pelos ruídos apresentados pelos sensores.

Quando tratados probabilisticamente ambos os problemas possuem soluções simples, mas para isso é necessário que o outro já tenha sido resolvido, como é o caso da localização dado o mapa do ambiente (Dellaert et al.; 1999) ou do mapeamento dada a localização do robô a cada instante (Thrun; 2002a). Isso acontece porque ambos os problemas estão vinculados entre si, e por isso não podem ser tratados independentemente. A localização é necessária no mapeamento porque as observações realizadas pelos sensores do robô são relativas à sua posição no ambiente, portanto a sua localização é necessária para distribuir essas informações em um sistema global de coordenadas. Já a localização depende do mapeamento porque o robô deve ser capaz de reconhecer estruturas no ambiente de maneira a obter estimativas absolutas de sua posição em relação a elas e com isso se posicionar dentro do mapa.

Dessa forma, caso uma dessas estimativas apresente erros eles serão propagados para a outra, aumentando a sua imprecisão, e devido à natureza iterativa do processo esses erros rapidamente se acumularão e invalidarão os resultados finais. Torna-se necessário, portanto, lidar com ambos os problemas simultaneamente, procurando minimizar globalmente as incertezas envolvidas ao invés de lidar com elas individualmente. Com isso os erros gerados a cada iteração pelos sensores utilizados são sistematicamente eliminados antes que se acumulem, mantendo as incertezas inerentes às estimativas de localização e mapeamento estáveis durante todo o período de navegação.

2.1. Localização

Um componente fundamental para qualquer sistema autônomo de navegação é a capacidade de se localizar dentro do ambiente no qual se encontra (Jensfelt; 2001). Durante a navegação um robô recebe sinais de controle que fazem com que ele se movimente pelo ambiente, partindo de um ponto e chegando até outro. Em uma navegação autônoma o robô deve ser capaz de calcular iterativamente quais são os sinais de controle necessários para que ele consiga alcançar o seu objetivo de maneira segura e eficiente. Dessa forma, ele deve ser capaz de monitorar constantemente a sua localização relativa às estruturas que existem ao seu redor, de maneira a perceber o seu deslocamento através do ambiente e conseguir definir se esse deslocamento fez com que ele se aproximasse do seu destino ou não.

2.1.1. Odometria

O modelo mais simples de localização envolve apenas parâmetros internos do robô, e por isso não depende da obtenção de informações do ambiente ao seu redor. Um sistema de odometria consegue, com o auxílio de *encoders*, determinar a rotação dos eixos dos atuadores responsáveis pela movimentação do robô em um determinado intervalo de tempo, e integrando esses valores consegue calcular qual o deslocamento realizado nesse intervalo. Esse deslocamento é gerado de acordo com um vetor de controles que indica, por exemplo, as velocidades que o robô deve apresentar nesse intervalo e pode ser fornecido por um controlador humano ou calculado pelo robô de acordo com o objetivo que se deseja alcançar. Supondo que a localização

$\boldsymbol{x_t}$ do robô no instante imediatamente anterior é conhecida, assim como o vetor de controles $\boldsymbol{u_t}$, é possível determinar iterativamente qual será a sua localização $\boldsymbol{x_{t+1}}$ através de uma função de transição $g(\boldsymbol{x_t}, \boldsymbol{u_t})$ que depende dos parâmetros do problema.

$$\boldsymbol{x_{t+1}} = g(\boldsymbol{x_t}, \boldsymbol{u_t}) \tag{2.1}$$

Essa propagação da localização no tempo não leva em consideração fenômenos imprevisíveis, como deslizamento de rodas e terrenos irregulares, e por isso os seus resultados são apenas uma estimativa do estado atual do robô naquele instante. Essa incerteza pode ser modelada com uma distribuição de probabilidades $p(\boldsymbol{x_{t+1}}|\boldsymbol{x_t}, \boldsymbol{u_t})$, ou seja, a probabilidade do robô se encontrar na posição $\boldsymbol{x_{t+1}}$ dado que ele partiu da posição $\boldsymbol{x_t}$ e se deslocou de acordo com o vetor de controle $\boldsymbol{u_t}$. Essa distribuição é obtida empiricamente de maneira a quantificar a sensibilidade que o sistema possui em relação surgimento de erros, e seus parâmetros são individuais para cada robô.

A forma da distribuição também pode variar, e isso define as ferramentas que podem ser utilizadas para o seu tratamento. Caso seja modelada como uma distribuição gaussiana, cujas propriedades são necessárias em algumas aplicações (Welch, Bishop; 1995), ela será definida por dois parâmetros: um vetor de médias $\overline{\boldsymbol{x}}_{\boldsymbol{t+1}}$ que representa o estado com maior probabilidade de ser o correto, e uma matriz de covariância $\boldsymbol{\Sigma_{t+1}}$ que quantifica a incerteza em relação a essa média. Para distribuições genéricas uma ferramenta comum é o Filtro de Partículas (Rekleitis; 2003) (Thrun; 2002b), que simula qualquer distribuição a partir de um conjunto finito de amostras, cada uma representando um possível estado de localização do robô. A Figura 2.2 mostra o efeito do acúmulo de erros de localização sob a forma do espalhamento das partículas por uma área cada vez maior.

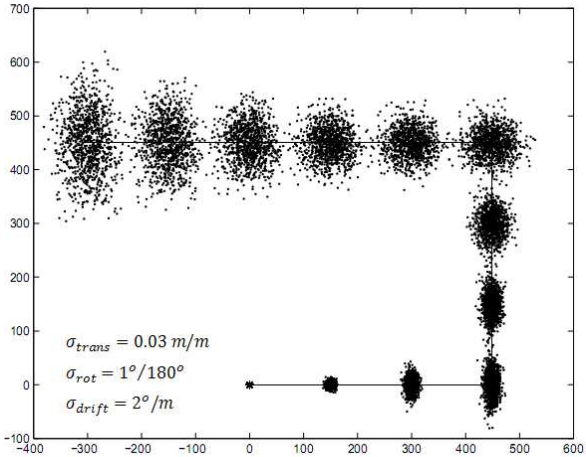


Figura 2.2 - Aumento de incerteza em um Filtro de Partículas. O robô começa em (0,0) e segue até (-300, 450), com erros gaussianos de média $\mu = 0$ e desvios padrão σ distintos para cada tipo de deslocamento.

Qualquer que seja a distribuição utilizada é importante perceber que, devido ao caráter iterativo que esse modelo possui a incerteza inerente à localização tende a aumentar com o tempo, pois é propagada para as próximas iterações e se acumula com os erros gerados em instantes posteriores (Figura 2.2). Não há um limite superior para essa incerteza, e por isso ela se tornará monotonicamente maior durante todo o período de navegação, eventualmente invalidando as estimativas obtidas e impedindo que o resultado seja utilizado em qualquer aplicação posterior, como é o caso do próprio mapeamento.

2.1.2. Estimativa de Localização

Uma estimativa de localização cuja incerteza possui um limite superior só pode ser obtida quando o robô utiliza referenciais absolutos de posicionamento, que não dependem do seu estado de localização anterior (Bailey; 2002). Sistemas de posicionamento global, como o GPS, são uma alternativa, contudo existem situações onde esse sistema não pode ser utilizado, como é o caso da navegação em ambientes internos, ou em que ele não fornece informações confiáveis ou com a freqüência necessária. Outra possibilidade é a busca por estruturas no ambiente, aqui supostas fixas, que o robô pode utilizar como pontos de referência para poder se localizar (Betke, Gurvits; 1997). Estruturas do ambiente utilizadas com essa finalidade são conhecidas como marcos ("*landmarks*"), e devem possuir propriedades que permitam a sua detecção e posterior reconhecimento (Altermatt et al.; 2004).

Supondo (Figura 2.3) que em um determinado instante t o robô é capaz de reconhecer no ambiente um conjunto N de marcos cujas posições absolutas $\boldsymbol{\theta_n} = \{x, y\}^T$ já lhe são conhecidas, é possível determinar qual a sua estimativa de posição a partir dos valores $\boldsymbol{z_k^t} = \{d, \varphi\}^T$ fornecidos pelas observações que foram correspondidas com os marcos. A posição desses marcos no ambiente não depende de valores de localização do robô, e por isso a estimativa obtida será absoluta, limitada apenas pela incerteza relativa à posição do marco no ambiente e do sensor utilizado para realizar a observação. Essa estimativa pode ser incorporada àquela obtida incrementalmente pela odometria de maneira a gerar uma terceira estimativa mais precisa do que as duas iniciais (Negenborn; 2003). De acordo com a natureza do sensor utilizado essa informação pode não ser completa, o que requer a utilização de abordagens diferentes. Sensores de sonar e rádio, devido à grande dispersão angular, podem gerar observações onde apenas a distância pode ser utilizada com precisão (Kehagias, Diugash, Singh; 2006), enquan-

to sensores visuais permitem, com apenas uma imagem, a obtenção de observações referentes apenas à orientação das estruturas observadas (Bekris, Glick, Kavraki; 2006).

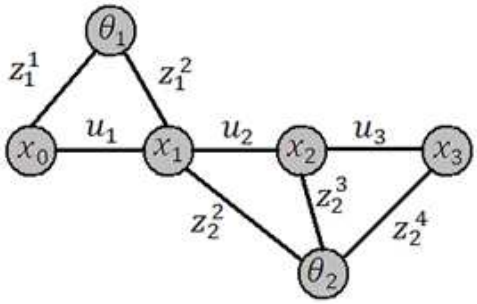


Figura 2.3 - Relações entre mapeamento e localização. O robô se desloca entre as posições x_t de acordo com os comandos u_t e detecta os marcos θ_n através das observações z_n^t.

A informação absoluta que um mapa do ambiente é capaz de fornecer também faz com que seja possível resolver problemas clássicos, como é o caso da localização global (Jensfelt, Kristensen; 1999), onde o robô não conhece sua posição inicial dentro do seu modelo de mapa, e o "seqüestro" de robôs (Fox, Burgard, Thrun; 1999), onde o robô é subitamente levado para outra posição e deve se localizar. Contudo, não é sempre que um mapa confiável do ambiente pode ser fornecido anteriormente à navegação, seja porque o próprio ambiente ainda não foi explorado ou porque ele ainda não é conhecido, e nessas situações o robô deve também construir incrementalmente o mapa que utilizará para se localizar, sendo essa a base para o surgimento do problema do SLAM.

2.2. Mapeamento

A utilização de mapas é necessária para que um robô consiga construir um modelo do mundo ao seu redor e utilizar esse conhecimento para conseguir cumprir os seus objetivos. Para que um robô possa lidar com o mundo ao seu redor ele deve primeiramente ser capaz de perceber os objetos ao seu redor, preferencialmente antes de colidir com eles. Isso é feito através de sensores que interagem com o ambiente de alguma forma, como, por exemplo, explorando propriedades da luz (câmeras, laser, infravermelho) ou de ondas mecânicas (sonares). Conforme se movimenta pelo ambiente o robô realiza observações das estruturas que se encontram ao seu redor, e pode decidir por apenas reagir a elas de acordo com comportamentos programados (Balch, Arkin; 1993) ou então armazenar a informação contida nessas observações para que possam ser recuperadas posteriormente (Thrun; 2002a).

Esse armazenamento de informações é feito sob a forma de um mapa, que representa o modelo que o robô possui do mundo e armazena de maneira consistente as observações que ele coleta através dos seus sensores. Um mapa pode ser visto como um conjunto de variáveis (Pierce, Kuipers; 1994), cada uma capaz de descrever de maneira única uma determinada parcela de interesse do ambiente. Quando o robô realiza uma observação ele inicialmente deve determinar a qual região do ambiente essa observação está vinculada, e então atualizá-la no mapa de maneira que ela reflita o estado observado. Esse reconhecimento de regiões é feito a partir de um conjunto de características que cada variável possui e que a torna única dentro do mapa, cujo conteúdo depende do tipo de sensor e de mapa que está sendo utilizado. Caso uma região que ainda não possua uma variável correspondente seja observada essa variável é adicionada ao mapa, e será atualizada quando essa mesma região for observada novamente.

O posicionamento absoluto de uma estrutura $\boldsymbol{\theta_n^t}$ dentro do ambiente depende da localização $\boldsymbol{x_t}$ do robô, pois os sensores estão embarcados nele, e da informação contida na observação $\boldsymbol{z_k^t}$ realizada por esses sensores. De maneira similar ao que ocorre com a localização, os sensores responsáveis pelo mapeamento do ambiente não fornecem informações exatas, pois estão sujeitos a erros que não podem ser previstos ou modelados. Dessa forma, cada uma das informações incorporadas ao mapa possuirá um determinado valor de incerteza, que fará com que o mapa obtido seja apenas uma estimativa da realidade, e as suas variáveis se tornem variáveis aleatórias (Brezetz et al.; 1996). Essa imprecisão pode refletir de duas maneiras no mapa obtido: uma observação pode se correspondida com a variável errada ou então uma variável pode ser atualizada com um estado que não corresponde à realidade. Erros de correspondência podem ser evitados com a utilização de conjuntos de características mais robustos (Olson; 2002) e erros de atualização são corrigidos através do acúmulo de observações (Nieto et al.; 2003), que tende a fazer com que erros aleatórios sejam eliminados e o estado da variável convirja para o estado real.

Existem diversas maneiras de se construir o mapa de um ambiente, cada uma levando em consideração diferentes aspectos da informação coletada pelos sensores (Gutmann, Konolige; 1999). Mapas métricos e de características procuram criar uma representação espacial das estruturas ao redor do robô, distribuindo-as dentro de um sistema global de coordenadas. Mapas topológicos se baseiam na construção de grafos que conectam diferentes áreas do ambiente de uma maneira qualitativa (Choset, Nagatani; 2001). Esse trabalho usa representações espaciais para resolver o problema do SLAM, portanto a seguir são descritos em mais detalhes os mapas métricos e de características.

2.2.1. Mapas Métricos

Uma maneira de se armazenar as informações coletadas pelos sensores do robô é através dos chamados mapas métricos, que fazem uso de uma estrutura conhecida como grade de ocupação (Elfes; 1989). Uma grade de ocupação, aqui suposta bidimensional, pode ser modelada como uma matriz $\boldsymbol{M^t}$ onde cada célula $\boldsymbol{m^t_{x,y}}$ é uma variável que representa uma região discretizada e bem definida do ambiente e armazena o estado em que ela se encontra em um determinado instante t. Cada uma dessas variáveis é caracterizada pelas suas coordenadas no ambiente, e o seu estado indica a probabilidade daquela região estar vazia (pode ser atravessada pelo robô) ou ocupada (possui algum obstáculo que impede a navegação) dados os estados de localização $\boldsymbol{x_{0:t}}$ do robô e o conjunto de observações $\boldsymbol{Z_{0:t}}$ realizadas desde o início da navegação. Por convenção, m^t_{xy} indica o estado ocupado da variável e $\bar{m}^t_{xy}$ o estado livre.

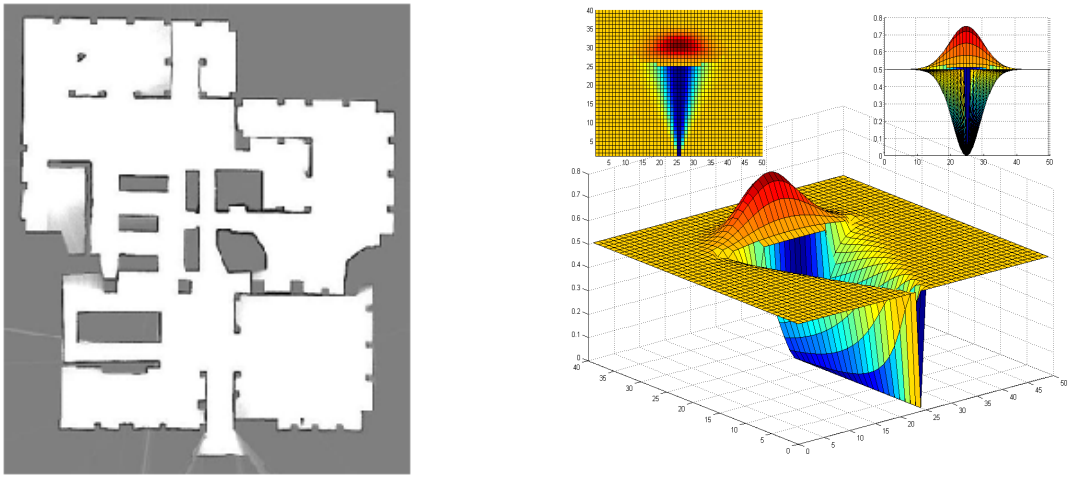
$$\boldsymbol{M^t} = \begin{Bmatrix} \boldsymbol{m_{0,0}} & \boldsymbol{m_{1,0}} & \cdots & \boldsymbol{m_{X-1,0}} & \boldsymbol{m_{X,0}} \\ \boldsymbol{m_{1,0}} & \boldsymbol{m_{1,1}} & \cdots & \boldsymbol{m_{X-1,1}} & \boldsymbol{m_{X,1}} \\ \vdots & & \ddots & \vdots & \\ \boldsymbol{m_{0,Y-1}} & \boldsymbol{m_{1,Y-1}} & \cdots & \boldsymbol{m_{X-1,Y-1}} & \boldsymbol{m_{X,Y-1}} \\ \boldsymbol{m_{0,Y}} & \boldsymbol{m_{1,Y}} & \cdots & \boldsymbol{m_{X-1,Y}} & \boldsymbol{m_{X,Y}} \end{Bmatrix} \qquad \begin{aligned} \boldsymbol{m^t_{xy}} &= \{\boldsymbol{\psi}\} \\ \boldsymbol{\psi^t_{xy}} &= p\left(\boldsymbol{m^t_{xy}} | \boldsymbol{z_{0:t}}, \boldsymbol{x_{0:t}}\right) \end{aligned} \tag{2.2}$$

No inicio da navegação cada uma das regiões apresenta um estado que indica o conhecimento que se possui em relação àquela região, e caso não haja nenhum conhecimento é atribuído o valor 0.5, indicando 50% de probabilidade de existir algum obstáculo. Conforme o robô se movimenta ele coleta informações a partir dos seus sensores e as utiliza para atualizar os estados das variáveis referentes às regiões observadas de maneira a fazê-los convergir para aqueles apresentados na realidade. Essa atualização é geralmente feita através do Teorema de Bayes (Murphy; 1999), um resultado da teoria de probabilidades que permite atualizar iterativamente a distribuição de probabilidades de uma variável aleatória conforme novas informações referentes a ela são obtidas. A formulação do Teorema de Bayes é apresentada em (2.3), onde $\boldsymbol{Z_t}$ indica o conjunto de observações $\boldsymbol{z^t_k}$ realizadas naquele instante e $\boldsymbol{x_t}$, a posição do robô naquele instante.

$$p\left(\boldsymbol{m^t_{xy}} | \boldsymbol{Z_t}, \boldsymbol{x_t}\right) = \frac{p\left(\boldsymbol{m^t_{xy}}\right) p\left(\boldsymbol{Z_t} | \boldsymbol{m^t_{xy}}, \boldsymbol{x_t}\right)}{p\left(\boldsymbol{m^t_{xy}}\right) p\left(\boldsymbol{Z_t} | \boldsymbol{m^t_{xy}}, \boldsymbol{x_t}\right) + p\left(\bar{\boldsymbol{m}}^{\boldsymbol{t}}_{\boldsymbol{xy}}\right) p\left(\boldsymbol{Z_t} | \bar{\boldsymbol{m}}^{\boldsymbol{t}}_{\boldsymbol{xy}}, \boldsymbol{x_t}\right)} \tag{2.3}$$

Como cada variável do mapa possui uma posição fixa no ambiente, a incerteza inerente às observações fornecidas pelos sensores faz com que seja atualizado um conjunto de variáveis,

de acordo com a probabilidade de cada uma delas representar a posição que foi efetivamente observada. Supondo que os sensores do robô conseguem detectar apenas regiões ocupadas e que os sinais enviados e recebidos viajam no espaço antes de interagir com os objetos detectados, pode-se considerar qualquer resultado de observação como ocupado e todo o espaço intermediário como livre. Na Figura 2.4b é apresentado um modelo probabilístico para um sensor de distância bidimensional, no caso o sonar. A região de maior probabilidade indica a posição da estrutura de acordo com a observação, e o espalhamento dessa probabilidade representa a incerteza quanto a essa estimativa. As regiões de menor probabilidade representam o caminho percorrido pelo sonar até alcançar o obstáculo, que se encontra vazio e por isso suas variáveis correspondentes podem ser atualizadas de acordo.



(a) (b)

Figura 2.4 - Mapas métricos.
(a) Grade de Ocupação Probabilística (b) Distribuição de Probabilidades

A vantagem da utilização de mapas métricos está no fato deles fornecerem uma representação densa do ambiente, o que facilita a utilização posterior de algoritmos como planejamento de trajetórias (Stentz; 1994) (Ersson, Hu; 2001) ou de exploração (Zelinsky; 1992). Contudo, esse tipo de mapa possui um elevado custo computacional, tanto no ponto de vista de processamento quanto de memória, pois é necessário armazenar e lidar com uma grande quantidade de informações. Adicionalmente, a caracterização de regiões é feita tendo como base apenas as suas coordenadas absolutas no ambiente, o que torna o reconhecimento bastante sensível a erros de mapeamento e localização, gerando uma grande quantidade de erros de correspondência. Dessa forma, algoritmos de SLAM que fazem uso de mapas métricos geralmente se limitam a utilizar a informação como foi obtida pelos sensores, sem o estabelecimento de marcos para auxiliar na navegação (Eliazar, Parr; 2003).

2.2.2. Mapas de Características

Outra maneira de se armazenar informações coletadas pelos sensores do robô é através dos chamados mapas de características, que filtram essas informações em busca de padrões relevantes de alguma forma para a aplicação onde esse mapa será utilizado e os armazenam sob a forma de pontos esparsos (Buschka; 2006), enquanto o resto é descartado. Assim sendo, um mapa é uma estrutura composta por um conjunto de variáveis, cada uma contendo um conjunto de características que a define de maneira única dentro do mapa e uma estimativa de posição que indica as suas coordenadas absolutas no ambiente (e a sua incerteza).

(2.4)

Essa abordagem gera um menor custo computacional de manutenção, pois lida com uma menor quantidade de informações (Figura 2.5a). Contudo, surge o custo decorrente da busca por padrões nas informações que sejam relevantes para a aplicação em questão, assim como a caracterização desses padrões de maneira a permitir a correspondência com estruturas semelhantes. Como as variáveis em um mapa de características não possuem posições fixas no espaço como acontece em um mapa métrico, a incerteza quanto a essa estimativa pode ser incorporada diretamente no parâmetro que a define, sob a forma de uma distribuição de probabilidades similar àquela utilizada no tratamento do problema de localização (Figura 2.5b). Supõe-se aqui que as estruturas que cada variável representa no ambiente não se movimentam, e por isso essa distribuição de probabilidades não se propaga no tempo, permanecendo constante uma vez definida.

(a) *(b)*

Figura 2.5 - Mapas de características (Montemerlo; 2003).
(a) Mapa resultante (árvores) (b) Distribuição de Probabilidades

No início da navegação o mapa se encontra vazio, pois o ambiente ao redor do robô é desconhecido, e conforme o robô se movimenta ele coleta informações com seus sensores em busca de padrões relevantes que possam ser incorporados ao mapa. Quando um padrão é encontrado, o seu conjunto de características $\boldsymbol{\psi_k^t}$ é comparado com as características $\boldsymbol{\psi_n^t}$ de cada uma das variáveis do mapa, em busca de padrões semelhantes. Se não houver correspondência supõe-se que o robô está observando uma região nova do ambiente, e esse padrão é armazenado sob a forma de uma nova variável, cuja posição absoluta é calculada a partir da informação fornecida pela observação e a estimativa de posição do robô naquele instante. Se houver correspondência essa informação é utilizada para refinar as estimativas de posição daquela variável e diminuir a sua incerteza inerente, aumentando com isso a precisão do mapa.

2.3. Algoritmos de SLAM

Como o próprio nome diz, o problema do SLAM surge quando ambos os problemas apresentados anteriormente precisam ser resolvidos simultaneamente, pois o robô não conta com informações precisas referentes a nenhum deles. Dessa forma, soluções individuais não são suficientes, pois os erros que cada estimativa possui se propagariam para a outra e rapidamente se acumulariam em um círculo vicioso. Um mapa impreciso não será capaz de gerar uma estimativa precisa de localização, que por sua vez aumentará a incerteza de mapeamento. É necessário lidar com ambos os problemas simultaneamente, procurando uma solução que minimize globalmente as incertezas envolvidas.

A primeira formalização matemática de uma solução para o problema do SLAM foi apresentada durante a década de 80 por uma série de artigos (Smith, Cheeseman; 1986) (Smith, Self, Cheeseman; 1987) (Durrant-Whyte; 1988) (Smith, Self, Cheeseman; 1990), e em 1989 foram realizadas as primeiras implementações dessa solução por Mourtarlier e Chatila. Contudo, trabalhos anteriores como os de (Taylor; 1976) e (Brooks; 1982) já haviam tentado quantificar as incertezas inerentes a um processo através de limiares mínimos e máximos, estabelecendo faixas de tolerância para uma dada operação e tentando controlar essa faixa para que ela se mantivesse estável. Em (Chatila, Laumond; 1985) foi determinado, para o robô HILARE, um

valor escalar representativo da incerteza da localização de um robô que leva em consideração tanto a sua posição quanto a sua orientação.

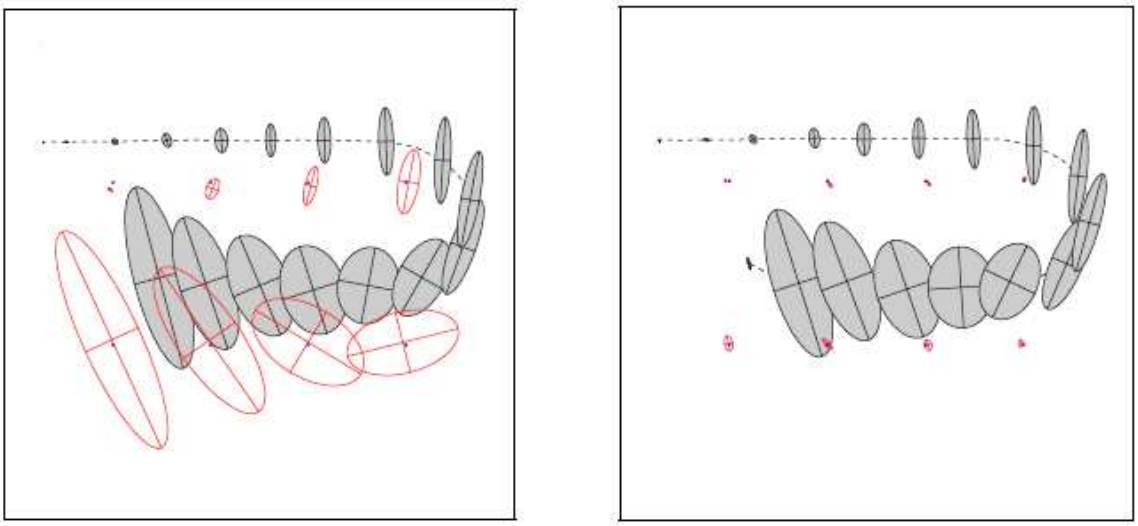
Essa solução para o problema do SLAM utiliza um filtro estocástico conhecido como Filtro Estendido de Kalman (Kalman; 1960), ou EKF, para acompanhar simultaneamente os estados de localização e mapeamento do robô, o que lhe rendeu o nome de EKF-SLAM ou SLAM Estocástico. A teoria de Filtros de Kalman fornece uma solução fechada e recursiva para o problema de se acompanhar uma variável aleatória de interesse conforme ela evolui no tempo (Simon, Uhlmann; 1997), e já era tradicionalmente utilizada em soluções individuais para o problema de localização e mapeamento. O EKF-SLAM procurou unificar esses dois problemas ao utilizar um mesmo EKF para acompanhar simultaneamente o estado de localização do robô e dos marcos que ele observa no ambiente. As Equações 2.5 e 2.6 exemplificam o vetor de estados $\boldsymbol{y_t}$ e a matriz de covariância $\boldsymbol{\Sigma}_t$ utilizados pelo EKF-SLAM para o caso da navegação bidimensional, onde a localização do robô pode ser descrita como $\boldsymbol{x_t} = \{x, y, \theta\}$ e a posição tridimensional dos marcos como $\boldsymbol{\theta_n^t} = \{x, y, z\}$.

$$\begin{aligned} y_t &= \{\boldsymbol{x_t} \quad \boldsymbol{\theta_1^t} \quad \boldsymbol{\theta_2^t} \quad \cdots \quad \boldsymbol{\theta_N^t}\}^T \\ &= \{x_t \quad y_t \quad \theta_t \quad x_1^t \quad y_1^t \quad z_1^t \quad x_2^t \quad y_2^t \quad z_2^t \quad \cdots \quad x_N^t \quad y_N^t \quad z_N^t\}^T \end{aligned} \tag{2.5}$$

$$\Sigma_t = \begin{bmatrix} \sigma(x_t,x_t)^2 & \sigma(x_t,y_t)^2 & \sigma(x_t,\theta_t)^2 & & \sigma(x_t,x_N^t)^2 & \sigma(x_t,y_N^t)^2 & \sigma(x_t,z_N^t)^2 \\ \sigma(y_t,x_t)^2 & \sigma(y_t,y_t)^2 & \sigma(y_t,\theta_t)^2 & \cdots & \sigma(y_t,x_N^t)^2 & \sigma(y_t,y_N^t)^2 & \sigma(y_t,z_N^t)^2 \\ \sigma(\theta_t,x_t)^2 & \sigma(\theta_t,y_t)^2 & \sigma(\theta_t,\theta_t)^2 & & \sigma(\theta_t,x_N^t)^2 & \sigma(\theta_t,y_N^t)^2 & \sigma(\theta_t,z_N^t)^2 \\ & \vdots & & \ddots & & \vdots & \\ \sigma(x_N^t,x_t)^2 & \sigma(x_N^t,y_t)^2 & \sigma(x_N^t,\theta_t)^2 & & \sigma(x_N^t,x_N^t)^2 & \sigma(x_N^t,y_N^t)^2 & \sigma(x_N^t,z_N^t)^2 \\ \sigma(y_N^t,x_t)^2 & \sigma(y_N^t,y_t)^2 & \sigma(y_N^t,\theta_t)^2 & \cdots & \sigma(y_N^t,x_N^t)^2 & \sigma(y_N^t,y_N^t)^2 & \sigma(y_N^t,z_N^t)^2 \\ \sigma(z_N^t,x_t)^2 & \sigma(z_N^t,y_t)^2 & \sigma(z_N^t,\theta_t)^2 & & \sigma(z_N^t,x_N^t)^2 & \sigma(z_N^t,y_N^t)^2 & \sigma(z_N^t,z_N^t)^2 \end{bmatrix} \tag{2.6}$$

No início da navegação o robô conta apenas com o seu estado de localização e a sua incerteza inerente, e conforme se movimenta essa incerteza tende a aumentar devido ao acúmulo de erros gerado pelo seu modelo de odometria. Ao observar um marco m_n^t o robô deve estimar a posição que ele ocupa no ambiente antes de adicioná-lo em seu mapa, e para isso deve utilizar a sua própria localização $\boldsymbol{x_t}$ e as informações fornecidas pela observação $\boldsymbol{z_k^t}$. Dessa forma, parte da incerteza referente à posição do marco será devido à incerteza de localização do robô e parte será devido aos erros aleatórios gerados pelos sensores que realizaram a observação. A matriz de covariância $\boldsymbol{\Sigma_t}$, mantida pelo EKF-SLAM, é capaz de armazenar essas relações de dependência entre variáveis através dos elementos não-diagonais que possui.

A Figura 2.6 mostra a evolução do EKF-SLAM conforme o robô se movimenta partindo da posição superior esquerda, indicando a sua posição e a dos marcos que ele observa conforme navega, assim como as suas incertezas correspondentes (elipses cinzas e vermelhas, respectivamente). Na Figura 2.6a o robô navega sem reconhecer marcos, apenas os armazenando em seu mapa, e como a incerteza referente à localização do robô aumenta com o passar do tempo o mesmo acontecerá com a incerteza de posicionamento dos marcos observados em instantes distintos, pois os erros de localização são propagados para o mapeamento e são responsáveis por parte dela. Os marcos são considerados imóveis, portanto suas incertezas, uma vez definidas, não aumentam com o passar do tempo.



(a) *(b)*

Figura 2.6 - Evolução do EKF-SLAM (Thrun, Fox, Burgard; 2005).
(a) Antes da atualização (b) Depois da atualização.

Na Figura 2.6b o robô observa novamente o primeiro marco, que possui uma incerteza pequena porque foi observado em um instante inicial de navegação, antes que os erros de odometria se acumulassem. A partir dessa informação o robô consegue uma nova estimativa de localização adicional àquela fornecida pelo seu modelo interno de movimentação, e pode utilizá-la para refinar esse estado e diminuir a sua incerteza de localização. Como a incerteza de todos os marcos observados até então dependiam em parte dessa incerteza, a sua diminuição permite que todas as estimativas de mapeamento também sejam atualizadas. O resultado é uma alteração que afeta todas as variáveis envolvidas no problema, tornando localização e mapeamento globalmente mais precisos. Trabalhos posteriores (Leonard, Durrant-Whyte; 1992) mostraram que essa dependência entre variáveis de localização e mapeamento é necessária para o tratamento completo do problema do SLAM.

Contudo, o EKF-SLAM possui uma grande limitação gerada justamente por essa característica, que é o seu custo quadrático em relação à quantidade de marcos devido à manutenção da matriz de covariância apresentada. Pode-se perceber em (2.5) que o tamanho do vetor de estados $\boldsymbol{y_t}$ utilizado pelo EKF-SLAM é igual a $3N + 3$, onde N é a quantidade de marcos armazenados no mapa. Isso faz com que $\boldsymbol{\Sigma_t}$ seja proporcional ao quadrado desse valor, e por isso operações feitas com ela terão um custo computacional $O(N^2)$, inviabilizando sua utilização em aplicações que possuam grandes quantidades de marcos devido ao tempo necessário para se realizar a atualização global de variáveis.

Esse é conhecido como o Dilema do EKF-SLAM (Thrun, Fox, Burgard; 2005), onde a maior vantagem dessa solução gera uma condição onde ela própria não pode ser utilizada. A atualização global que o EKF-SLAM proporciona é especialmente interessante em situações onde existem grandes quantidades de marcos, pois permite que a consistência entre estimativas seja mantida mesmo quando um determinado conjunto de marcos deixa de ser observado. Porém, o alto custo computacional necessário para que isso ocorra faz com que o EKF-SLAM só possa ser utilizado em aplicações onde isso não ocorre, minimizando o efeito positivo da atualização global. Aplicações reais do SLAM em navegação autônoma necessitam de grandes quantidades de marcos para conseguir resultados satisfatórios (Bailey; 2002), e evoluções nas técnicas de sensoriamento tornaram possível obtê-los (Csorba; 1997), tornando essa uma capacidade fundamental em algoritmos que se propõem a resolver esse problema.

Isso faz com que atualmente o EKF-SLAM possua um valor principalmente teórico, servindo como base para o surgimento de algoritmos que procuram explorar outras propriedades do problema de maneira a obter os mesmos resultados, ou aproximados, de maneira mais eficiente (Guivant, Nebot; 2001). Adicionalmente, em trabalhos recentes (Castellanos, Neira, Tardós; 2004) a consistência do EKF-SLAM em aplicações de navegação é questionada devido a fatores como a linearização que ele impõe sobre as funções utilizadas.

2.3.1. Fatoração

É notado em (Murphy; 1999) que o problema do SLAM, dado a localização do robô, pode ser decomposto de maneira exata em um conjunto de problemas menores de estimação que podem ser então resolvidos de maneira independente. O FastSLAM (Montemerlo; 2003) utiliza essa propriedade para, ao invés de lidar com uma única matriz de alta dimensionalidade, lidar

com um conjunto de matrizes menores que podem ser resolvidas eficientemente e com um escalonamento computacional menor. Essa solução faz com que a complexidade do problema do SLAM deixe de ser dependente do tamanho do mapa do ambiente, mas sim da quantidade de marcos observados a cada momento, que é aproximadamente constante durante toda a navegação. Perde-se com isso a capacidade de atualização global, mas a utilização de múltiplos mapas consegue simular esse fenômeno, como será mostrado adiante.

Ao invés de se lidar com cada marco individualmente pode-se também dividir o mapa construído pelo robô em um conjunto de mapas menores, que são atualizados localmente com um custo computacional menor. Soluções que utilizam essa abordagem são apresentadas em (Leonard, Feder; 1999), (Williams; 2001) e (Chong, Kleeman; 1999), conseguindo bons resultados localmente, mas com dificuldades em fechar *loops* muito grandes devido à divergência global gerada pelas atualizações locais. Algoritmos específicos para o problema do fechamento do *loop* (Kaess, Dellaert; 2005) são utilizados para suprir essa dificuldade, detectando quando o robô retornou a um ponto previamente visitado para somente então atualizar globalmente o mapa. Em (Dissanayake, Durrant-Whyte, Bailey; 2000) é apresentada uma maneira de se quantificar a quantidade de informação que cada parcela do mapa possui, o que permite eliminar aquelas menos representativas e diminuir a complexidade do mapa sem comprometer de maneira significativa os resultados finais.

2.3.2. Observações Negativas

O EKF-SLAM, da maneira como foi proposto, é incapaz de lidar com observações negativas, ou seja, a ausência de um marco que deveria ter sido observado, de acordo com o alcance e capacidade dos sensores, e que não foi. Essa capacidade é importante para a eliminação de "marcos espúrios" (Montemerlo; 2003), ou seja, marcos inexistentes que foram adicionados ao mapa ou que representam uma estrutura que já foi armazenada em um instante anterior. Abordagens que procuram minimizar o surgimento desses marcos lidam com as duas parcelas que definem a probabilidade de uma observação corresponder a um marco, que é a proximidade entre suas posições e a semelhança entre suas características. Intuitivamente, marcos mais distantes entre si diminuem a probabilidade de que ocorram correspondências falsas, assim como marcos mais distintos. Mas isso leva ao surgimento de mapas muito esparsos, pois

grande parte da informação coletada é descartada de forma a prevenir erros de correspondência, e aquela que resta muitas vezes não será suficiente para se resolver o problema do SLAM.

Técnicas que procuram lidar com observações negativas no EKF-SLAM de maneira a eliminar marcos espúrios adicionados ao mapa incluem a utilização de vetores de marcos temporários (Thrun, Fox, Burgard; 2005). Um marco temporário tem suas estimativas atualizadas, mas não interfere na estimativa de localização do robô (a dependência entre ambos é intencionalmente definida como nula). Quando ele é observado uma quantidade suficiente de vezes é promovido a um marco verdadeiro e passa a contribuir de maneira completa no processo. Outra maneira é a introdução de um índice que indica a probabilidade de existência de cada um dos marcos (Montemerlo; 2003). Sempre que ele é observado esse índice aumenta, e quando a sua estimativa de posição está dentro da área dos sensores e ele não é observado o índice diminui. Caso esse índice se torne menor do que um limiar o marco é considerado inexistente e é removido do mapa, assim como o vínculo que ele possui com outras variáveis.

(a) *(b)*

Figura 2.7 - Influência das observações negativas (Montemerlo; 2003).
O retângulo vermelho indica uma região que não possui marcos para detecção.
(a) Sem observações negativas (b) Com observações negativas

2.3.3. Associação de Dados

A atualização global que o EKF-SLAM impõe sobre as suas variáveis também o torna bastante sensível a correspondências erradas, que fazem com que as estimativas de localização e mapeamento se afastem da realidade ao mesmo tempo em que a incerteza referente a elas diminuem, dificultando a recuperação posterior do estado correto. Esse problema de relacionar

as observações realizadas pelo robô em um determinado instante com a informação armazenada em seu mapa é conhecido como o problema de associação de dados (Nieto et al.; 2003). Vários algoritmos de SLAM procuram resolvê-lo lidando com múltiplas hipóteses independentes de localização ou de mapeamento, mapeamento, sejam métricos como no caso do DP-SLAM (Eliazar, Parr; 2003) ou de características como no FastSLAM (Montemerlo et al.; 2002).

Ao se lidar com múltiplos mapas do ambiente, cada hipótese pode carregar consigo uma associação diferente de dados (Figura 2.8), criando ambigüidades que são mantidas até que sejam naturalmente resolvidas pela incorporação de novas informações. Quando isso acontece, os mapas que se mostram falsos são descartados sem nenhuma influência no resultado final. Técnicas eficientes de tratamento de informações eliminam a cópia redundante de informações e tornam possível lidar com milhares de mapas simultâneos do ambiente. Em (Nebot et al.; 2002) é mostrada uma abordagem que suspende a atualização do EKF-SLAM quando surge uma ambigüidade e institui um Filtro de Partículas vinculado a múltiplos mapas para lidar com o problema até que ele seja resolvido. Esse processo simula (Leonard et al.; 2002) o vetor temporário de marcos utilizado para diminuir o número de correspondências falsas, com a diferença de que não gera nenhum custo computacional extra, pois é resolvido naturalmente durante uma das etapas do Filtro de Partículas (a Reamostragem).

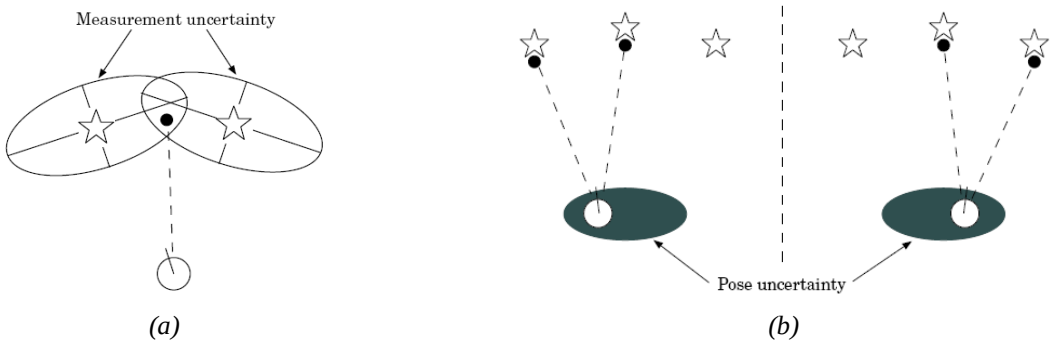


Figura 2.8 - Tratamento de ambigüidades em múltiplas hipóteses (Montemerlo; 2003). (a) Mapeamento (b) Localização

2.3.4. Offline SLAM

Outra limitação que a utilização de filtros traz é o descarte de informações que é feito depois que as mesmas são incorporadas, o que impede a utilização posterior caso necessário. Soluções *off-line* para o problema do SLAM, introduzidas em 1997 por Lu e Milios e

implementadas no mesmo ano por Gutmann e Nebel (1997), permitem a solução do SLAM de maneira que a informação de um determinado instante possa ser utilizada para refinar as estimativas de localização e mapeamento em todos os instantes anteriores. O problema se torna então um problema de otimização de todo o conjunto de dados, que procura minimizar a incerteza global do sistema (Duckett, Marsland, Shapiro; 2002). O GraphSLAM, proposto recentemente (Thrun, Fox, Burgard; 2005) (Thrun, Montemerlo; 2006), consegue remover dependências entre o robô e as posições dos marcos observados ao substituí-las por relações de dependência entre posições do robô em diferentes instantes, diminuindo o custo computacional da otimização resultante sem abrir mão do vínculo entre variáveis.

2.4. Marcos no SLAM

Para que um marco possa ser utilizado pelo robô como um ponto de referência que o auxilie na navegação, ele deve ser capaz de detectá-lo no ambiente e, depois, deve se capaz de reconhecê-lo de maneira inequívoca dentre outros marcos (Smith, Self, Cheeseman; 1990). A falha em detectar um marco diminui o número de referenciais que o robô utiliza para corrigir as suas estimativas, e a falha em reconhecer um marco pode gerar erros de correspondência que introduzem informações erradas nas estimativas de localização e mapeamento. Assim sendo, é fundamental que os sensores do robô sejam capazes de localizar uma quantidade suficiente de marcos no ambiente e que eles sejam distintos o suficiente para garantir uma correspondência robusta.

Soluções iniciais para o problema do SLAM com o uso de marcos utilizavam estruturas artificiais, adicionadas previamente ao ambiente e de fácil detecção. Esses marcos podem ser ativos, enviando sinais de informações diretamente para o robô (Kleeman; 1992) ou passivos, possuindo padrões facilmente reconhecíveis pelos sensores do robô (Brady et al.; 1987). Contudo, a introdução de marcos artificiais requer a alteração prévia do ambiente, algo que não é possível em todos os casos e que fere a premissa básica que é a navegação por regiões inicialmente desconhecidas. Ainda assim, essa abordagem permitiu o surgimento de robôs autônomos em ambientes controlados e previsíveis, como em linhas de montagem de fábricas (Hu, Gu; 2000), que podem ser modificadas sem grandes dificuldades e apresentarão sempre os mesmos tipos de movimentos.

Procurou-se então utilizar estruturas naturais ao próprio ambiente como marcos (Olson; 2002), de maneira a eliminar a necessidade de modificá-lo e ampliar a gama de situações reais onde o SLAM pode ser utilizado. A utilização de marcos naturais é limitada pela capacidade de caracterização do ambiente dos sensores do robô, que conseguem extrair apenas determinadas propriedades das estruturas ao seu redor. Sensores de distância conseguem fornecer apenas informações relativas à posição dos objetos em relação ao robô, geralmente em um único plano, o que dificulta a caracterização única dos objetos que o robô pode utilizar como marcos. Dessa forma, é possível obter apenas padrões simples de perfis pré-determinados, como postes (Press, Austin; 2004) e cantos (Altermatt et al.; 2004) em ambientes estruturados (Tardós et al.; 2001).

Sensores visuais conseguem, a partir das imagens fornecidas, uma caracterização muito mais rica das estruturas que formam o ambiente ao redor do robô (Shi, Tomasi; 1994). Essa caracterização permite a extração de marcos em ambientes não estruturados, seja através de um banco de dados pré-estabelecido composto por imagens de objetos que podem ser utilizados como marcos (Asmar, Zelek, Abdallah; 2006) ou pela determinação de estruturas subjetivas no ambiente que possuam alguma propriedade atrativa, como a facilidade de reconhecimento posterior sob diferentes pontos de vista (Lowe; 1999), que são então utilizadas como marcos (Jung; 2004). Essa capacidade faz com que sistemas de auxílio visual sejam especialmente úteis na resolução do problema do SLAM quando o robô navega por ambientes desconhecidos e não estruturados.

Devido a essas razões, esse trabalho procurou utilizar sensores visuais na solução do problema do SLAM, determinando com o auxílio de um sistema de visão omnidirecional os marcos que o robô utilizará para corrigir a sua localização durante a navegação. O próximo capítulo discutirá as dificuldades geradas por essa abordagem e os métodos já desenvolvidos que permitem a obtenção eficiente e robusta de marcos em imagens de maneira que possam ser utilizados por algoritmos de SLAM. É apresentado também o sistema de visão omnidirecional utilizado, assim como suas propriedades atrativas e limitações em relação ao problema que esse trabalho se propõe a resolver.

3. Sistemas de Visão

A utilização de imagens como uma forma de obtenção de informações do ambiente tem se tornado cada vez mais comum em tarefas de navegação autônoma, devido a fatores como o baixo custo e facilidade de implementação (Sujan, Meggiolaro, Belo; 2005). Contudo, o maior atrativo da utilização de sensores visuais está na riqueza de informações que eles são capazes de fornecer, qualitativa e quantitativamente. O rápido aumento no poder de processamento computacional faz com que seja possível atualmente lidar com toda essa informação, permitindo uma melhor compreensão do ambiente e aumentando a quantidade de dados que o robô dispõe para tomar decisões (Se, Lowe, Little; 2001).

Em (Kortenkamp, Bonasso, Murphy; 1998) é apresentada uma maneira de se combinar a informação de sensores visuais e de distância para estimar a posição de objetos observados, contornando uma limitação apresentada por sensores visuais que é a impossibilidade de se obter informações de distância com apenas uma imagem. Thrun (2001) descreve soluções estatísticas que procuram detectar a presença de objetos em movimento em imagens, o que permite o acompanhamento de alvos e o estabelecimento de marcos em ambientes dinâmicos que o SLAM pode utilizar (Avots, Thibaux, Thrun; 2002). A utilização de sensores visuais ativos (Davison, Murray; 1998) tem como base a determinação de trajetórias para o robô e orientação para os seus sensores de maneira a acelerar o processo de localização e mapeamento, diminuindo a complexidade computacional resultante.

Uma imagem é uma caracterização espacial densa do ambiente que define, de acordo com a resolução utilizada, cada uma das suas regiões com um parâmetro de cor que pode variar dentro de um intervalo de possibilidades. Essa caracterização densa e gradual permite o estabelecimento de uma grande quantidade de padrões bem definidos na imagem, cada uma representando uma propriedade do ambiente que o descreve de determinada maneira. Asada e Brady (1986) descrevem a curvatura de áreas na imagem em busca de padrões como retas, curvas e cantos, comuns em ambientes internos. O algoritmo de Hough (Duda, Hart; 1972), inicialmente utilizado para determinar segmentos de reta em imagens, foi posteriormente generalizado (Ballard; 1981) para detectar formas geométricas complexas que permitem parametrização. O SIFT utiliza espaços de escalas e orientações principais para encontrar regiões invariantes na imagem que possam ser correspondidas de maneira robusta em diferentes

pontos de vista. Uma extensa descrição de métodos de detecção de padrões no ambiente pode ser encontrada em (Tuytelaars, Mikolajczyk; 2006).

Ao observar uma determinada estrutura do ambiente o robô descreve e armazena as suas características relevantes em seu mapa, e espera-se que quando ele observar novamente essa mesma estrutura ele obtenha o mesmo conjunto de características. Essa semelhança permite que o robô corresponda essas observações e estabeleça um vínculo espacial e temporal entre as duas regiões, pois ambas representam uma mesma entidade no ambiente. Supondo que essa entidade não se movimentou no intervalo de tempo entre as duas imagens (ou que o seu deslocamento nesse intervalo é conhecido), a sua posição tridimensional pode ser calculada, refinando essa estimativa dentro do mapa de características. A partir desse momento essa estrutura se torna um marco no ambiente que o robô pode utilizar como ponto de referência durante a navegação para resolver o problema do SLAM (Prasser, Wyeth; 2003).

3.1. Características

Antes que uma imagem do ambiente possa ser utilizada pelo robô é necessário que ela seja processada de forma a extrair um determinado conjunto de informações que sejam relevantes e o auxiliem no cumprimento do seu objetivo. Cada um desses pedaços de informação extraídos da imagem recebe o nome de característica ("*feature*"), e pode representar (Figura 3.1) tanto uma propriedade global, como a sua intensidade média, quanto uma propriedade local restrita a uma certa área da imagem, como o gradiente entre dois pixels que pode representar uma região limite entre objetos. Como durante a navegação o robô precisa reconhecer objetos que compõem a imagem que ele possui do ambiente, nessas aplicações é mais comum a utilização de características locais, às quais esse trabalho se referirá apenas como características.

(a) *(b)* *(c)*

Figura 3.1 - Detecção de características.
(a) Imagem Original (b) Característica Global (contexto) (c) Características Locais (linhas)

Como definido em (Trucco, Verri; 1998) características representam propriedades significativas e detectáveis de uma imagem. Por significativas entende-se que elas estão associadas a processos relevantes na formação daquela imagem em específico e que contêm alguma característica importante segundo algum critério. Por detectáveis entende-se que elas devem poder ser localizadas dentro da imagem e se repetirão dadas as mesmas condições. No caso da navegação de robôs isso é desejável porque permite o reconhecimento de objetos no ambiente, algo necessário para tarefas como cálculo de distâncias por processamento estéreo ou determinação de marcos que o robô pode utilizar para se localizar durante a navegação.

A utilização de características em tarefas de processamento de imagens teve origem em trabalhos realizados por Moravec (Moravec; 1979), que criou o termo pontos de interesse ("*interest points*") para definir os pixels que desempenhavam papéis importantes na composição da imagem a partir das suas relações com seus vizinhos. Com o passar do tempo o termo ponto de interesse perdeu força diante da denominação de característica ("*feature*") porque essas estruturas possuem uma dimensão espacial, ou seja, cobrem uma determinada região da imagem, não sendo restritas a um único pixel. Adicionalmente, múltiplas características podem se sobrepor, compartilhando informações de uma mesma região da imagem.

De acordo com (Tuytelaars, Mikolajczyk; 2006), uma característica ideal deve possuir as seguintes propriedades:

- **Repetibilidade**: Alterações nas condições de observabilidade de um objeto, como mudanças de luminosidade ou de ponto de vista, não devem alterar as características detectadas. Pode ser obtida de duas maneiras:
    - **Invariância**: As alterações sofridas pela imagem são modeladas de forma a permitir uma compensação antes das características serem detectadas.
    - **Robustez**: Diminuição da rigorosidade na detecção das características, o que pode comprometer a precisão.
- **Distinção**: Os padrões de diferentes características devem apresentar uma alta variação, para permitir uma correspondência única.
- **Localidade**: Sua representação deve tomar uma área pequena da imagem, de forma a não sofrer oclusão ou representar múltiplos objetos simultaneamente.
- **Quantidade**: Deve existir em grandes quantidades.
- **Precisão**: Os padrões que a definem devem ser precisamente determinados.
- **Eficiência**: Sua detecção deve ser rápida.

Na prática, uma característica nunca será ideal, e diferentes métodos de detecção encontrarão conjuntos que satisfazem em maior ou menor grau cada um desses requisitos. Na realidade, alguns desses requisitos são mutuamente excludentes, como é o caso de distinção e localidade, pois quanto mais local a característica é menos informação ela possuirá para a etapa de correspondência. Valores elevados de invariância também diminuem a distinção, pois uma parte da informação adquirida é extraída para gerar as transformações que levam à invariância. Uma alta distinção, por sua vez, diminui a repetibilidade, pois uma característica muito específica possui uma menor probabilidade de ser correspondida com outra.

3.2. Descritores

Depois que uma característica é localizada na imagem as suas propriedades devem ser modeladas de alguma forma, para que ela possa ser armazenada e utilizada posteriormente na correspondência com outras características. Essa modelagem é feita através de um descritor, que idealmente possui propriedades semelhantes às de uma característica (Tuytelaars, Mikolajczyk; 2006):

- **Repetibilidade**: Dadas duas características com propriedades similares, os seus descritores resultantes também devem ser similares entre si.
- **Distinção**: O descritor deve conseguir armazenar informação suficiente para ser capaz de distinguir entre duas características diferentes.
- **Compactação**: Informações redundantes ou altamente correlacionadas não devem ser armazenadas, visando com isso poupar memória e tempo de processamento durante a correspondência.
- **Eficiência**: As informações contidas em um descritor devem ser armazenadas de maneira eficiente, permitindo um fácil acesso e utilização.

Da mesma forma que diferentes características capturam diferentes propriedades da imagem, diferentes descritores são capazes de capturar diferentes propriedades de uma única característica. A maneira como uma característica é descrita é tão importante quanto a maneira como é obtida, pois é a partir dessa descrição que ela será armazenada e utilizada posteriormente. Assim sendo, existem diversos tipos diferentes de descritores, cada um procurando melhores resultados diante de certas situações, como é o caso da utilização de histogramas (Lazebnik,

Schmid, Ponce; 2003) (Lowe; 2004), equações diferenciais (Koenderink, Doorn; 1987) ou transformações para o espaço de freqüências (Bigun, Buf; 1992). Em geral (Lowe; 1999), um descritor é representado por um vetor numérico onde cada um de seus coeficientes indica uma propriedade independente referente à característica que está sendo descrita.

3.3. Correspondência

Como dito anteriormente, a função de um descritor é armazenar as propriedades principais de uma característica de forma a permitir a sua correspondência posterior com características obtidas em outras imagens. Supondo que as estruturas existentes no ambiente mantêm as suas propriedades com o passar do tempo, pode-se dizer que as características obtidas a partir dessa estrutura também possuirão propriedades semelhantes. Essa semelhança entre dois vetores descritores $\boldsymbol{\psi^m}$ e $\boldsymbol{\psi^n}$ pode ser calculada pela distância euclidiana d entre ambos no espaço k-dimensional de acordo com (3.1), onde K representa a dimensão do vetor descritor. Quanto menor for essa distância, maior será a probabilidade de ambos representarem uma mesma estrutura no ambiente.

$$d(\boldsymbol{f_m}, \boldsymbol{f_n}) = \sqrt{\sum_{i=0}^{K} (\psi_{\mathrm{i}}^m - \psi_{\mathrm{i}}^n)^2} \tag{3.1}$$

Um descritor é usualmente avaliado de acordo com dois critérios. O primeiro deles é a taxa de correspondência entre as características de um mesmo objeto quando visto de maneiras diferentes (Mikolajczyk, Schmid; 2002), como em diferentes pontos de vista ou luminosidade. O segundo critério é a taxa de acertos conseguida por um classificador em um conjunto de imagens, com base em um banco de dados que contém um modelo das características referentes às possibilidades de classificação (Kadir, Brady, Zisserman; 2004). Diferentes aplicações requerem descritores com melhor desempenho em diferentes critérios, e existem situações onde um bom desempenho não é visto como uma qualidade. O SLAM, por exemplo, se beneficia mais de descritores capazes de reconhecer objetos sob diferentes pontos de vista, enquanto a classificação de acordo com o tipo de objeto pode gerar correspondências erradas em ambientes com objetos similares.

3.4. Triangulação

A única informação espacial que uma imagem consegue fornecer em relação a cada uma de suas características são as coordenadas $\boldsymbol{p}$ do pixel que elas ocupam. Essa informação, juntamente com a geometria da câmera e do sistema de visão, permite calcular o raio de luz que refletiu na estrutura observada e gerou aquele pixel, restringindo a sua posição dentro do ambiente a uma linha. Não é possível restringir essa posição a um único ponto, e por isso não há como obter informações de distância a partir de uma única imagem. Para isso é necessário que uma mesma estrutura seja observada a partir de dois pontos de vista diferentes, ou seja, que esteja presente em duas imagens obtidas a partir de posições distintas e conhecidas.

Essa configuração pode ser devido à utilização de múltiplas câmeras em posições diferentes do robô (Murray, Little; 2000) ou à utilização de uma única câmera que obtém múltiplas imagens em instantes distintos de navegação (Gaspar; 2002). Com isso torna-se possível, através da triangulação (Gluckman, Nayar, Thorek; 1998), encontrar o ponto onde ambas as linhas se cruzam, cada uma tendo como origem a posição do robô no instante em que a imagem foi obtida, e assim determinar tridimensionalmente a estrutura no ambiente. A Figura 3.2 mostra um exemplo de triangulação realizada com uma única câmera deslocada no ambiente, saindo do ponto 1 e seguindo até o ponto 2 com um deslocamento δ_{12}.

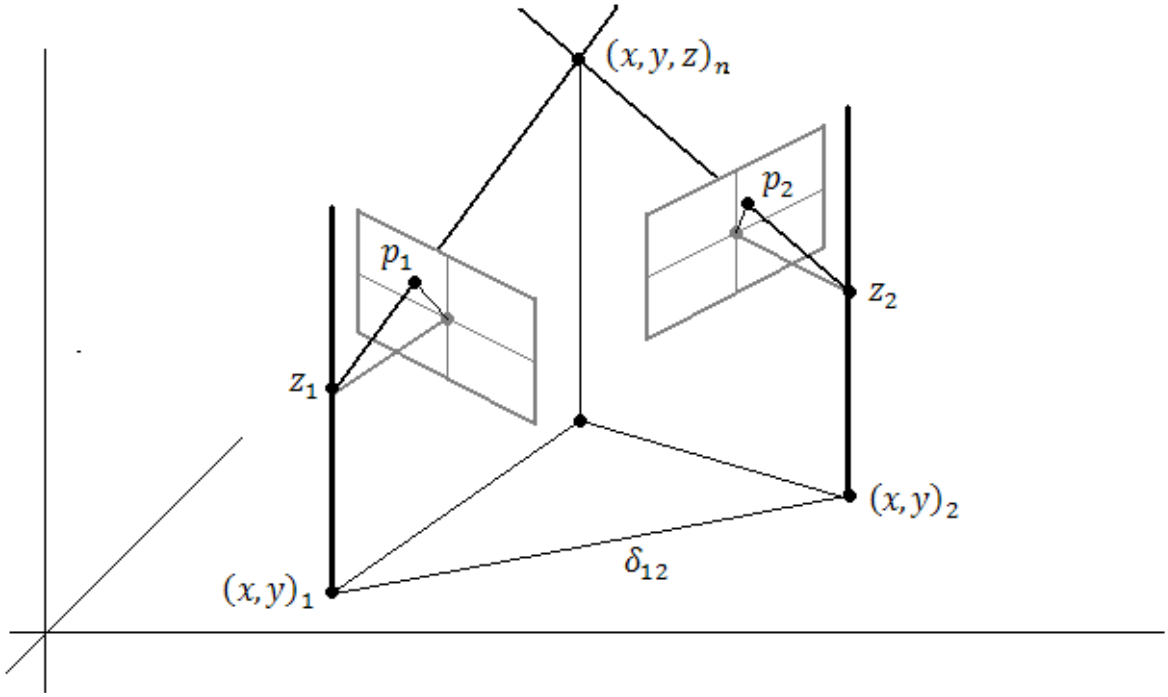


Figura 3.2 - Triangulação com uma única câmera deslocada no ambiente.

3.5. Marcos Visuais

A grande maioria dos algoritmos de SLAM que utilizam sistemas visuais como forma de se obter informações do ambiente fazem uso das próprias características obtidas da imagem como marcos (Davison; 2003) (Kwok, Dissanayake; 2004) (Se, Lowe, Little; 2002). Diferentes métodos de se extrair características viáveis para a utilização como marcos incluem o SIFT (Lowe; 2004), Detectores de Harris (Harris, Stephens; 1988) e regiões normalizadas de alto contraste (Shi, Tomasi; 1994). Em (Mikolajczyk, Schmid; 2002) é realizada uma extensa pesquisa referente a métodos de extração de características em imagens e a sua utilização como marcos.

A utilização de características da imagem diretamente como marcos permite uma elasticidade muito maior nos algoritmos resultantes, pois não é necessário estabelecer nenhuma restrição quanto ao tipo de marco que será utilizado, ele é definido dinamicamente sem nenhum vínculo semântico. Por outro lado surge o problema de escalabilidade, pois conforme os ambientes se tornam maiores a quantidade de marcos cresce rapidamente, tornando o processo computacionalmente inviável. Em (Jung; 2004) é apresentada uma maneira de se agrupar diversas características em um único marco, ao invés de utilizá-las individualmente, o que contribui para a diminuição do custo computacional final. É importante perceber que mesmo assim não há uma informação semântica que os relacione entre si, apenas propriedades inerentes à imagem na qual foram obtidos, o que permite a aplicação do algoritmo em diferentes tipos de ambiente.

Outra abordagem consiste na incorporação de semântica às características utilizadas, definindo aquelas que o algoritmo deve utilizar como marcos durante a navegação. Em (Panzieri, Pascucci, Ulivi; 2001) é apresentado um algoritmo de SLAM que utiliza como marcos fontes de luz artificiais circulares posicionadas no teto, segmentando-as para a detecção e reconhecimento. Marcos verticais são utilizados em (Kim, Sukkarieh; 2003) para localização e mapeamento na navegação de aviões não tripulados, cuja posição é determinada pela projeção dessas estruturas nas imagens obtidas. Árvores são utilizadas como marcos por (Asmar, Zelek, Abdallah; 2006) e postes em (Fitzgibbons; 2004), se aproveitando da organização natural que certos tipos de ambientes (como parques e ruas) tendem a possuir. Embora essa abordagem consiga resultados mais consistentes, os algoritmos obtidos estão limitados às regiões para as quais foram desenvolvidos, o que compromete a sua generalidade.

3.6. Visão Omnidirecional

Sensores de visão omnidirecional representam uma família de sensores visuais que são capazes de obter simultaneamente informações referentes a todo o ambiente ao redor da câmera (Grassi Jr., Okamoto Jr.; 2006). Além da utilização de câmeras verdadeiramente omnidirecionais (Nalwa; 1996), existem várias outras maneiras de se obter essa propriedade através de arranjos especiais. Por exemplo, câmeras rotatórias ou múltiplas câmeras podem obter diversas imagens que são então combinadas para formar uma única imagem omnidirecional (Peleg, Ben-Erza; 1999). Espelhos especiais podem refletir a luz de todo o ambiente na câmera, conseguindo com isso imagens omnidirecionais com câmeras comuns (Baker, Nayar; 1998). Diferentes tipos de geometria de espelho, como esférica, cônica, parabólica ou hiperbólica produzem resultados diferentes nas imagens obtidas.

Dentre essas possibilidades, os sistemas omnidirecionais obtidos com a utilização de espelhos são mais compactos e por isso mais indicados para serem embarcados em robôs móveis. Dentre as geometrias de espelho possíveis a hipérbole possui a propriedade de foco único de projeção, o que facilita a construção de espelhos e permite a utilização de câmeras comuns na construção do sistema de visão omnidirecional (Svoboda, Pajdla; 2002). Espelhos elípticos devem ser côncavos para apresentarem essa mesma propriedade, o que dificulta a sua construção. Espelhos parabólicos não podem ser utilizados com câmeras convencionais, pois redirecionam a luz paralelamente, sem um foco único de projeção.

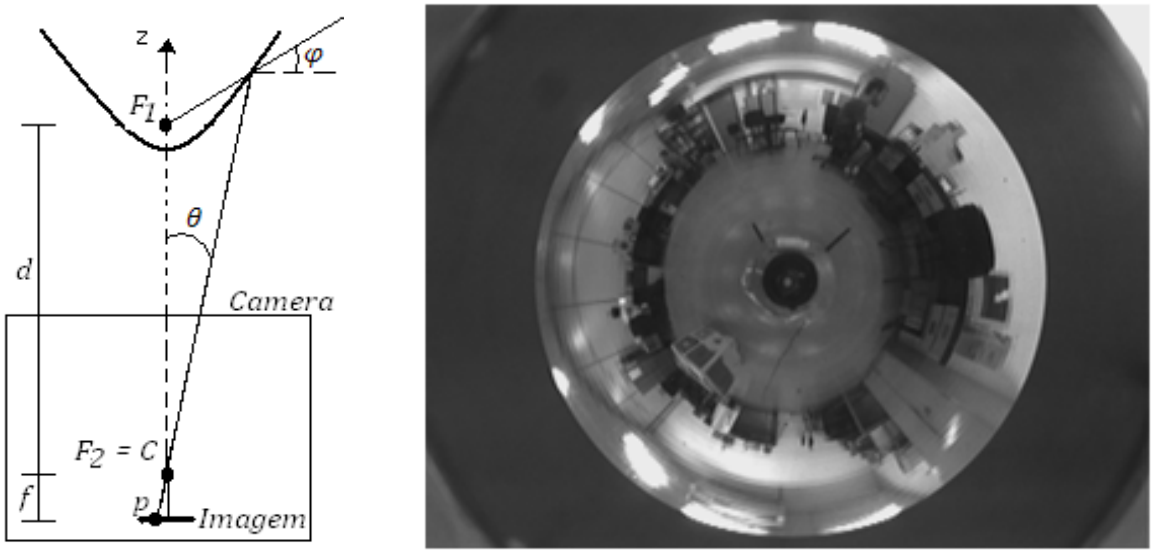


Figura 3.3 - Imagens omnidirecionais com espelhos hiperbólicos.
(a) Esquema de montagem (b) Exemplo de imagem

A Figura 3.3b mostra um exemplo de imagem omnidirecional obtida através do espelho hiperbólico esquematizado na Figura 3.3a. A câmera é montada na vertical e aponta para o espelho, que se encontra acima dela, fixo em um suporte a uma distância que faz com que o seu foco inferior F_2 coincida com o foco C da câmera. A luz do ambiente é refletida pelo espelho e enviada para a câmera, que a transforma na imagem omnidirecional. A posição do pixel p na imagem define o ângulo θ entre o raio de luz e a vertical, que por sua vez define o ângulo φ entre a horizontal e o raio de luz incidente no espelho, e com isso qual região do ambiente o pixel representará. Conforme θ aumenta o mesmo acontece com φ, e caso φ se torne positivo aquele pixel estará observando o infinito (ou o objeto mais próximo naquela direção).

Como se pode perceber pela Figura 3.3a, para $\theta = 0$ temos $\varphi = -\pi/2$, o que já era esperado. Conforme θ aumenta, no início o valor de φ em pouco se altera, pois o espelho hiperbólico possui uma curvatura pequena em sua região central. Contudo, quando θ começa a atingir as regiões mais externas do espelho essa curvatura se acentua cada vez mais (Figura 3.4), e o valor de φ se torna extremamente sensível a qualquer variação. Esse efeito é responsável pela diminuição da resolução nas regiões mais distantes do centro de uma imagem omnidirecional. Nessas regiões um pixel pode representar uma área de dezenas de metros, fazendo com que qualquer imprecisão na localização de uma característica dentro da imagem crie um erro substancial nas estimativas de posição obtidas com essa informação.

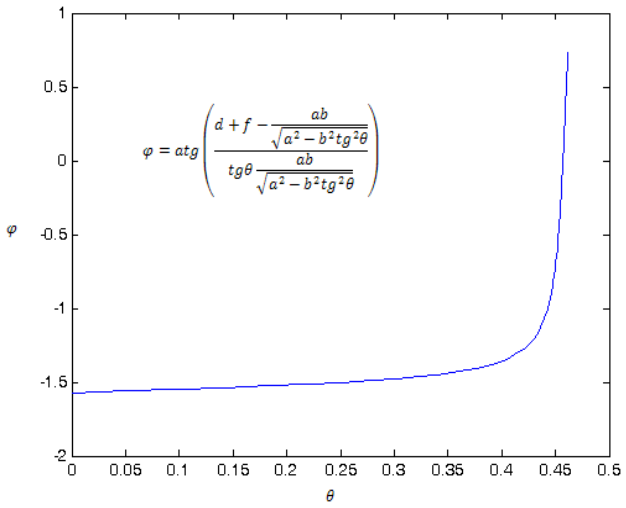


Figura 3.4 - Relação entre θ e φ.

Outra limitação enfrentada pelas imagens omnidirecionais construídas dessa forma é a deformação que os seus objetos sofrem, devido à projeção da superfície do espelho em uma imagem bidimensional. Grande parte das técnicas desenvolvidas pela computação visual leva

em consideração a geometria convencional e por isso não apresentam mesma qualidade nos resultados quando aplicadas em geometrias diferentes como a omnidirecional. Assim, é comum que imagens omnidirecionais sejam retificadas antes do processamento visando eliminar deformações, por meio de técnicas como a utilização de câmeras virtuais (Akihiko, Imiya; 2004).

Contudo, mesmo diante dessas limitações os sensores de visão omnidirecional ainda são amplamente utilizados em uma grande gama de aplicações, inclusive na área de navegação autônoma. Algoritmos eficientes de triangulação capazes de fornecer informações de distância dos objetos em relação ao robô são apresentados em (Zhu; 2001), permitindo a construção de mapas tridimensionais densos do ambiente (Ikeda, Miura; 2006) e o posicionamento dos marcos utilizados pelo SLAM. O amplo campo de visão que sensores omnidirecionais apresentam permite a detecção e reconhecimento simultâneos de um maior número de marcos, e por estarem mais afastados entre si esses marcos permitem uma triangulação mais precisa.

4. Solução Proposta

Esse trabalho propõe a utilização de um sensor de visão omnidirecional para a extração de marcos naturais no ambiente que possam ser utilizados como pontos de referência por um robô de maneira a auxiliar a sua navegação por ambientes desconhecidos. A utilização de sensores de visão omnidirecional gera um conjunto de benefícios que tem sido explorado em diversos trabalhos. Em (Andreasson, Duckett; 2004) esse tipo de sensor é utilizado na solução do problema de localização global em mapas topológicos devido à quantidade de marcos que podem ser obtidos com uma única imagem, o que permite uma melhor caracterização de cada região do ambiente. Um processo similar foi descrito em 1993 por Horswill, onde cada imagem omnidirecional é caracterizada globalmente e utilizada como um único marco. Em (Goedemé et al.; 2007) é utilizado o mesmo princípio para se resolver o problema da localização incremental do robô, aproveitando-se do amplo campo de visão que uma imagem omnidirecional proporciona e que faz com que cada marco observado permaneça visível por um intervalo de tempo maior durante a navegação.

O problema do mapeamento é abordado em (Wahlren, Duckett; 2005), onde imagens do ambiente são obtidas e armazenadas juntamente com o vínculo estabelecido entre suas características correspondidas. O mapa em si só é construído quando necessário, permitindo com isso o acúmulo de observações antes que sejam tomadas decisões de associação de dados e aumentando a qualidade do fechamento de *loops*. Kim e Chung (2003) apresentam uma solução para o problema do SLAM que utiliza dois sistemas de visão omnidirecional para obter imagens em pontos de vista distintos sem a necessidade de movimentação prévia do robô. Em trabalhos anteriores (Guizilini, Okamoto Jr.; 2005) o autor explorou diferentes geometrias de espelhos visando conseguir o mesmo efeito com apenas uma imagem.

Procurou-se no trabalho aqui desenvolvido criar uma estrutura que tirasse o máximo proveito das propriedades inerentes a esse tipo de sensor. A detecção e reconhecimento de características nas imagens omnidirecional obtidas é realizada de acordo com os critérios de um algoritmo conhecido como SIFT (Lowe; 2004). Em (Mikolajczyk, Schmid; 2002) é realizado um estudo que o elegeu o SIFT como o algoritmo mais estável para a obtenção de marcos devido a propriedades das suas características como invariância a escala e rotação. Essa invariância é importante no reconhecimento de objetos sob diferentes pontos de vista, con-

forme o robô navega pelo ambiente. Os marcos utilizados são naturais ao próprio ambiente e nenhuma suposição foi feita quanto ao tipo de estruturas que o compõem, mantendo o princípio de que o robô não deve conhecer o ambiente ao seu redor no início do processo.

A informação obtida a partir do sensor de visão omnidirecional é incorporada às estimativas de localização e mapeamento de acordo com um algoritmo de SLAM conhecido como FastSLAM (Montemerlo; 2003). O FastSLAM mantém um Filtro de Partículas para lidar com a localização e um mapa de características para lidar com o mapeamento, tratando cada um dos marcos independentemente. O problema da associação de dados é tratado com a utilização de múltiplos mapas, tornando o FastSLAM robusto a eventuais correspondências falsas, algo comum na utilização de sensores visuais devido à grande quantidade de informação que deve ser processada. Técnicas eficientes de manutenção e acesso a informação permitem a utilização simultânea de milhares de marcos mantendo um desempenho em tempo real, e é proposto e descrito aqui uma modificação na implementação tradicional do FastSLAM que permite lidar com grandes quantidades de marcos de maneira ainda mais eficiente.

No início da navegação o robô está localizado em um ponto arbitrário do ambiente e o seu mapa se encontra vazio, indicando o seu desconhecimento em relação aos objetos ao seu redor. Ao se movimentar, a sua localização é atualizada pelo seu sistema de odometria e o conjunto de partículas se espalha por uma determinada área, indicando o aumento da imprecisão de localização (etapa PF: Predição). É obtida uma imagem do ambiente, que é processada pelo SIFT em busca de marcos que o robô pode utilizar como pontos de referência. Esses marcos são adicionados de maneira independente aos mapas correspondentes a cada uma das partículas, gerando as múltiplas hipóteses de localização e mapeamento (etapa EKF). Como é utilizada apenas uma câmera, é impossível obter diretamente informações de posição, e os marcos são inicialmente adicionados sem essa informação. Ao invés disso, são armazenadas as coordenadas da característica na imagem e a localização do robô naquele instante.

Posteriormente, conforme novas imagens são obtidas, os marcos extraídos podem ser comparados com aqueles já existentes no mapa, e caso haja correspondência é possível triangular as duas orientações e posições, e com isso estimar a posição do marco observado (etapa EKF). A partir desse ponto, novas observações referentes a esse marco gerarão informações relativas não apenas à sua posição, permitindo refinar ainda mais essa estimativa, mas também à posição do robô naquele instante. Torna-se possível então utilizar essa informação para alterar a distribuição de probabilidades das partículas que acompanham a localização do robô no ambi-

ente (etapa PF: Atualização). Partículas que possuem uma hipótese de mapeamento similar àquela observada pelo robô naquele instante recebem um peso maior e por isso mesmo terão uma maior probabilidade de serem mantidas e duplicadas durante a PF: Reamostragem (quando ela se mostra necessária). Essa localização é propagada para o próximo instante, reiniciando o processo iterativo do algoritmo.

A Figura 4.1 apresenta um esquema do algoritmo proposto nesse trabalho, mostrando os diferentes componentes utilizados e a função de cada um deles. Em seguida, cada uma das etapas envolvidas é descrita conceitualmente, assim como a maneira como se comunicam de forma a permitir que os resultados desejados sejam alcançados.

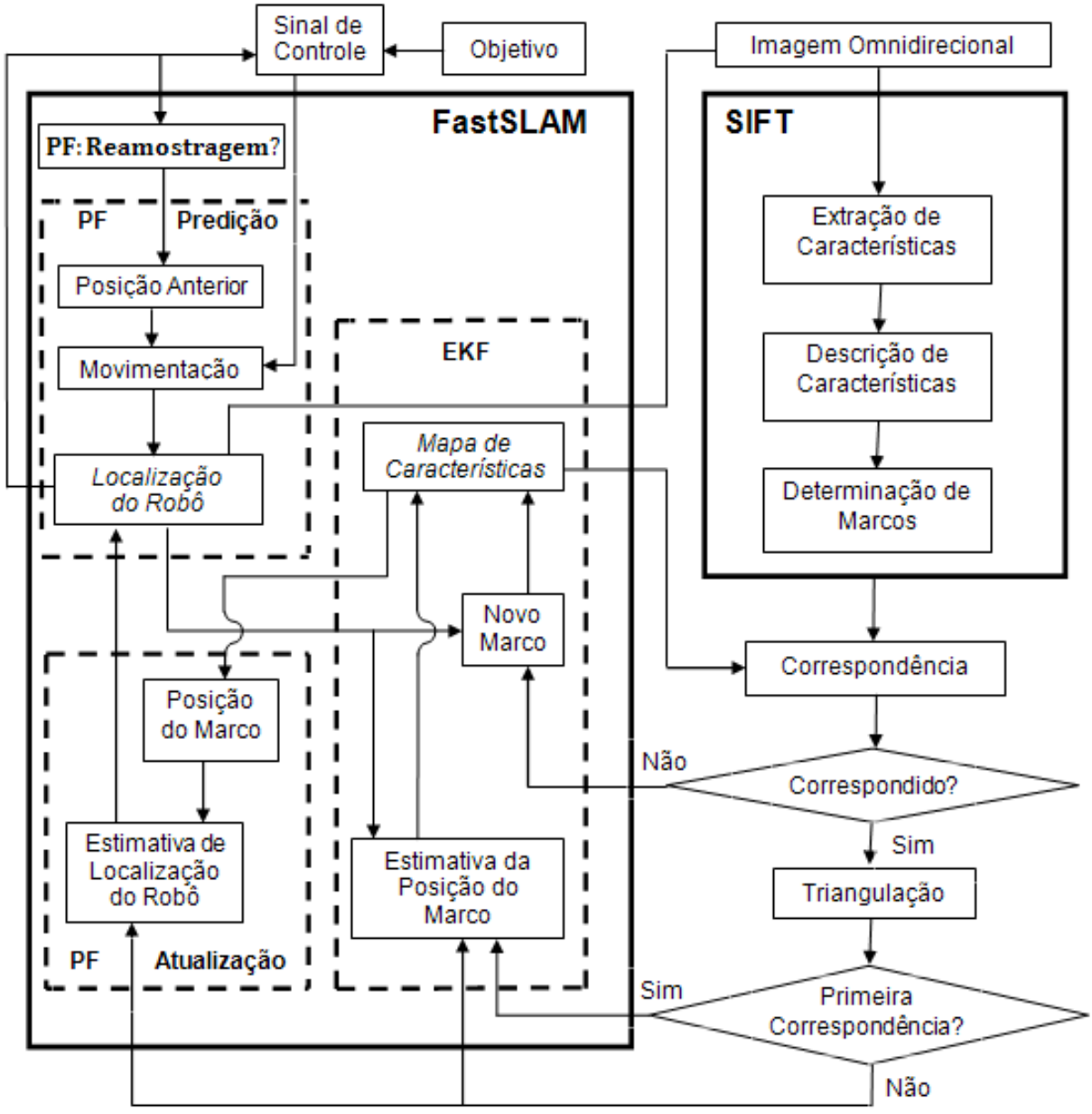


Figura 4.1 - Diagrama do algoritmo proposto.

4.1. SIFT

O SIFT (*Scale Invariant Feature Transform*) foi introduzido por Lowe em 1999 como um algoritmo de reconhecimento de objetos e tem evoluído constantemente em trabalhos posteriores (Brown, Lowe; 2002) (Lowe; 2004), além de encontrar aplicações em diversas outras áreas, entre elas o próprio SLAM (Se, Lowe, Little; 2001) (Ledwich, Williams; 2004). A busca pelas características é realizada em diversos espaços de escala e freqüência, garantindo uma distribuição bastante homogênea que reduz os efeitos negativos causados por oclusões ou ruídos. A quantidade de características obtida em cada imagem é bastante elevada e pode ser controlada a partir da calibração de diversos parâmetros, gerando combinações mais eficientes em determinadas situações. Essas características podem ser utilizadas diretamente (Se, Lowe, Little; 2001) ou então reunidas, semanticamente (Panzieri, Pascucci, Ulivi; 2001) ou não (Jung; 2004), para formar os marcos que o robô utilizará na navegação.

O algoritmo de SIFT utilizado nesse trabalho tem como entrada uma imagem omnidirecional obtida a partir do sistema de visão embarcado no robô, e é armazenada também a estimativa de posição do robô no instante em que a imagem foi obtida, para que posteriormente possa ser feita uma triangulação caso haja correspondência entre marcos. Essa imagem omnidirecional passa pelas três etapas do SIFT e, como resultado, é obtido um conjunto de características que representam os marcos detectados na imagem naquele instante.

4.1.1. Extração de Características

No algoritmo de SIFT a extração de características é realizada em etapas visando uma maior eficiência computacional. Inicialmente os pixels da imagem são percorridos em diversos espaços de escala e aqueles que apresentarem um extremo de intensidade (mínimo ou máximo) são armazenados como candidatos a características. Em teoria esses pixels já poderiam ser utilizados, pois apresentam as propriedades que definem uma característica de acordo com o SIFT. Porém, muitos desses pixels são instáveis e sensíveis a ruídos, dificultando a correspondência e diminuindo a qualidade do resultado final. Assim sendo, esse conjunto inicial de características é filtrado de acordo com dois critérios diferentes, e aqueles que falharem em pelo menos um deles são eliminados do conjunto.

4.1.1.1. Detecção de Extremos

A primeira etapa na extração de características em um algoritmo de SIFT consiste na busca por pontos de extremo de intensidade dentro de diversas escalas possíveis, o que é feito gerando-se uma estrutura conhecida como espaço de escalas (Witkin; 1983), responsável pela invariância a alterações de escala que as características resultantes apresentam. Mostra-se (Lindeberg; 1993) que, sob um conjunto razoável de suposições, o único espaço de escalas possível é aquele gerado pela função gaussiana. Assim sendo, cada imagem em uma escala é definida como uma função $L(i,j,\sigma)$ gerada pela convolução de uma máscara gaussiana $G(i,j,\sigma)$ com uma imagem $I(i,j)$.

$$L(i,j,\sigma) = G(i,j,\sigma) * I(i,j) \tag{4.1}$$

Onde $G(i,j,\sigma)$ é o operador gaussiano bidimensional com média μ nula e matriz de covariância Σ diagonal com autovalores iguais a σ. Em (Lowe; 1999) é proposta a utilização das diferenças de gaussianas $D(i,j,\sigma)$ na composição do espaço de escalas de onde serão extraídos os primeiros candidatos a características.

$$\begin{aligned} D(i,j,\sigma) &= \big(G(i,j,k\sigma) - G(i,j,\sigma)\big) * I(i,j) \\ &= L(i,j,k\sigma) - L(i,j,\sigma) \end{aligned} \tag{4.2}$$

Essa função é escolhida devido à sua facilidade de obtenção (a convolução gaussiana precisa ser calculada de qualquer forma para o cálculo dos descritores que o SIFT utiliza) e também porque apresenta uma aproximação para o laplaciano gaussiano $\sigma^2\nabla^2 G$. Em (Lindeberg; 1994) é mostrado que a invariância total quanto a mudanças de escala é obtida através da normalização do laplaciano pelo fator σ^2. A relação entre esse laplaciano e a função $D(i,j,\sigma)$ pode ser obtida a partir da equação de difusão de calor (parametrizada em termos de σ ao invés da forma mais usual $t = \sigma^2$).

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G \tag{4.3}$$

Uma aproximação linear da derivada parcial em (4.3) nos leva a:

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(i,j,k\sigma) - G(i,j,\sigma)}{k\sigma - \sigma} \tag{4.4}$$

Que pode ser reescrito como:

$$G(i,j,k\sigma) - G(i,j,\sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{4.5}$$

Mostra-se com isso que a utilização de diferenças gaussianas na determinação de características invariantes a alterações de escala já incorpora o fator σ^2 de normalização proposto por Lindeberg. A constante $(\mathrm{k}-1)$ não se altera durante a construção do espaço e por isso não afeta a detecção de extremos. O erro de aproximação diminui conforme k se aproxima da unidade, mas testes empíricos mostrados em (Lowe; 2004) mostraram não haver um impacto significativo nos resultados. A Figura 4.2a mostra uma maneira eficiente de se construir o espaço de escalas de diferenças de gaussianas.

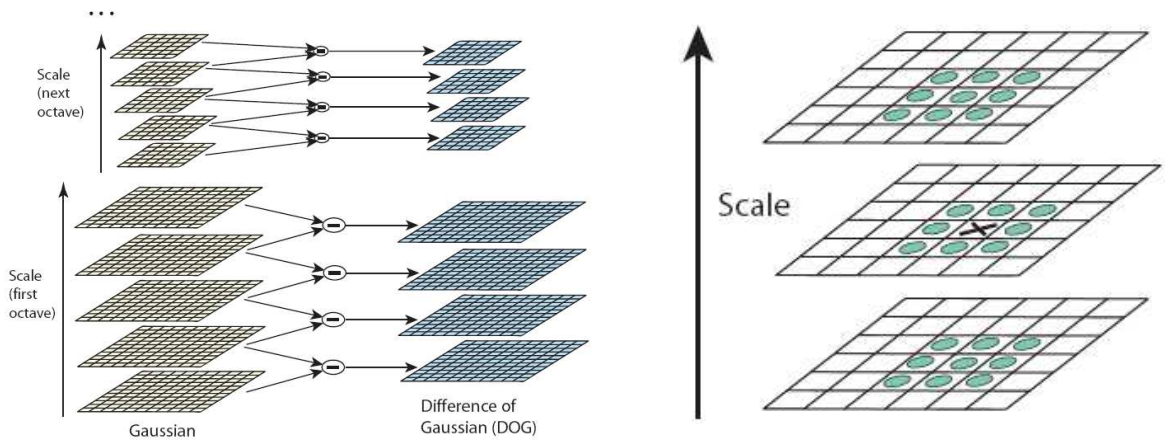


Figura 4.2 - Espaço de escalas (Lowe; 2004).
(a) Construção (b) Detecção de Extremos

Cada uma das imagens utilizadas na construção do espaço de escalas é obtida através da convolução da imagem inicial com uma função gaussiana de desvio padrão σ multiplicado pelo coeficiente k, que aumenta seguindo uma progressão geométrica até que o desvio padrão efetivo $k\sigma$ dobre, completando com isso uma oitava. Dessa forma, para uma oitava composta por N imagens o coeficiente k de cada uma delas será igual a $2^{n/N}$. Quando uma oitava é completada, a próxima é construída tendo como base a última imagem da oitava anterior, amostrada de forma a eliminar as linhas e colunas ímpares (ou pares). Essa amostragem diminui o tamanho das imagens da próxima oitava por um fator de 4 (e o desvio padrão efetivo pela metade), diminuindo significativamente o custo computacional da construção das próximas oitavas sem interferir de maneira significativa no resultado final. Durante a construção da primeira oitava pode-se ainda dobrar as dimensões da imagem (interpolando os pixels criados) de maneira a procurar por extremos em escalas maiores.

Com as imagens gaussianas calculadas, as diferenças de gaussiana são obtidas diretamente pela subtração de uma imagem com a sua anterior, formando uma pirâmide de diferenças gaussianas. Um extremo é encontrado pela comparação de um pixel com os seus vizinhos, levando em consideração também as imagens superior e inferior (Figura 4.2b). Caso esse pixel seja um máximo ou um mínimo, ele representa um extremo e se torna um candidato a característica $\boldsymbol{f_m} = \{i, j, o, n\}$, com $\{i, j\}$ sendo as coordenadas do pixel na imagem e $\{o, n\}$ a oitava à qual a imagem pertence e a sua posição dentro dela.

4.1.1.2. Eliminação por Contraste

O primeiro filtro utilizado na eliminação de características instáveis se baseia no contraste entre pixels, eliminando aqueles que possuem um baixo contraste em relação à sua vizinhança. Implementações iniciais do SIFT (Lowe; 1999) utilizavam diretamente a posição do candidato a característica na imagem em que ele foi extraído. Em trabalhos posteriores (Brown, Lowe; 2002) mostrou-se que interpolar os pontos próximos para se localizar a verdadeira posição do extremo gera um aumento significativo na qualidade dos resultados, especialmente nas áreas de baixa resolução que imagens omnidirecionais apresentam. Essa interpolação é feita através de uma expansão de Taylor centrada em $\boldsymbol{f_m}$, com $\Delta\boldsymbol{f_m}$ representando a distância entre a posição real do extremo e $\boldsymbol{f_m}$. Como nenhuma diferença de gaussianas é construída utilizando-se duas oitavas diferentes (Figura 4.2a), não é necessário derivar parcialmente em relação a essa coordenada.

$$D(\Delta\boldsymbol{f_m}) = D(\boldsymbol{f_m}) + \frac{\partial D(\boldsymbol{f_m})^T}{\partial \boldsymbol{f}}\Delta\boldsymbol{f_m} + \frac{1}{2}\Delta\boldsymbol{f_m}^T\frac{\partial^2 D(\boldsymbol{f_m})}{\partial \boldsymbol{f}}\Delta\boldsymbol{f_m} \tag{4.6}$$

As derivadas parciais referentes a cada coordenada podem ser aproximadas pela diferença entre pixels vizinhos naquela direção, como mostrado abaixo.

$$\frac{\partial D(\boldsymbol{f_m})}{\partial f} = \begin{bmatrix} \frac{\partial D(\boldsymbol{f})}{\partial i} \\ \frac{\partial D(\boldsymbol{f})}{\partial j} \\ \frac{\partial D(\boldsymbol{f})}{\partial n} \end{bmatrix}_{\boldsymbol{f}=\boldsymbol{f_m}} = \frac{1}{2}\begin{bmatrix} D(i+1,j,n) - (i-1,j,n) \\ D(i,j+1,n) - (i,j-1,n) \\ D(i,j,n+1) - (i,j,n-1) \end{bmatrix}_{\boldsymbol{f}=\boldsymbol{f_m}} \tag{4.7}$$

Derivando (4.6) e igualando a zero, encontra-se o valor do offset Δf_m.

$$\Delta f_m = -\frac{\partial^2 D(\boldsymbol{f_m})}{\partial \boldsymbol{f}^2}^{-1} \frac{\partial D(\boldsymbol{f_m})}{\partial \boldsymbol{f}} \tag{4.8}$$

Se o offset Δf_m de alguma coordenada for maior do que 0.5 o extremo estará mais próximo de outro ponto da mesma imagem ou das imagens vizinhas no espaço de escalas. O valor de f_m é então alterado para refletir a nova posição, e a interpolação é realizada novamente agora em relação a esse novo ponto. Quando o valor convergir, esse offset pode ser utilizado para determinar o contraste $D(\Delta f_m)$ de f_m através da substituição de (4.8) em (4.6).

$$|D(\Delta \boldsymbol{f_m})| = \left| D(\boldsymbol{f_m}) + \frac{1}{2} \frac{\partial D(\boldsymbol{f_m})^T}{\partial \boldsymbol{f}} \Delta \boldsymbol{f_m} \right| > C \tag{4.9}$$

Onde C é um limiar predeterminado de contraste que elimina os candidatos que falharem na comparação como sendo instáveis para serem utilizados como características de acordo com o SIFT.

4.1.1.3. Eliminação por Curvaturas

A detecção de extremos realizada pelo espaço de escalas de diferenças gaussianas tende a produzir muitos candidatos ao longo de bordas, mesmo quando essas regiões são instáveis. Em visão computacional, bordas são definidas como regiões na imagem que possuem uma diferença acentuada entre as suas duas curvaturas principais (Trucco, Verri; 1998). Assim sendo, o segundo filtro utilizado pelo SIFT procura eliminar candidatos a características que possuem uma relação entre curvaturas principais elevada.

As curvaturas principais de um pixel podem ser obtidas através do cálculo dos autovalores da sua matriz Hessiana H_m, mostrada a seguir (4.10). Como nesse filtro não há transição entre imagens, não é necessário utilizar as coordenadas n de escala e o de oitava, que serão omitidas para simplificar a notação.

$$H_m = \begin{bmatrix} D_{ii} & D_{ji} \\ D_{ij} & D_{jj} \end{bmatrix} \tag{4.10}$$

Assim como foi feito anteriormente, as derivadas parciais podem ser calculadas pela diferença de intensidade dos pixels vizinhos naquela direção.

$$\begin{aligned} D_{ii} &= \frac{D(i+1,j) - D(i-1,j)}{2} \\ D_{jj} &= \frac{D(i,j+1) - D(x,y-1)}{2} \\ D_{ij} = D_{ji} &= \frac{D(i+1,j+1) + D(i-1,j-1) - D(i+1,j-1) - D(i-1,j+1)}{4} \end{aligned} \tag{4.11}$$

Como apenas a proporção entre autovalores é relevante, não há a necessidade de calculá-los explicitamente (Harris, Stephens; 1988). Pode-se obter a soma deles pelo traço da Hessiana $tr(H_m)$ e o produto pelo determinante da Hessiana $det(H_m)$.

$$tr(H_m) = D_{ii} + D_{ij} = \alpha + \beta \tag{4.12}$$

$$det(H_m) = D_{ij}D_{jj} - D_{ij}^2 = \alpha\beta \tag{4.13}$$

O determinante de H_m deve ser positivo porque de outra forma o ponto teria sido descartado por não representar um extremo. Definindo a relação entre curvaturas como um valor r de forma que $\alpha = r\beta$, temos:

$$\frac{tr(H_m)^2}{det(H_m)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{4.14}$$

Que já não depende dos valores individuais de curvatura, apenas da proporção entre elas. Quando $r = 1$ as duas curvaturas serão iguais, e quanto maior r maior será a diferença entre elas. Assim sendo, a eliminação de candidatos a feature com alta relação de curvaturas pode ser feita pela seguinte comparação:

$$\frac{tr(H_m)}{det(H_m)} < \frac{(R+1)^2}{R} \tag{4.15}$$

Onde *R* é um limiar máximo de relação entre curvaturas que uma característica pode possuir para ser considerada estável de acordo com esse critério. Implementações tradicionais do SIFT utilizam um limiar máximo igual a 10 e nesse trabalho percebeu-se que em imagens omnidirecionais deve-se utilizar um valor próximo de 2 para que seja mantida a mesma taxa de eliminação de candidatos, devido à maior diferença entre curvaturas que esse tipo de imagem tende a possuir.

4.1.2. Cálculo do Descritor

O vetor descritor do SIFT é baseado em um histograma de orientações, que define a maneira como os pixels ao redor da característica se relacionam entre si. A cada orientação $\theta(i,j)$ é atribuído um valor de magnitude $m(i,j)$ que indica a influência que ela possui sobre o resultado final, e para obter invariância rotacional o SIFT institui uma orientação e magnitude principais para cada característica, e então monta o seu vetor descritor relativamente a esses valores. Assim, uma característica rotacionada apresentará a mesma distribuição, e por isso poderá ser correspondida.

$$\theta(i,j) = atan\left(\frac{L(i,j+1) - L(i,j-1)}{L(i+1,j) - L(i-1,j)}\right) \tag{4.16}$$

$$m(i,j) = \sqrt{\big(L(i+1,j) - L(i-1,j)\big)^2 + \big(L(i,j+1) - L(i,j-1)\big)^2} \tag{4.17}$$

Para se obter a orientação principal de uma característica uma janela é posicionada ao seu redor e os valores de orientação e magnitude de cada um dos pixels nessa região são calculados. Os valores de magnitude são então posicionados em um histograma composto pelas orientações calculadas, como mostrado na Figura 4.3. Para evitar instabilidades causadas por alterações bruscas em regiões distantes da característica, as magnitudes obtidas são ponderadas por uma distribuição gaussiana centrada nas coordenadas da característica e com desvio padrão igual àquele utilizado na suavização gaussiana da imagem de onde a característica foi obtida.

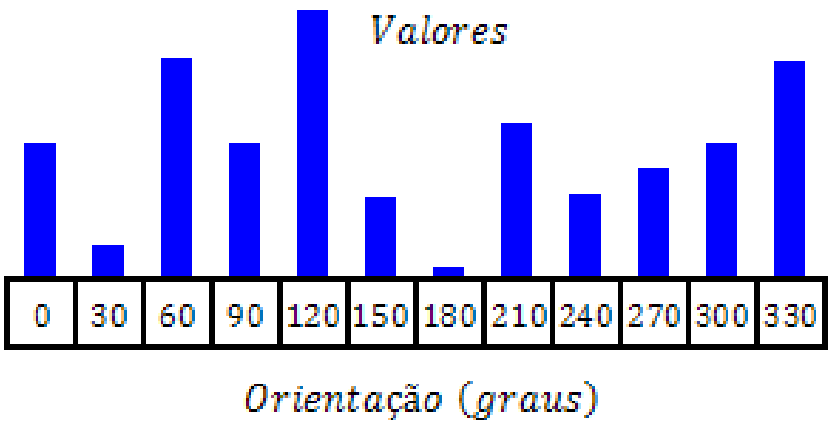


Figura 4.3 - Histograma de orientação.

O ponto mais alto do histograma será a orientação principal da característica, e caso haja outro pico com valor suficientemente elevado (Lowe propõe um limiar de 80% em relação ao principal) outra características é criada com as mesmas propriedades e apresentando outra orientação principal. Assim, um mesmo pixel pode possuir diversas orientações vinculadas a características distintas, o que contribui para uma melhor representação da imagem e correspondências mais robustas.

A construção do vetor descritor é feita posicionando-se outra janela ao redor da característica, que é dividida em subjanelas de acordo com a complexidade do vetor descritor final desejado. A orientação e magnitude dos pixels pertencentes a cada janela são calculadas e para cada subjanela é montado um histograma que armazena esses valores, onde as orientações são discretizadas também de acordo com a complexidade desejada do vetor descritor final. A Figura 4.4 apresenta um exemplo da construção do vetor onde são consideradas 4 subjanelas e 8 valores diferentes de orientação (o círculo azul indica a distribuição gaussiana utilizada). Novamente é utilizada uma suavização gaussiana (círculo azul) visando diminuir a influência de alterações bruscas em regiões distantes da característica

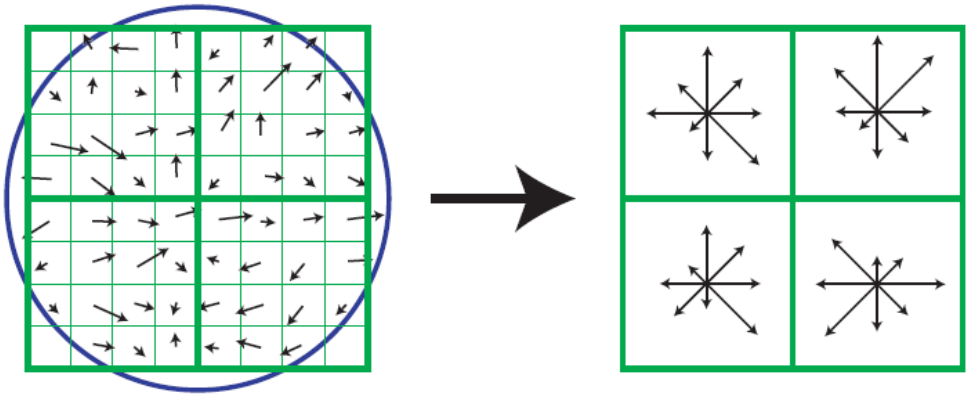


Figura 4.4 - Construção do descritor (Lowe; 2004).

O vetor descritor armazena a magnitudes referentes a cada uma das orientações, rotacionadas de acordo com a orientação principal calculada anteriormente. No exemplo da Figura 4.4 o vetor descritor final possuirá 36 posições, enquanto implementações tradicionais do SIFT costumam apresentar descritores compostos por 128 valores, obtidos em 16 histogramas com 8 orientações cada um. Embora essa quantidade seja alta em relação a outros descritores, ela permite uma alta taxa de correspondência em objetos que foram submetidos a deformações, fazendo com que o SIFT apresente certo grau de invariância em também em relação a essa transformação.

Para se obter invariância relativa à luminosidade as magnitudes contidas no descritor são normalizadas, pois dessa forma alterações globais de luminosidade não afetarão os resultados obtidos. Contudo, variações não-lineares podem ocorrer devido a efeitos de saturação da câmera, o que interfere na distribuição das magnitudes, mas não afeta de forma perceptível as orientações. Assim, uma invariância parcial quanto a essas alterações pode ser obtida limitando-se a influência das magnitudes mais elevadas, saturando o seu valor em um determinado limiar. Depois disso o descritor é novamente normalizado, e estará completo para ser utilizado na correspondência entre características.

4.1.3. Determinação dos Marcos

Uma vez que as características da imagem já tenham sido extraídas e descritas, é necessário que elas sejam transformadas nos marcos que o robô utilizará para resolver o problema do SLAM durante a navegação. Uma abordagem possível consiste na utilização direta das características extraídas como marcos (Se, Lowe, Little; 2002), utilizando o vetor descritor como o conjunto de propriedades que define o marco e permite a sua correspondência posterior. Caso haja uma correspondência, a posição da característica no ambiente pode ser estimada por triangulação, completando o conjunto de propriedades que um marco deve possuir para ser utilizado pelo SLAM da maneira como proposto aqui.

Contudo, essa abordagem tende a gerar uma quantidade muito grande de marcos, pois todas as características extraídas das imagens coletadas são adicionadas ao mapa de características para serem utilizadas pelo SLAM. Uma grande parte dessas características pode ser considerada espúria, por não representar nenhum objeto consistente no ambiente, e por isso não serão estáveis para serem correspondidas em imagens posteriores. Adicionalmente, a correspondência entre características individuais pode gerar uma grande quantidade de erros, devido a fatores como similaridade entre objetos no ambiente. Essas correspondências erradas podem propagar informações incorretas para as estimativas de localização e mapeamento e aumentar a imprecisão dos resultados.

Uma alternativa já discutida é a atribuição de relações semânticas aos objetos correspondidos, de maneira a utilizar apenas características que possuam um determinado conjunto de propriedades pré-estabelecidas. Mas essa abordagem vai contra a proposta desse trabalho, que é a navegação em qualquer tipo de ambiente. Por isso, optou-se aqui pela determinação de con-

juntos não semânticos de características (Jung; 2004), de acordo com as relações que elas apresentam dentro da imagem de onde foram extraídas. Foram escolhidos dois critérios diferentes para a determinação desses conjuntos, a distância entre características na imagem e a diferença de contraste entre elas. Supõe-se aqui que características que representem uma mesma estrutura estarão próximas entre si e possuirão um contraste similar na imagem.

Inicialmente a posição de cada característica é comparada com a posição das outras extraídas da imagem, e aquelas que estiverem suficientemente próximas de acordo com algum limiar pré-estabelecido de distância seguem para a segunda comparação, que é a diferença de contraste que elas apresentem. Caso também estejam dentro do limiar de contraste elas são reunidas com a característica inicial e o processo é repetido com elas, até que não reste nenhuma característica que satisfaça ambos os critérios. Se a quantidade de características reunidas dessa forma for maior do que um limiar de tamanho estabelecido ele é considerado um "objeto" no ambiente e essas características serão utilizadas como marcos nessa iteração do algoritmo, caso contrário são eliminadas. Dessa forma a quantidade de marcos gerada a cada instante diminui consideravelmente, mas aqueles gerados serão mais estáveis nas etapas posteriores.

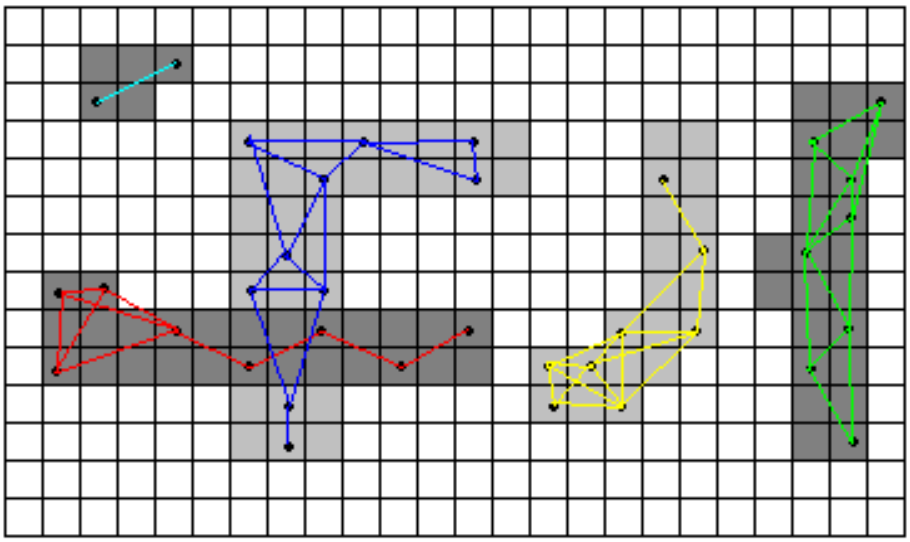


Figura 4.5 - Conjuntos de características. Os pontos pretos indicam as características que foram localizadas na imagem (limiar: 3 pixels).

As características referentes a cada objeto observado são armazenadas independentemente no mapa de característica do robô, cada uma possuindo o seu próprio valor de incerteza e de posicionamento. O índice do objeto ao qual pertencem é mantido de forma a permitir o estabelecimento de vínculos durante a etapa de correspondência, como mostrado a seguir.

4.2. Correspondência

A correspondência entre características é realizada pela comparação entre os valores contidos em seus vetores descritores em busca daqueles que se encontrarem mais próximos no espaço *k*-dimensional. Uma característica será correspondida com outra quando a distância euclidiana (3.1) entre ambas for a menor possível dado o conjunto de características possíveis da imagem a qual cada uma delas pertence. Para aumentar a confiabilidade dos resultados são descartadas as correspondências onde a proporção entre a menor distância euclidiana obtida e a segunda for maior do que um determinado limiar. Dessa forma são eliminadas correspondências ambíguas, como é o caso daquelas extraídas do plano de fundo da imagem, e que por isso não são estáveis como marcos.

Contudo, o custo computacional gerado por uma busca exaustiva pela melhor correspondência entre dois conjuntos de características é elevado devido à alta dimensionalidade que os vetores devem possuir para conseguir descrever de maneira satisfatória cada característica. Outro fator limitante é a grande quantidade de características que uma única imagem tende a possuir, o que aumenta o espaço de possibilidades de correspondência. Algoritmos eficientes de busca, como as *kd-tree* (Friedman, Bentley, Finkel; 1977) não apresentam uma melhora significativa para casos com dimensões elevadas, portanto não solucionam esse problema. Em (Beis, Lowe; 1997) é proposta uma solução probabilística para o problema da correspondência, que consegue encontrar as características com menor distância euclidiana em aproximadamente 95% das tentativas com um ganho computacional de duas ordens de grandeza (Figura 4.6), chamado BBF (Best Bin Fit).

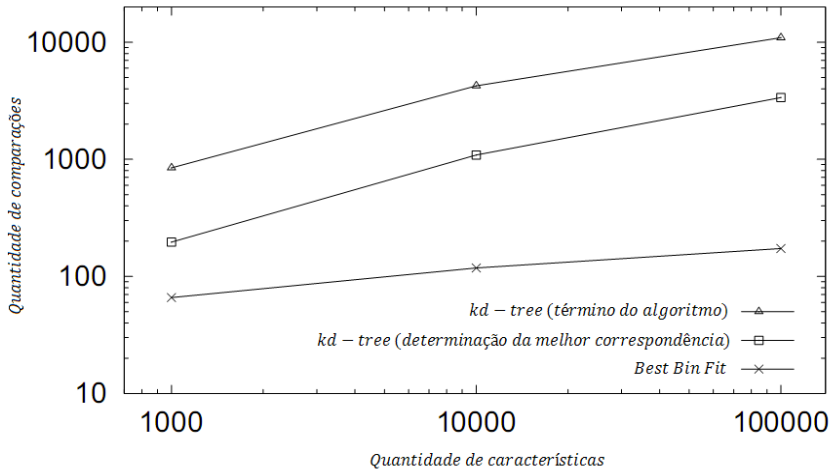


Figura 4.6 - Comparação entre kd-tree e Best Bin Fit (Arya, Mount; 1993).

O BBF mantém uma versão modificada da *kd-tree*, que organiza os componentes do vetor descritor de cada característica de forma que valores semelhantes fiquem próximos entre si. Quando uma busca é realizada são verificadas primeiramente as distâncias referentes a determinadas dimensões do vetor (Arya, Mount; 1993), e a busca é interrompida quando aquela região de características se mostra distante daquela que está sendo correspondida. Essa busca probabilística gera um ganho computacional de duas ordens de magnitude e gera resultados não ótimos em aproximadamente 5% das correspondências. O BBF se torna especialmente eficiente em casos onde características ambíguas são descartadas, como é o caso daquelas geradas pelo plano de fundo da imagem (Lowe; 1999).

Ainda assim, a comparação individual entre características pode ocasionar erros de correspondência, especialmente em ambientes estruturados que tendem a possuir repetição de padrões, como portas em um corredor e árvores em um parque. Dessa forma, esse trabalho apresenta e utiliza uma segunda etapa na correspondência entre características, que leva em consideração os conjuntos estabelecidos durante a determinação de marcos. Inicialmente as características de cada um dos objetos armazenados no mapa são correspondidas com as características extraídas da imagem, em busca de semelhanças que indiquem o reconhecimento de estruturas. Em seguida é calculada a porcentagem de correspondências que foram feitas em relação à quantidade de características que o objeto possui, e caso esse valor esteja abaixo de um limiar essas correspondências são ignoradas. A probabilidade de que um conjunto de características seja corespondido de maneira errada é menor do que a probabilidade de apenas uma o seja, o que aumenta a confiabilidade dos resultados finais e diminui consideravelmente o custo computacional do processo.

Adicionalmente, dessa forma é possível instituir uma maneira de se eliminar marcos considerados espúrios, de acordo com a influência que eles apresentam durante a correspondência de seus objetos. Caso um objeto seja consistentemente reconhecido, mas uma de suas características não o seja, ela pode ser removida do mapa de maneira a diminuir a quantidade de marcos com os quais o robô deve lidar. Se isso fizer com que a quantidade de características do objeto caia abaixo de um limiar estabelecido, outras características que cumpram os requisitos de distância e de contraste na imagem onde ele foi correspondido podem ser incorporadas a ele, e se isso não for possível o objeto como um todo pode ser eliminado como não sendo mais relevante para a localização e mapeamento.

4.3. Triangulação

A triangulação é uma etapa necessária quando sensores de visão são utilizados para que seja possível conseguir informações de distâncias de marcos em relação ao robô. Cada observação de um marco realizada gera o conjunto $\{\boldsymbol{x_s^p}, \boldsymbol{c}\}$ de informações, onde $\boldsymbol{x_s^p} = \{x, y, \theta\}$ representa a posição e orientação do robô no instante de obtenção da imagem (supondo navegação bidimensional) e $\boldsymbol{c} = \{i, j\}$ indica as coordenadas do marco dentro dela. Além disso, a geometria do espelho e do sistema de visão é conhecida, permitindo que seja determinado o raio de luz $\boldsymbol{r}$ que deu origem àquele pixel e com isso o segmento de reta no qual o marco extraido deve estar localizado. Caso esse marco seja correspondido em uma imagem posterior é possível obter um segundo segmento de reta, e a intersecção entre ambos determinará a posição tridimensional do marco no ambiente.

A Figura 4.7 apresenta um diagrama das principais entidades geométricas na triangulação e como elas se relacionam entre si. Os pontos F_1 e F_2 reresentam os focos inferior e superior da hipérbole, sendo que F_1 deve coincidir com o ponto focal da câmera. Os parâmetros a e b são parte da equação de hipérbole H do espelho e z_h é a translação vertical realizada no espelho visando garantir o alinhamento entre o foco da câmera e o foco inferior da hipérbole. O ponto I pode ser obtido pela conversão das coordenadas $\boldsymbol{c}$ em unidades de distância (o fator de conversão é próprio da câmera utilizada), e a sua altura é calculada a partir da distância focal f em relação a F_1.

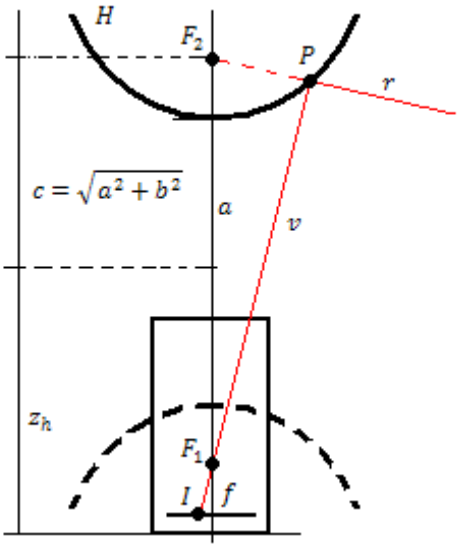


Figura 4.7 - Entidades e parâmetros relevantes na triangulação.

O vetor v que modela o raio de luz incidente na câmera pode ser obtido a partir dos pontos F_1 e I como mostrado abaixo.

$$\boldsymbol{v} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} + \alpha \begin{Bmatrix} i \\ j \\ k \end{Bmatrix} = I + \alpha(F - I) \tag{4.18}$$

Contudo, $\boldsymbol{v}$ ainda não é o raio desejado, pois o marco extraído na imagem representa uma estrutura que foi refletida no espelho hiperbólico utilizado, e por isso é necessário definir qual o vetor que representa esse mesmo raio de luz antes de ser refletido. Para isso devemos calcular inicialmente o ponto *P* na hipérbole H que intersecta $\boldsymbol{v}$.

$$P = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_v + \alpha \begin{Bmatrix} i \\ j \\ k \end{Bmatrix}_v \qquad A = \left(\frac{k_v}{a_h}\right)^2 - \left(\frac{i_v}{b_h}\right)^2 - \left(\frac{j_v}{b_h}\right)^2$$

$$B = 2\left(\frac{z_v k_v}{a_h^2} - \frac{x_v i_v}{b_h^2} - \frac{y_v j_v}{b_h^2} - \frac{k_v z_h}{a_h^2}\right) \tag{4.19}$$

$$\alpha = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad C = \left(\frac{z_v}{a_h}\right)^2 - \left(\frac{x_v}{b_h}\right)^2 - \left(\frac{y_v}{b_h}\right)^2 - 1 - 2\frac{z_v z_h}{a_h^2} + \left(\frac{z_h}{a_h}\right)^2$$

Esse ponto *P* e o foco superior da hipérbole F_2 definem o vetor $\boldsymbol{r}$ incidente que, após ser refletido, dá origem ao vetor $\boldsymbol{v}$ que é capturado pela câmera e foi detectado como representativo de um marco no ambiente. Adicionalmente, é necessário rotacionar e transladar $\boldsymbol{r}$ para que ele leve em consideração a posição e orientação do robô no instante em que a imagem foi obtida, de acordo com o sistema de coordenadas absoluto utilizado na construção do mapa.

$$\boldsymbol{r} = P + \alpha(F_2 - P)\begin{bmatrix} \cos(\theta_v) & -\sin(\theta_v) & x_v \\ \sin(\theta_v) & \cos(\theta_v) & y_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

Essa informação é então armazenada na variável aleatória correspondente a esse marco no mapa de características. Posteriormente, caso haja uma correspondência entre o marco e uma nova característica, a posição da estrutura representada por ambos pode ser calculada como a intersecção entre $\boldsymbol{r_1}$ e $\boldsymbol{r_2}$, ou seja, é o ponto definido pela interseção de dois vetores. Devido ao limite de resolução das imagens utilizada e as imprecisões na correspondência e nos parâmetros utilizados é possível que ambos os raios não se interseccionem, o que em teoria impediria a triangulação. Para evitar instabilidades numéricas em situações como essa os dois vetores são inicialmente projetados no solo e a intersecção entre eles é calculada nessa projeção, de maneira a se obter as coordenadas x e y do marco. Em seguida é calculada a média da

altura entre os dois vetores nessas coordenadas, determinado a coordenada *z* do marco. Essa informação é então incorporada à variável correspondente a esse marco no mapa, gerando uma primeira estimativa de posição ou então refinando aquela que já existe.

$$\boldsymbol{\theta_n} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} (K_1 x_{r1} - K_2 i_{r2} + y_{r2} - y_{r1})/(K_1 - K_2) \\ y_{r1} + K_1(x - x_{r1}) \\ \left(z_{r1} + z_{r2} + \dfrac{(x - x_{r1})}{i_{r1}} + \dfrac{(x - x_{r2})}{i_{r2}}\right)/2 \end{Bmatrix} \qquad \begin{matrix} K_1 = \dfrac{j_{r1}}{i_{r1}} \\ \\ K_2 = \dfrac{j_{r2}}{i_{r2}} \end{matrix} \tag{4.21}$$

4.4. FastSLAM

O algoritmo escolhido nesse trabalho para incorporar as informações obtidas pelo sistema de visão omnidirecional é o FastSLAM, desenvolvido por Montemerlo em parceria com diversos outros pesquisadores (Montemerlo et al.; 2002) (Montemerlo; 2003). Desde então ele vem se tornando cada vez mais popular em tarefas de localização e mapeamento simultâneos, gerando diversos trabalhos que procuram testar e ampliar os seus limites (Haehnel et al.; 2003) (Nieto et al.; 2006) (Thrun et al.; 2004) (Bailey, Nieto, Nebot; 2006) ou aumentar a sua eficiência (Montemerlo et al.; 2003).

A grande contribuição do FastSLAM para o problema da localização e mapeamento simultâneos está na sua capacidade de lidar de uma forma eficiente com grandes quantidades de marcos, algo importante quando são utilizados sensores visuais. Para conseguir isso é explorada a propriedade mostrada em (Murphy; 1999) que diz que o problema do SLAM pode ser dividido em diversos problemas menores de estimação, um para cada variável que é relevante na localização e no mapeamento. Em particular, utiliza-se a noção de independência entre observações que diz que, se a localização do robô é de alguma forma conhecida, o posicionamento dos marcos observados pode ser fatorado em problemas independentes entre si. Essa fatoração é mostrada a seguir, onde $\boldsymbol{x_t}$ é a posiçao do robô, $\boldsymbol{Z_t}$ é o conjunto de observações, $\boldsymbol{u_t}$ é o vetor de controles utilizado e $\boldsymbol{M_t}$ é o mapa composto por $\boldsymbol{m_N^t}$ marcos.

$$p(\boldsymbol{x_t}, \boldsymbol{M_t} | \boldsymbol{Z_t}, \boldsymbol{u_t}) = \underbrace{p(\boldsymbol{x_t} | \boldsymbol{Z^t}, \boldsymbol{u_t})}_{localização} \underbrace{\prod_{i=0}^{N} p(\boldsymbol{m_n^t} | \boldsymbol{x_t}, \boldsymbol{Z_t}, \boldsymbol{u_t})}_{mapeamento} \tag{4.22}$$

Obviamente a localização do robô não é conhecida, mas pode ser aproximada com um Filtro de Partículas, onde cada partícula representa uma possível localização do robô e é tratada internamente como se fosse correta. Para cada uma das partículas o problema de mapeamento do ambiente pode ser subdividido em N problemas de estimação independentes, cada um referente a um único marco, e é tratado por um EKF. Assim sendo, o FastSLAM lida simultaneamente com NP EKFs, um para cada marco para cada partícula, e um Filtro de Partículas para estimar a localização do robô. Esse valor pode parecer alto, mas todas as matrizes possuem uma dimensionalidade constante e baixa (igual a 3 no caso de marcos tridimensionais contra $3N + 3$ no EKF-SLAM), e por serem independentes entre si apenas as estimativas referentes aos marcos observados são atualizadas a cada instante. Implementações eficientes conseguem no FastSLAM um custo computacional proporcional a $P\,logN$ (contra N^2 no EKF-SLAM), permitindo a utilização de milhares ou até mesmo (Montemerlo et al.; 2002) milhões de marcos simultaneamente com um custo computacional aceitável.

Diferentemente do que acontece na abordagem tradicional do EKF-SLAM, o FastSLAM lida com múltiplas hipóteses de mapa, armazenando-as juntamente com a hipótese de localização que cada partícula representa. Assim sendo, é possível lidar com ambigüidades de forma natural, pois diferentes mapas podem possuir diferentes hipóteses de associação de dados, e aquelas que se mostrarem erradas são eliminadas implicitamente durante a etapa de Reamostragem do Filtro de Partículas. Essa propriedade, juntamente com o fato de que o FastSLAM atualiza seus marcos localmente, faz com que ele seja menos sensível a erros de correspondência. Adicionalmente, a utilização de múltiplos mapas gera um efeito de atualização global similar àquele apresentado pelo EKF-SLAM, pois quando uma partícula é eleita a mais provável não é apenas a distribuição de probabilidades da localização do robô que se altera, mas também a de todos os marcos armazenados, refletindo a sua hipótese de mapeamento que também se torna a mais provável.

O alto custo computacional gerado pela necessidade de se manter múltiplos mapas é amortecido por algoritmos eficientes de armazenamento e acesso a informações, que explora as redundâncias existentes entre marcos cuja mesma estimativa é compartilhada por mais do que uma partícula. Consegue-se dessa forma um custo computacional que não depende do tamanho do mapa armazenado, mas sim da quantidade de marcos observados pelo robô a cada instante, que é um valor aproximadamente constante durante todo o período de navegação.

4.4.1. Mapeamento

A etapa de mapeamento no FastSLAM tem como entrada um conjunto de marcos obtidos a partir da imagem omnidirecional coletada pelo sensor de visão, processada de acordo com o SIFT. Esses marcos são correspondidos com aqueles que já estão armazenados do mapa de características do robô e podem ser processados de maneira diferente de acordo com o número de correspondências que já foram realizadas. Quando um marco é observado pela primeira vez ele é adicionado ao mapa sem nenhuma estimativa, de maneira a permitir a sua posterior correspondência e triangulação. Quando a primeira correspondência é realizada é possível por triangulação obter uma primeira estimativa de posição para esse marco, assim como da incerteza referente a essa estimativa. A partir desse momento, novas correspondências gerarão novas estimativas que podem ser utilizadas para refinar aquela já existente, diminuindo a sua incerteza. Essas atualizações são feitas de acordo com as equações do Filtro Estendido de Kalman, supondo distribuições de probabilidade gaussianas e linearização de funções.

Inicialmente é necessário definir um modelo de erros do sistema de visão que quantifique a distribuição de probabilidades que cada uma das estimativas obtidas a partir dele possuirá. Supondo distribuições gaussianas pode-se definir a média como a própria estimativa obtida na triangulação, definindo que ela possui a maior probabilidade de representar o estado correto do robô (os erros envolvidos são aleatórios com média nula). Já a matriz de covariância é obtida empiricamente a partir de testes que quantifiquem a sensibilidade do sistema ao surgimento de erros. A matriz de erros $\boldsymbol{Q}$ utilizada nesse trabalho é mostrada em (4.23), onde σ_{ij} indica a sensibilidade que o sistema possui para o surgimento de erros do tipo i devido a uma estimativa do tipo j. Por exemplo, a sensibilidade do robô ao surgimento de erros na estimativa de distância devido aos valores referentes à sua estimativa de orientação angular em relação ao eixo *xy* é dada pelo coeficiente $\sigma_{d\varphi}$.

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_{dd} & \sigma_{d\theta} & \sigma_{d\varphi} \\ \sigma_{\theta d} & \sigma_{\theta\theta} & \sigma_{\theta\varphi} \\ \sigma_{\varphi d} & \sigma_{\varphi\theta} & \sigma_{\varphi\varphi} \end{bmatrix} \tag{4.23}$$

Devido à deformação que a geometria omnidirecional gera sobre diferentes regiões da imagem podem ser utilizados valores diferentes de sensibilidade de acordo com as coordenadas do pixel que representa a característica. Em especial, regiões mais distantes do centro serão mais sensíveis do que as centrais, gerando uma matriz $\boldsymbol{Q}$ com valores mais elevados.

4.4.1.1. Adição de Marcos

Um marco observado pela primeira vez em uma imagem não possui uma estimativa de posição no ambiente, e por isso também não possui uma incerteza em relação a essa posição (pode-se, ao invés disso, dizer que essa incerteza é infinita). A única informação disponível nesse instante é a posição $\boldsymbol{x_s^p} = \{x, y, \theta\}$ ocupada pelo robô no instante em que o marco foi observado de acordo com a hipótese de localização da partícula p e as coordenadas $\boldsymbol{c} = \{i, j\}$ do pixel que representa o marco na imagem, além do vetor descritor $\boldsymbol{\psi}$ da característica que gerou o marco. Esses valores são armazenados na variável correspondente do mapa de característica para que possam ser recuperados posteriormente durante a etapa de correspondência e, caso ela seja positiva, de triangulação. Esse processo é realizado uma vez para cada marco detectado pela primeira vez para cada partícula do Filtro de Partículas, sendo que um mesmo marco adicionado em diferentes partículas difere apenas pela posição do robô, possuindo as mesmas coordenadas na imagem e vetor descritor.

$$\boldsymbol{m_{n,p}^t} = \{\boldsymbol{c}, \boldsymbol{x_s^p}, \boldsymbol{\psi}\} \tag{4.24}$$

4.4.1.2. Atualização de Marcos

Quando uma característica extraída de uma imagem é correspondida com um marco armazenado no seu mapa é possível calcular o vetor de observações $\boldsymbol{z_{n,k}^{p,t}}$ referente ao marco que foi correspondido, a partir da sua estimativa de posição $\boldsymbol{\theta_k^t}$ no ambiente e da posição do robô $\boldsymbol{x_t^p}$ naquele instante. Esse vetor é composto por três valores, o primeiro correspondente à distância d entre o sensor e o marco e os outros dois correspondentes aos ângulos θ e φ que o posicionam unicamente no espaço. Esse vetor de observações é definido em (4.25) e será diferente para cada partícula, pois cada uma possui uma hipótese de localização diferente.

$$\boldsymbol{z_{n,k}^{p,t}} = \begin{Bmatrix} d \\ \theta \\ \varphi \end{Bmatrix} = \begin{Bmatrix} \sqrt{\left(x_n^t - x_t^p\right)^2 + \left(y_n^t - y_t^p\right)^2 + \left(z_n^t - z_t^p\right)^2} \\ atan2\left(y_n^t - y_t^p \ , \ x_n^t - x_t^p\right) - \theta_t^p \\ atan2\left(z_n^t - z_t^p \ , \ \sqrt{\left(x_n^t - x_t^p\right)^2 + \left(y_n^t - y_t^p\right)^2}\right) \end{Bmatrix} \tag{4.25}$$

A atualização de marcos pode ocorrer de duas maneiras diferentes. Caso essa seja a primeira correspondência, o marco ainda não contará com nenhuma estimativa de posição, e por isso essa primeira estimativa será incorporada completamente à sua variável, assim como a sua incerteza. Caso seja uma correspondência posterior, essa estimativa, assim como a sua incerteza, será utilizada para atualizar aquela já existente no marco, de maneira a diminuir a sua incerteza inerente.

Primeira Correspondência

Quando um marco $\boldsymbol{m_n}$ é correspondido pela primeira vez, a sua estimativa de posição é definida pelo resultado $\boldsymbol{\theta_k^t}$ da triangulação realizada. Resta então calcular a matriz de covariância da incerteza envolvida $\boldsymbol{\Sigma_{n,p}^t}$ nessa medida, o que pode ser feito a partir do modelo de erros do sensor (4.23) e do vetor de observações (4.25) obtido também a partir da triangulação. Antes de ser utilizado, o modelo de erros deve ser projetado nos eixos cartesianos, de maneira a indicar a área no ambiente onde o marco pode estar posicionado. Essa projeção é feita a partir da matriz R, que incorpora os valores contidos em $\boldsymbol{z_{n,k}^{p,t}}$ e determina em quais eixos as sensibilidades apresentadas na matriz $\boldsymbol{Q}$ serão relevantes para o surgimento de erros.

$$\boldsymbol{R_{k,p}^t} = \begin{bmatrix} \cos(\theta)\cos(\varphi) & -d\sin(\theta)\cos(\varphi) & 0 \\ \sin(\theta)\cos(\varphi) & d\cos(\theta)\cos(\varphi) & 0 \\ -\sin(\varphi) & 0 & -d\sin(\varphi) \end{bmatrix} \tag{4.26}$$

A matriz de covariância $\Sigma_{n,p}^t$ é então dada então pela seguinte multiplicação matricial:

$$\boldsymbol{\Sigma_{n,p}^t} = \left(\boldsymbol{R_{k,p}^t}\right)\boldsymbol{Q}\left(\boldsymbol{R_{k,p}^t}\right)^T \tag{4.27}$$

Esses dois valores são adicionados à variável representativa do marco no ambiente apresentada em (4.24), completando-a com uma estimativa de posição e sua respectiva incerteza como mostrado em (4.28). Para efeitos de triangulação, as informações referentes às coordenadas c do pixel na imagem são atualizadas com os valores obtidos na última observação, e com isso a posição do robô x^p no instante em que observou a imagem também o será.

$$\boldsymbol{m_{n,p}^t} = \left\{\boldsymbol{c}, \boldsymbol{x_s^p}, \boldsymbol{\psi}, \boldsymbol{\theta}, \boldsymbol{\Sigma}\right\} \tag{4.28}$$

Correspondências Posteriores

A atualização do estado dos marcos armazenados no mapa de características é realizada de acordo com as equações do EKF, individualmente para cada marco em cada uma das partículas. O sistema de equações não-lineares genéricas que o EKF se propõe a resolver é mostrado a seguir:

$$\begin{aligned} \boldsymbol{x_{t+1}} &= g(\boldsymbol{x_t}, \boldsymbol{u_t}) + \boldsymbol{v_t} \\ \boldsymbol{z_t} &= h\left(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t}\right) + \boldsymbol{\omega_t} \end{aligned} \tag{4.29}$$

Onde $\boldsymbol{v_t}$ e $\boldsymbol{\omega_t}$ são ruídos referentes aos modelos de movimentação e de medidas que os sensores utilizados para obter essas estimativas possuem. Esse trabalho parte do pressuposto de que os marcos observados não se movimentam, portanto os seus estados de localização não se propagam no tempo. Os ruídos referentes ao sistema de visão são modelados pela matriz $\boldsymbol{Q}$ definida em (4.23) e são considerados constantes, não variando no tempo (embora possam variar de acordo com a posição do pixel que representa a característica na imagem). Dessa forma, esse sistema pode ser simplificado da seguinte forma:

$$\begin{aligned} \boldsymbol{x_{t+1}} &= \boldsymbol{x_t} \\ \boldsymbol{z_t} &= h\left(\boldsymbol{x_t}, \boldsymbol{\theta_{n,p}^t}\right) + \boldsymbol{Q} \end{aligned} \tag{4.30}$$

A função que relaciona a localização do robô $\boldsymbol{x_t^p}$ em um determinado instante t com a posição de um marco $\boldsymbol{\theta_{n,p}^t}$ já armazenado em seu mapa, ambas de acordo com as hipóteses referentes à partícula p, é indicada abaixo.

$$h\left(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t}\right) = \begin{Bmatrix} \sqrt{(x_n^t - x_t)^2 + (y_n^t - y_t)^2 + (z_n^t - z_t)^2} \\ atan2(y_n^t - y_t \ , \ x_n^t - x_t) - \theta_t \\ atan2\left(z_n^t - z_t \ , \ \sqrt{(x_n^t - x_t)^2 + (y_n^t - y_t)^2}\right) \end{Bmatrix} \tag{4.31}$$

Como a formulação matemática do Filtro de Kalman se baseia em sistemas lineares, é necessário linearizar a função $h\left(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t}\right)$. O Jacobiano $\boldsymbol{H_{n,p}^t}$ apresentado em (4.32) consegue essa linearização através de uma expansão de Taylor, que calcula o valor da função em seu ponto

de maior probabilidade (valor $\boldsymbol{\theta_{n,p}^{t}}$ obtido na triangulação) e utiliza as derivadas parciais em relação a cada uma das coordenadas do marco para estimar o valor no ponto desejado.

$$\boldsymbol{H_{n,p}^{t}} = \left\{ \frac{\partial h(\boldsymbol{x_t^p}, \boldsymbol{x_{n,p}^t})}{\partial x} \quad \frac{\partial h(\boldsymbol{x_t^p}, \boldsymbol{x_{n,p}^t})}{\partial y} \quad \frac{\partial h(\boldsymbol{x_t^p}, \boldsymbol{x_{n,p}^t})}{\partial z} \right\}\Bigg|_{x_t=\mu_t} = \left(\begin{array}{ccc} \frac{\Delta x}{\sqrt{q}} & \frac{\Delta y}{\sqrt{q}} & -\frac{\Delta z}{\sqrt{q}} \\ -\frac{\Delta y}{\sqrt{d}} & \frac{\Delta x}{\sqrt{d}} & 0 \\ -\frac{\Delta z \Delta x}{q\sqrt{d}} & -\frac{\Delta z \Delta y}{q\sqrt{d}} & -\frac{\sqrt{d}}{q} \end{array} \right)\Bigg|_{x=\mu_t} \tag{4.32}$$

Onde $\Delta x = x_n^t - x_t^p$, $\Delta y = y_n^t - y_t^p$, $\Delta z = z_n^t - z_t^p$, $q = \Delta x^2 + \Delta y^2 + \Delta z^2$ *e* $d = \Delta x^2 + \Delta y^2$. A diferença $\boldsymbol{y_{k,n}^{p,t}}$ entre o vetor de observações $\boldsymbol{z_{n,k}^{p,t}}$ obtido a partir dos valores de triangulação e o vetor $h(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t})$ obtido a partir dos valores contidos no mapa do robô indica o erro entre o que foi observado e o que deveria ter sido.

$$\boldsymbol{y_{k,n}^{p,t}} = \boldsymbol{z_{n,k}^{p,t}} - h(\boldsymbol{x_t^p}, \boldsymbol{\theta_{n,p}^t}) \tag{4.33}$$

A covariância da inovação $S_{n,p}^t$, que indica a incerteza relativa a esse erro e que posteriormente será utilizada também na atualização dos pesos das partículas, é calculada com o auxílio do jacobiano $H_{n,p}^t$ da observação e do modelo de erros Q.

$$\boldsymbol{S_{n,p}^t} = (\boldsymbol{H_{n,p}^t})\boldsymbol{\Sigma_{n,p}^t}(\boldsymbol{H_{n,p}^t})^T + \boldsymbol{Q} \tag{4.34}$$

O Ganho de Kalman $\boldsymbol{K_{n,p}^t}$ é definido como:

$$\boldsymbol{K_{n,p}^t} = \boldsymbol{\Sigma_{n,p}^t}(\boldsymbol{H_{n,p}^t})^T(\boldsymbol{S_{n,p}^t})^{-1} \tag{4.35}$$

Com isso é possível calcular a nova posição do marco (4.36) e a sua matriz de covariância correspondente (4.37).

$$\boldsymbol{\theta_{n,p}^{t+1}} = \boldsymbol{\theta_{n,p}^t} + \boldsymbol{K_{n,p}^t y_{k,n}^{p,t}} \tag{4.36}$$

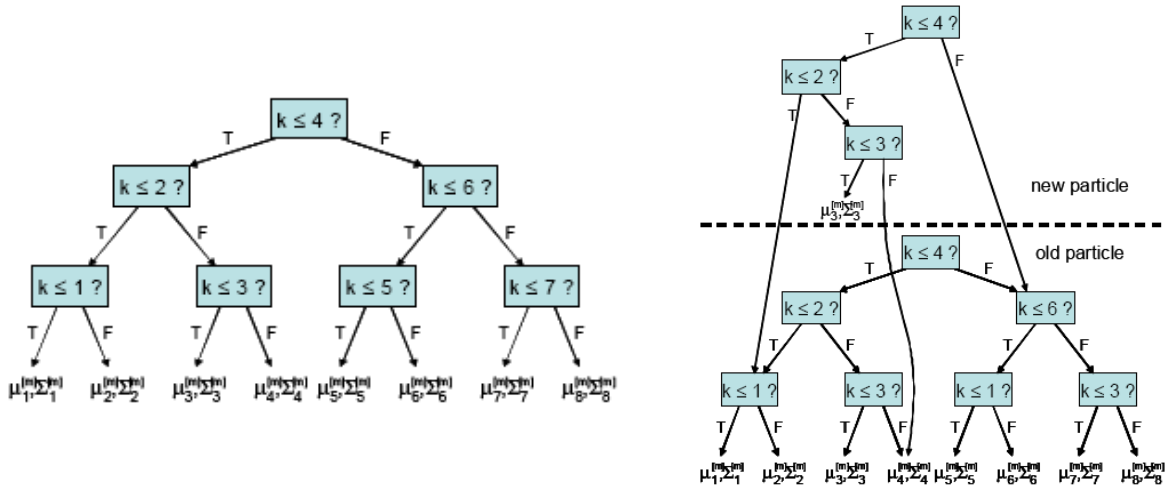
$$\boldsymbol{\Sigma_{n,p}^{t+1}} = (\boldsymbol{I} - \boldsymbol{K_{n,p}^t H_{n,p}^t})\boldsymbol{\Sigma_{n,p}^t} \tag{4.37}$$

Essas estimativas substituem aquelas armazenadas na variável correspondente ao marco no mapa. A incorporação de novas informações referentes a esse marco faz com que a sua incerteza diminua, aumentando a precisão da estimativa final de posicionamento e melhorando os resultados finais de mapeamento.

4.4.1.3. Árvore de Marcos

Uma implementação direta da etapa de mapeamento no FastSLAM possui um custo computacional proporcional à quantidade de partículas e de marcos utilizados. Isso porque cada partícula carrega consigo um mapa composto por estimativas de posição de cada um de seus marcos, e deve propagar essas estimativas no tempo de acordo com a evolução do Filtro de Partículas. O custo computacional relativo à quantidade de partículas não pode ser evitado porque cada partícula representa uma hipótese independente. Contudo, devido à relação de independência que foi estabelecida entre marcos, grande parte da informação contida nos mapas construídos dessa forma será redundante, pois diversas partículas eventualmente compartilharão de uma mesma estimativa referente a um marco, principalmente após uma etapa de Reamostragem.

Tirando proveito disso, o FastSLAM consegue diminuir o seu custo computacional para uma função logarítmica em relação à quantidade de marcos, permitindo a utilização simultânea de quantidades muito maiores do que seria possível de outra forma. Para isso é utilizada uma árvore binária (Figura 4.8a), onde a estimativa de cada marco é armazenada em uma das folhas (nós inferiores da árvore). Partículas que compartilham de uma mesma estimativa relativa a um marco compartilharão dessa mesma folha, o que elimina a cópia de informações redundantes em diversas partículas. Conforme novas estimativas referentes a um marco são incorporadas são geradas novas folhas que as armazenam e novos caminhos que apontam para elas, mantendo os outros intactos (Figura 4.8b)



(a) *(b)*

Figura 4.8 - Árvore binária de marcos tradicional (Montemerlo; 2003).
(a) Formato estático (b) Adição de novas estimativas

Nesse trabalho a estrutura apresentada acima foi modificada de maneira a se obter um resultado em média mais eficiente, que no pior caso alcança o custo logarítmico da implementação padrão utilizada pelo FastSLAM. Ao invés de armazenar as estimativas apenas nas folhas da árvore binária elas são armazenadas em todos os nós (Figura 4.10a), gerando uma economia de memória (são necessários menos nós) e de tempo (não é necessário percorrer toda a árvore para se chegar em alguns dos nós). Cada marco recebe um índice único que indica qual a sua posição dentro da árvore, e ela é manuseada de maneira a sempre permanecer balanceada, com a alteração de sua raiz, quando necessário. Esse balanceamento é realizado pela rotação da raiz da árvore, realizada no sentido anti-horário (Figura 4.9) devido ao acréscimo de marcos em ordem ascendente de índices que faz a árvore ficar mais populada do lado direito.

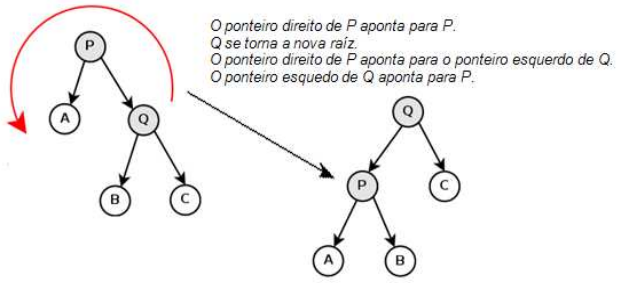
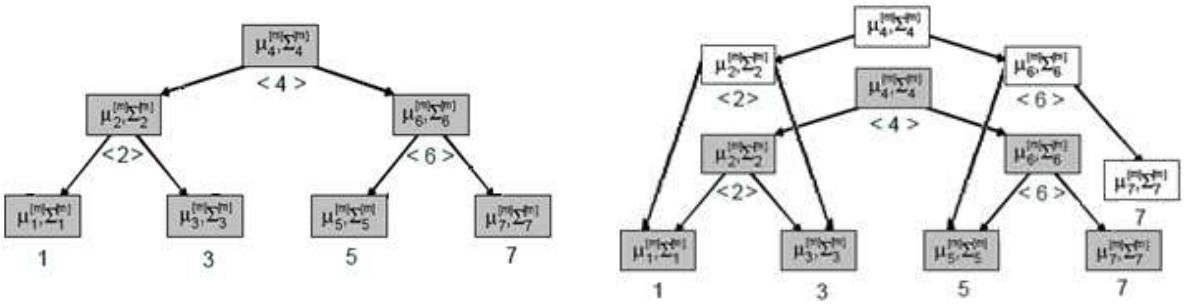


Figura 4.9 - Rotação de nós para a esquerda.

Quando um marco é observado pela partícula e a sua estimativa de posição é alterada, um novo nó é gerado para armazenar essa estimativa, e um novo caminho que leva até ele é construído na árvore binária da partícula. Caso esse caminho passe por outras estimativas, elas também serão reconstruídas, realizando uma cópia aparentemente desnecessária de informações. Contudo, caso esses marcos sejam observados em um instante posterior, suas estimativas poderão ser atualizadas sem a necessidade de construir outro caminho, pois ele já será único àquela partícula. De qualquer forma, a implementação apresentada em trabalhos tradicionais de FastSLAM requer a criação de nós responsáveis pela busca binária, gerando um custo semelhante.



(a) *(b)*

Figura 4.10 - Árvore binária proposta. (a) Formato estático (b) Adição de novas estimativas

A Figura 4.10b mostra um exemplo da estrutura descrita aqui, para o caso em que dois marcos (2 e 7) foram observados. Os nós em cinza são referentes à árvore binária compartilhada, e os em branco foram reconstruídos devido às estimativas incorporadas pela partícula em seu mapa. Uma implementação tradicional como aquela mostrada em (Montemerlo et al.; 2003) necessitaria de mais três nós para conseguir o mesmo resultado, dois para armazenar as estimativas propriamente ditas e um para completar o caminho que leva até a folha que armazenaria a estimativa do marco de identidade 2. Dessa forma é necessário um menor tempo de atualização para conseguir o mesmo resultado, o que gera um aumento de eficiência no algoritmo. Adicionalmente, a árvore binária tradicional necessita de 14 nós para armazenar essa informação, contra apenas 7 necessários na implementação proposta, gerando uma economia significativa de memória.

4.4.2. Localização

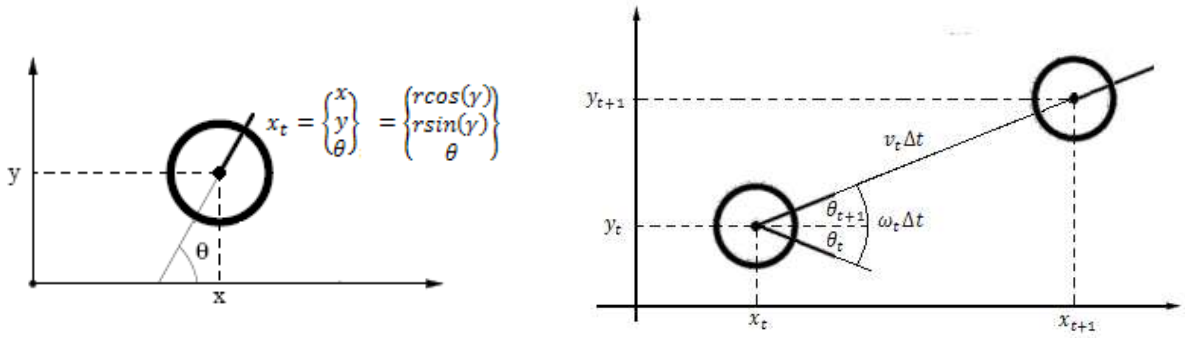
O problema da localização é tratado pelo FastSLAM por um Filtro de Partículas, que mantém um conjunto finito de amostras retiradas da distribuição de probabilidades que modela a posição do robô a cada instante (Liu, Chen; 1998). Cada partícula representa uma hipótese diferente de localização e mapeamento, e utiliza esses valores para determinar o seu peso, um indicador da sua probabilidade de representar os estados corretos do robô em um dado instante. Conforme novas informações relativas aos marcos observados são incorporadas a essas estimativas esse peso é alterado de maneira a refletir a distribuição de probabilidades das partículas naquele instante.

Um Filtro de Partículas é composto por três etapas que são realizadas em seqüência. Na Figura 4.11 essa seqüência é indicada, assim como a influência que cada uma das etapas possui sobre a distribuição das partículas no ambiente e sobre o peso de cada uma delas. A seguir são apresentados os conceitos e a formulação envolvida em cada uma dessas etapas.

Figura 4.11 - Etapas do Filtro de Partículas (regiões escuras possuem maior probabilidade de serem corretas).

4.4.2.1. Predição

A etapa de Predição no Filtro de Partículas não depende de informações externas, apenas do modelo de movimentação que o robô utiliza para estimar a sua localização conforme navega pelo ambiente. Inicialmente o estado de cada partícula é propagado de maneira determinística, de acordo com o vetor de controles u_t fornecido naquele instante pelo algoritmo de controle. Esse trabalho lidará apenas com robôs diferenciais não-holonômicos se movimentando por um plano, o que restringe esses graus de liberdade a apenas três, sendo dois necessários para descrever sua posição no plano e o terceiro para descrever sua orientação (Figura 4.12a). O deslocamento do robô é realizado partir de um vetor de velocidades $\boldsymbol{u_t} = \{v, \omega\}^T$ suposto constante em cada intervalo de tempo e cujos coeficientes agem de maneira independente no robô, primeiro o rotacionando ainda na posição inicial e em seguida transladando-o até a posição final (Figura 4.12b).



(a) *(b)*

Figura 4.12 - Modelo de odometria. (a) Estado de localização (b) Modelo de movimentação

A função de transição que modela esse deslocamento do robô é apresentada em (4.38) e permite a propagação do estado de localização de cada uma das partículas no tempo.

$$\boldsymbol{x_{t+1}^p} = \begin{Bmatrix} x \\ y \\ \theta \end{Bmatrix} = \boldsymbol{x_t^p} + \dot{\boldsymbol{x}}_t \Delta t = g\left(\boldsymbol{x_t^p}, \boldsymbol{u_t}\right) = \begin{Bmatrix} x_t + v_t \Delta t cos(\theta_t + \omega_t \Delta t) \\ y_t + v_t \Delta t sin(\theta_t + \omega_t \Delta t) \\ \theta_t + \omega_t \Delta t \end{Bmatrix} \tag{4.38}$$

Essa transição é determinística, ou seja, não leva em consideração os erros acumulados pelo robô durante o deslocamento. Esses erros podem ser causados devido a imprecisões na rotação e na translação do robô, e para maximizar o conjunto de erros possíveis (Thrun, Fox, Burgard; 2005) é incluído um terceiro grau de liberdade durante a movimentação, referente a

uma rotação ao redor da posição final, como mostrado na Figura 4.13. Assim, o robô inicialmente rotaciona na sua posição inicial, em seguida translada para a posição final e em seguida rotaciona nela de maneira a assumir a sua orientação final, e todos esses movimentos estão sujeitos a erros.

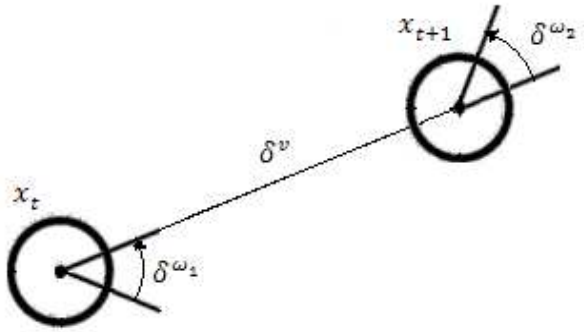


Figura 4.13 - Deslocamentos realizados durante a movimentação.

Esses três deslocamentos podem ser facilmente calculados a partir da geometria do problema. A primeira rotação é calculada subtraindo-se a orientação intermediária do robô (obtida após a primeira rotação) da sua orientação inicial. A translação é calculada pelo Teorema de Pitágoras levando-se em consideração o deslocamento do robô nas coordenadas x e y. A última rotação é calculada pela subtração da orientação final do robô da sua orientação intermediária.

$$\boldsymbol{\delta_t^p} = \begin{Bmatrix} \delta^{\omega_1} \\ \delta^{v} \\ \delta^{\omega_2} \end{Bmatrix} = \begin{Bmatrix} atan2(y_{t+1} - y_t, x_{t+1} - x_t) - \theta_t \\ \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} \\ \theta_{t+1} - \theta_t - \delta^{\omega_1} \end{Bmatrix} \tag{4.39}$$

Esses deslocamentos médios são utilizados como base para a incorporação do modelo de erros desenvolvido para o robô. Intuitivamente, quanto maior o deslocamento maior deverá será o erro incorporado à estimatva, de acordo com a sensibilidade que o robô possui para o surgimento de erros de cada natureza. Essa sensibilidade é estimada empiricamente através da atribuição de coeficientes α_{ij} que representam a propensão que o robô possui para o surgimento de erros do tipo j quando submetido a uma movimentação do tipo i. Por exemplo, a sensibilidade do robô ao surgimento de erros de translação quando ele se movimenta angularmente é regida pelo coeficiente $\alpha_{v\omega}$, enquanto a sua sensibilidade ao surgimento de erros de orientação quando ele se movimenta linearmente é regida pelo coeficiente $\alpha_{\omega v}$.

$$\alpha = \begin{bmatrix} \alpha_{vv} & \alpha_{v\omega} \\ \alpha_{\omega v} & \alpha_{\omega\omega} \end{bmatrix} \tag{4.40}$$

Multiplicando os deslocamentos obtidos pelos seus coeficientes correspondentes são determinados os desvios-padrão que representam a incerteza em relação a cada um dos deslocamentos realizados, modelando-os como uma distribuição gaussiana.

$$\boldsymbol{\sigma}_{\boldsymbol{p,t}}^{\boldsymbol{\delta}} = \begin{Bmatrix} \sigma^{\omega_1} \\ \sigma^{v} \\ \sigma^{\omega_2} \end{Bmatrix} = \begin{Bmatrix} \alpha_{\omega\omega}\delta_{p,t}^{\omega_1} + \alpha_{v\omega}\delta_{p,t}^{v} \\ \alpha_{\omega v}\left(\delta_{p,t}^{\omega_1} + \delta_{p,t}^{\omega_2}\right) + \alpha_{vv}\delta_{p,t}^{v} \\ \alpha_{\omega\omega}\delta_{p,t}^{\omega_2} + \alpha_{v\omega}\delta_{p,t}^{v} \end{Bmatrix} \tag{4.41}$$

A posição de cada uma das partículas é obtida por amostragem dentro dessa distribuição de probabilidades, onde a média é igual ao valor determinístico obtido em (4.38) e o desvio padrão é dado por (4.41). Uma maneira de se realizar essa amostragem é através da equação de Box-Muller, mostrada em (4.42) onde a e b são variáveis aleatórias uniformemente distribuídas entre [0,1].

$$\lambda(\mu, \sigma) = \mu + \sigma\left(\sqrt{-2\ln(a)}\cos(2\pi b)\right) \tag{4.42}$$

O resultado $\lambda(\mu, \sigma)$ obtido é uma variável aleatória gaussiana com média μ e desvio padrão σ. O deslocamento δ_t^p possui então a seguinte forma:

$$\boldsymbol{\delta}_{\boldsymbol{t}}^{\boldsymbol{p}} = \overline{\boldsymbol{\delta}}_{\boldsymbol{t}}^{\boldsymbol{p}} + \begin{Bmatrix} \lambda\left(0, \sigma_{p,t}^{\omega_1}\right) \\ \lambda\left(0, \sigma_{p,t}^{v}\right) \\ \lambda\left(0, \sigma_{p,t}^{\omega_2}\right) \end{Bmatrix} \tag{4.43}$$

Para se obter $\boldsymbol{x}_{\boldsymbol{t+1}}^{\boldsymbol{p}}$ é necessário projetar $\boldsymbol{\delta}_{\boldsymbol{t+1}}^{\boldsymbol{p}}$ de volta para o eixo cartesiano.

$$\boldsymbol{x}_{\boldsymbol{t+1}}^{\boldsymbol{p}} = \zeta\left(\boldsymbol{x}_{\boldsymbol{t}}^{\boldsymbol{p}}, \boldsymbol{u}_{\boldsymbol{t}}\right) = x_t^p + \begin{Bmatrix} \delta_{p,t}^{v}\cos\left(\theta_t + \delta_{p,t}^{\omega_1}\right) \\ \delta_{p,t}^{v}\sin\left(\theta_t + \delta_{p,t}^{\omega_1}\right) \\ \delta_{p,t}^{\omega_1} + \delta_{p,t}^{\omega_2} \end{Bmatrix} \tag{4.44}$$

FastSLAM 2.0

O FastSLAM 2.0, apresentado em (Montemerlo et al.; 2003) é uma evolução do algoritmo tradicional de FastSLAM que procura aumentar a eficiência do Filtro de Partículas utilizado para estimar a localização do robô a cada instante. Mais especificamente, o FastSLAM 2.0 procura aumentar a eficiência da etapa de *Predição* do Filtro de Partículas de forma a evitar o surgimento de partículas em regiões de baixa probabilidade. Para conseguir isso, as informa-

ções coletadas pelos sensores do robô são utilizadas diretamente na etapa de amostragem de partículas, ao invés de apenas na determinação do peso de cada uma delas. Não há nenhum aumento na qualidade dos resultados finais obtidos pelo FastSLAM 2.0 em relação à implementação original, apenas consegue-se obter os mesmos resultados com significativamente menos partículas. Em particular, para sistemas lineares o FastSLAM 2.0 consegue convergência para $P = 1$, constituindo dessa forma o único algoritmo de SLAM com tempo de atualização constante.

Com mostrado anteriormente, a etapa de Predição envolve a amostragem do estado de localização de cada uma das partículas dentro da distribuição de probabilidades gerada pela propagação desse estado no tempo, cuja média é dada por (4.38) e o desvio padrão por (4.41). No FastSLAM 2.0 essa distribuição de probabilidades é modificada pelas observações coletadas pelo que foram correspondidas com marcos que já possuem uma estimativa de posição, de maneira a diminuir a sua imprecisão e aumentar a probabilidade de que sejam amostradas partículas próximas do estado real de localização do robô. A covariância em relação a cada observação é calculada a partir da matriz $S_{n,p}^{t}$ de inovação calculada durante a etapa de atualização do mapeamento (4.34), e a estimativa quanto à posição do marco no ambiente pode ser transformada para uma estimativa quanto à localização do robô com o auxílio de $H_{n,p}^{t}$.

$$\boldsymbol{\Sigma_t^p} = \left(\boldsymbol{H_{n,p}^t}^{T}\left(\boldsymbol{S_{n,p}^t}\right)^{-1}\boldsymbol{H_{n,p}^t} + \left(\boldsymbol{\Sigma_t^p}\right)^{-1}\right)^{-1} \tag{4.45}$$

Em (4.45) é calculada a nova covariância da distribuição de probabilidades que será utilizada na amostragem durante a Predição. Já a nova média é calculada em (4.46), novamente utilizado a matriz de covariância da inovação $S_{n,p}^{t}$ e o jacobiano $H_{n,p}^{t}$.

$$\boldsymbol{x_t^p} = \boldsymbol{x_t^p} + \left(\boldsymbol{H_{n,p}^t}\right)^{T}\boldsymbol{S_{n,p}^t}\left(\boldsymbol{H_{n,p}^t}\right) \tag{4.46}$$

No caso de haver mais de uma observação correspondida, todo o processo deve ser repetido de maneira a incorporar a sua estimativa também. É importante lembrar que a alteração da localização estimada do robô fará com que o jacobiano $\boldsymbol{H_{n,p}^t}$ e a matriz de covariância $\boldsymbol{S_{n,p}^t}$ se alterem, e devam ser calculadas novamente. Adicionalmente, as adições em (4.45) e (4.46) devem ser feitas utilizando-se os resultados obtidos na incorporação da estimativa anterior, e não os originais obtidos na última iteração. Uma vez que todas as observações já tenham sido incorporadas, pode-se realizar a amostragem com a média e covariância resultantes.

O FastSLAM 2.0 tende a ser mais eficiente do que o FastSLAM especialmente em situações onde a precisão dos sensores de coleta de informações do ambiente é muito maior em relação à precisão do sistema de odometria utilizado, algo que acontece na grande maioria das aplicações (Montemerlo et al.; 2003). Isso faz com que a região de probabilidades onde uma partícula pode se localizar durante a predição se torne grande em comparação àquela que a Reamostragem gera uma vez que as observações são incorporadas, gerando um desperdício de partículas. Ao manter uma maior quantidade de partículas em áreas de alta probabilidade o FastSLAM 2.0 consegue também uma maior diversidade de hipóteses, o que resulta na capacidade de fechamento de *loops* mais extensos.

4.4.2.2. Atualização

Durante a etapa de Predição o estado de localização das partículas foi alterado pela movimentação do robô, de acordo com seus estados anteriores, o vetor de controles e a função de transição que os relaciona. O estado de mapeamento das partículas também foi alterado pelas etapas do EKF que modelam a posição de cada um dos marcos que elas possuem em seus respectivos modelo de mapa. A etapa de Atualização do Filtro de Partículas utiliza essas alterações para verificar quais partículas possuem o conjunto de estados de localização e mapeamento com maior probabilidade de representar o estado correto que o robô apresenta naquele instante. Essa probabilidade é modelada pelo peso que cada partícula possui, sendo que quanto maior o peso maior a probabilidade, e supõe-se que a soma de todos os pesos é igual a 1, ou seja, o estado correto do robô é representado de maneira exata por uma de suas partículas.

As mesmas informações que foram utilizadas para atualizar a estimativa de posição de um marco também podem ser utilizadas para atualizar a estimativa de posição do robô, pois esses valores estão relacionados entre si. Uma observação determina a posição de um marco no ambiente quando a localização do robô é conhecida, e uma observação também determina a localização do robô no ambiente caso a posição do marco seja conhecida. Dessa forma, quando não há nenhuma correspondência entre características e marcos a etapa de Atualização não acontece, e os pesos das partículas se mantém os mesmos durante essa iteração. Similarmente, caso ocorram várias correspondências os pesos são alterados seqüencialmente, devido à hipótese de independência estabelecida entre os marcos que compõem o mapa.

Quanto mais próximas a estimativas de localização e mapeamento da partícula estiverem da estimativa obtida pela observação correspondida, maior a probabilidade de que ela represente o estado correto do robô, e por isso maior será o seu peso. Essa semelhança é quantificada pelo vetor de diferenças $\boldsymbol{y_{k,n}^{p,t}}$ definido em (4.33), sendo que quanto maior ele for menor será a semelhança. A matriz de covariância da inovação $\boldsymbol{S_{n,p}^{t}}$ definida em (4.34) quantifica a imprecisão relacionada à observação correspondida, sendo que quanto maior ela for menor será o peso que ela possuirá sobre a atualização dos pesos. Essa atualização tem a forma de uma distribuição gaussiana multivariada, que leva em consideração as três parcelas do vetor de informações $z_{k,p}^{t}$ definido em (4.25) onde $N = 3$.

$$w_t^p = w_{t-1}^p \left(\frac{1}{(2\pi)^{N/2} \left| S_{n,p}^t \right|^{1/2}} \exp\left(-\frac{1}{2} \left(y_{k,n}^{p,t} \right) \left(S_{n,p}^t \right)^{-1} \left(y_{k,n}^{p,t} \right)^T \right) \right) \tag{4.47}$$

Depois que todas as atualizações tenham sido feitas os pesos do conjunto de partículas são normalizados (4.48) para que a sua soma volte a ser igual a 1. O resultado da etapa de Atualização é uma nova distribuição de probabilidades que leva em consideração as informações obtidas pelo robô naquele instante. A partícula com o maior peso representa o estado mais provável de localização e mapeamento para o robô, indicando qual a sua posição e a de todos os marcos já observados no ambiente.

$$w_{t+1}^p = \frac{w_{t+1}^p}{\sum_{i=0}^{P} w_{t+1}^p} \tag{4.48}$$

4.4.2.3. Reamostragem

A etapa de Reamostragem procura eliminar partículas que não representem uma hipótese válida para a localização do robô, ao mesmo tempo em que multiplica aquelas que possuem pesos elevados, aumentando a densidade de partículas na região que se supõe próxima da localização real do robô. Como a quantidade de partículas com as quais se lida é constante (ou ao menos finita) a densidade delas tende a diminuir conforme ocorre o espalhamento causado pela etapa de Predição, fazendo com que eventualmente não reste nenhuma partícula próxima do local onde o robô realmente se encontra e até a partícula de maior peso não representará uma estimativa precisa o suficiente.

Como essa proximidade é determinada pelo peso que cada partícula possui, a Reamostragem se baseia em procurar eliminar aquelas que possuem um peso baixo e criar novas cópias daquelas que possuem um peso elevado. Uma maneira simples de se realizar isso e que gera resultados bastante satisfatórios consiste em realizar uma amostragem no conjunto de partículas, utilizando os pesos como distribuição de probabilidades. Partículas com maior peso terão maior probabilidade de ser escolhidas, o que é desejável dado que isso também significa que elas têm uma maior chance de representarem o estado correto de localização do robô. Outros métodos de se realizar a Reamostragem são apresentados em (Rekleitis; 2003), e um método particularmente eficiente, que possui custo computacional linear em relação à quantidade de partículas, é encontrado (Thrun, Fox, Burgard; 2005).

$$\boldsymbol{S_{t+1}} = \{p_{t+1}^{R_1} p_{t+1}^{R_2}, \dots, p_{t+1}^{R_P}\} \ , \ r < \sum_{i=0}^{R} w_{t+1}^{i} \tag{4.49}$$

Inicialmente, é amostrado um número aleatório r de uma distribuição uniforme normalizada entre [0,1]. O conjunto de partículas $\boldsymbol{S_{t+1}^{-}}$ é então percorrido linearmente desde o seu início, e os pesos w_{t+1}^{p} de cada partícula são somados até que o resultado se torne maior do que r. A partícula p_t^R cujo peso fez com que a somatória ultrapassasse r será escolhida como amostra e adicionada no conjunto final de partículas $\boldsymbol{S_{t+1}}$. Partículas com maior peso contribuirão mais na somatória, aumentando a probabilidade de sejam responsáveis por ultrapassar o valor de r e com isso sejam escolhidas. Esse processo, equacionado em (4.49), é repetido até que $\boldsymbol{S_{t+1}}$ tenha a mesma quantidade de elementos que $\boldsymbol{S_{t+1}^{-}}$, sendo que uma única partícula pode ser escolhida mais de uma vez. Nesse caso o conjunto final de partículas possuirá estados redundantes, mas durante a próxima etapa de Predição eles se diferenciarão devido aos processos aleatórios envolvidos.

Diferentemente das outras etapas do Filtro de Partículas, a Reamostragem não precisa, e nem deve, ser realizada a cada iteração do algoritmo, apenas quando se torna necessária de acordo com algum critério pré-estabelecido. A principal razão para isso está na eliminação aleatória de partículas, que gera um empobrecimento da diversidade de estados do robô. Esse empobrecimento faz com que a variância do conjunto $\boldsymbol{S_t}$ diminua, pois as partículas se aglutinarão em uma única área, com pouca diferença entre os seus estados. Ao mesmo tempo, esse processo aumenta a variância do Filtro de Partículas como um estimador da variável de interesse, o que indica uma maior incerteza quanto ao resultado final.

Um exemplo extremo disso seria o caso de um robô imóvel ($\boldsymbol{x_t = x_{t+1}}$) e que não possui nenhum sensor para coletar informações do ambiente. No início existe uma incerteza quanto à sua localização, fazendo com que suas partículas se espalhem por uma determinada área. A aleatoriedade da Reamostragem faz com que algumas partículas sejam escolhidas em detrimento de outras, até que eventualmente a covariância do conjunto de partículas se anule, pois ele será composto por *P* cópias de uma única partícula. Essa convergência de partículas indica que o estado correto de localização foi encontrado, o que está em clara contradição com o simples fato de que o robô não possui sensores que permitam determinar a sua localização. Na realidade, essa convergência aconteceu devido a fatores puramente aleatórios que ocorrem durante a Reamostragem, ou seja, a covariância dessa estimativa é infinita.

É necessário então controlar a magnitude da variância do Filtro de Partículas, de forma a permitir o seu funcionamento e garantir a precisão das suas estimativas a cada instante. Em (Liu, Chen, Logvinenko; 2001) são apresentadas duas maneiras de se estimar a quantidade de partículas cujo peso é irrelevante para o processo em um determinado instante, indicando quando a Reamostragem deve ser realizada. A primeira é através do Coeficiente de Variação cv_t^2 (4.50), que quantifica a variância do conjunto $\boldsymbol{S_t}$, e a segunda é através do Tamanho Efetivo da Amostra ESS_t (4.51), que quantifica a quantidade de partículas que são significantes para a distribuição de probabilidades.

$$cv_t^2 = \frac{var\left(w_t^i\right)}{E^2\left(w_t^i\right)} = \frac{1}{P}\sum_{i=1}^{P}\left(Pw_t^i - 1\right)^2 \tag{4.50}$$

$$ESS_t = \frac{P}{1 + cv_t^2} \tag{4.51}$$

Um aumento no valor de cv^2 implica em uma variação maior entre os estados das partículas de $\boldsymbol{S_t}$, o que significa que elas estão se afastando cada vez mais, gerando uma densidade local menor. Da mesma forma, uma diminuição no valor de ESS_t significa uma menor quantidade de partículas que possuem um peso significativo, e com isso alguma chance de representar o estado correto da localização do robô naquele instante. Quando uma dessas duas variáveis cruza um determinado limiar, é realizada a Reamostragem.

4.5. Paralelização dos Algoritmos

O algoritmo apresentado nesse trabalho, da maneira como foi proposto, realiza cada uma das suas etapas seqüencialmente, aguardando os resultados de uma antes de seguir para a próxima. Inicialmente o robô se movimenta e então coleta uma imagem omnidirecional, que é processada pelo SIFT em busca de marcos que podem ser utilizados. Esse conjunto de marcos extraídos é comparado com aqueles armazenados no mapa em busca de correspondências e aqueles que forem correspondidos são utilizados para refinar as estimativas de cada partícula, enquanto aqueles que não foram são incorporados ao mapa. Para que essas informações possam ser utilizadas durante a navegação, é necessário que todo esse processo aconteça no intervalo entre coleta de dados, tanto de localização como de mapeamento, para evitar que o algoritmo perca sua sincronia com o robô e deixe de acompanhá-lo conforme ele se movimenta pelo ambiente.

Contudo, o algoritmo de SIFT possui um custo computacional elevado, que juntamente com a etapa de busca por correspondência pode fazer com que o tempo necessário para que todos os cálculos sejam realizados seja maior do que o intervalo entre iterações dos sensores. Uma solução é aumentar esse intervalo, de maneira a fornecer mais tempo para que os cálculos possam ser feitos. Mas essa solução diminui a controlabilidade do robô, pois as informações de localização se tornam mais espaçadas entre si, dificultando o processo de tomada de decisões durante a navegação. Por exemplo, ele pode ter alcançado o seu objetivo final, mas devido ao atraso na atualização da sua localização ele permanece se movimentando e o ultrapassa, entrando em um processo oscilatório em torno de um ponto.

A solução para esse problema utilizada nesse trabalho consiste na paralelização dos algoritmos de SIFT e de FastSLAM, pemitindo que o robô se localize com a freqüência necessária enquanto realiza ao mesmo tempo os cálculos de extração de marcos na em imagens omnidirecionais. Durante esse intervalo considera-se que o robô não obteve nenhuma informação do ambiente e, portanto, apenas a etapa de Predição será realizada iterativamente, com a sua localização se propagando no tempo. Dessa forma a incerteza referente a ela aumenta, pois os erros gerados a cada instante se acumulam. Quando os cálculos referentes ao SIFT são completados e o conjunto de marcos é obtido, a informação referente a eles é incorporada às estimativas de localização e mapeamento, eliminando os erros acumulados durante a navegação cega e restabelecendo a precisão dos resultados.

5. Resultados e Discussão

Esse capítulo apresenta os resultados obtidos nas etapas de implementação realizadas visando a obtenção da solução para o problema do SLAM como proposto no capítulo anterior. Em um primeiro momento essa solução foi dividida em duas etapas independentes entre si, que foram desenvolvidas paralelamente em ambientes controlados que permitissem o isolamento e quantificação dos erros envolvidos de maneira a verificar a capacidade de cada uma em resolver o problema ao qual se propõe. A primeira etapa é composta pelo algoritmo de SLAM propriamente dito, que lida com a maneira como a informação obtida a partir do sistema de reconhecimento de marcos é incorporada às estimativas de localização e mapeamento do robô. A segunda etapa é composta pelo algoritmo de reconhecimento de marcos, que lida com a obtenção da informação que será utilizada pelo algoritmo de SLAM sob a forma de pontos de referência durante a navegação.

Os resultados individuais obtidos em cada uma dessas etapas são apresentados como uma forma de se validar as soluções individuais escolhidas para se resolver cada um desses problemas, assim como as ferramentas e métodos utilizados. Pontos pertinentes que foram previstos durante a revisão bibliográfica ou que surgiram durante o desenvolvimento das soluções são também apresentados e discutidos, procurando-se com isso esclarecer a influência que eles possuirão sobre os resultados finais. Uma vez que ambos os algoritmos estivessem funcionando de maneira satisfatória, dados os resultados alcançados em testes controlados, seguiu-se para a última etapa, que se constituiu na união de ambos em uma solução fechada para o problema do SLAM da maneira como proposta no capítulo anterior, capaz de obter informações do ambiente e incorporá-la às estimativas durante a navegação. Essa solução foi submetida a testes reais de navegação, e os resultados alcançados são apresentados como uma maneira de validá-la como uma solução para o SLAM com Auxílio Visual Omnidirecional.

5.1. Localização e Mapeamento Simultâneos

A implementação isolada do algoritmo de SLAM foi realizada em ambientes virtuais com o auxílio do *Gazebo*, um simulador de robôs tridimensional *open-source* que funciona em plata-

formas Linux e MacOS. A comunicação entre algoritmo e robô é feita através do *Player*, uma interface que lida de maneira indistinta com robôs virtuais ou reais, permitindo uma transição rápida entre os testes feitos no simulador e a navegação em ambientes reais. A utilização de simuladores nas etapas iniciais de um algoritmo de navegação é importante por fornecer um ambiente controlado e seguro para os testes, além de facilitar a construção de percursos complexos e permitir a quantificação da melhoria nos resultados que um algoritmo de SLAM é capaz de proporcionar.

5.1.1. Simulador

Dentro do ambiente *Gazebo* foi construído inicialmente um modelo dinâmico de um robô não-holonômico com atuadores diferenciais, equivalente ao de um robô real existente (um *Magellan Pro* da *iRobot)*. As equações de odometria foram adicionadas de maneira a fornecer ao robô uma estimativa incremental de localização conforme se movimenta pelo ambiente. Também foram desenvolvidas rotinas de navegação que permitiram ao robô se movimentar de um ponto ao outro de maneira autônoma, de acordo com a teoria de controle ótimo apresentada em (Davis, Vinter; 1985). A aquisição de informações foi feita a uma taxa de 10 Hz, e estabeleceu-se que o algoritmo de SLAM final deveria ser capaz de manter a sincronia de comunicação com o robô dentro dessa freqüência de atualização.

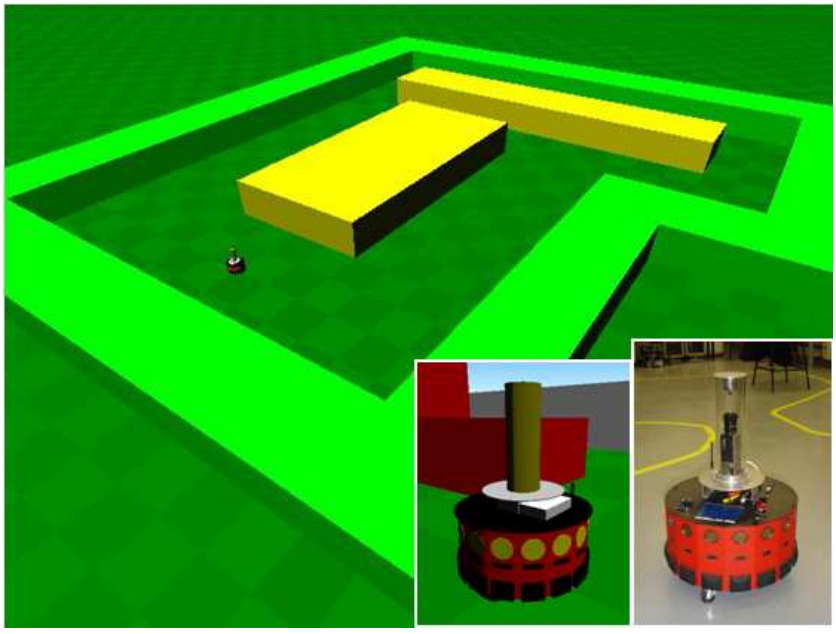


Figura 5.1 - Exemplo de modelo de robô e de ambiente no Gazebo.

O modelo construído foi equipado com um sensor laser que varre uma área de 180º à sua frente, com um alcance de 3 metros. Esse sensor foi utilizado na construção de um mapa métrico probabilístico do ambiente, com discretização de 50 milímetros e que não influencia de forma alguma no SLAM, sendo utilizado apenas como visualização do resultado de mapeamento (e, conseqüentemente, de localização). Como o Gazebo não possui suporte para modelos de espelho não foi possível simular o sistema de visão omnidirecional, pois ele depende de um espelho hiperbólico posicionado sobre a câmera. Esse sistema foi então substituído provisoriamente nos testes em ambientes virtuais por um segundo sensor laser que varre uma área de 360º ao redor do robô com um alcance de 4 metros, posicionado a uma altura de 0.5 metros acima do primeiro laser.

Postes elevados foram espalhados pelo ambiente de forma que possam ser detectados por esse sistema como mínimos locais (Figura 5.2) e utilizados como marcos artificiais. A correspondência entre marcos é feita pela minimização da distância euclidiana entre o conjunto observado e aquele armazenado no mapa de características do robô. Embora esse arranjo permitisse a obtenção direta de informações de distância, optou-se por abrir mão dessa informação e obtê-la por meio de triangulações, para tornar o algoritmo mais similar àquele utilizado em conjunto com o sistema de visão omnidirecional. Essa informação de distância foi, ao invés disso, utilizada na caracterização dos marcos, que devido à natureza da informação obtida a partir do sensor de laser foi feita tendo como base apenas a sua posição no ambiente em relação ao robô.

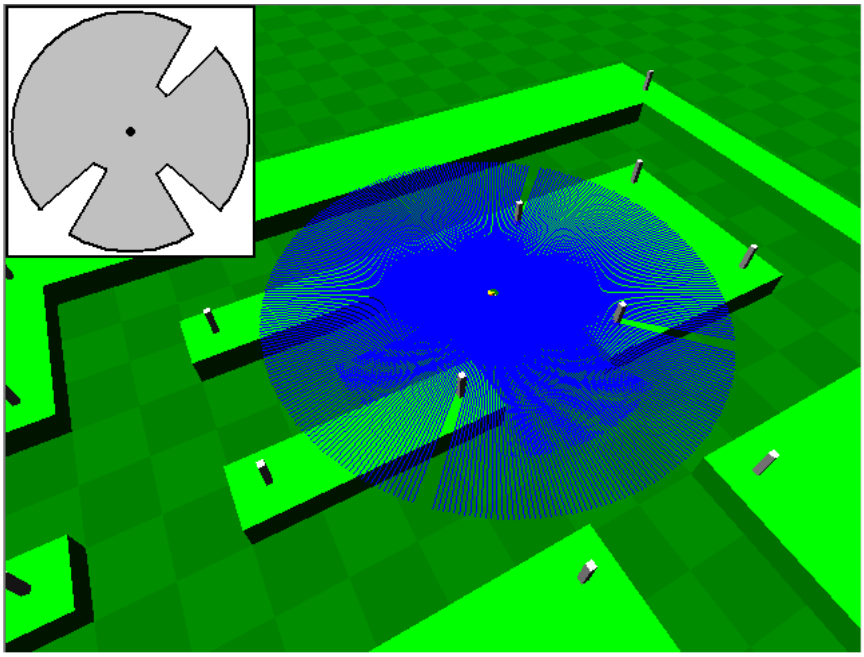
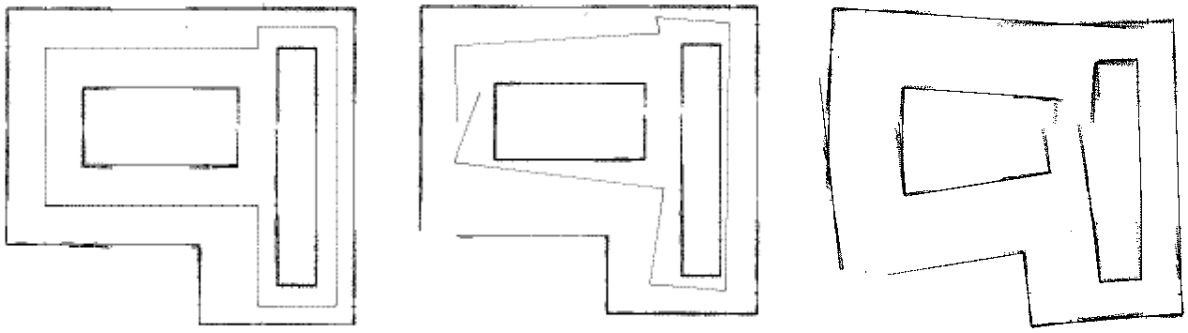


Figura 5.2 - Detecção de marcos no ambiente virtual.

Um ambiente virtual não contém os erros inerentes aos sensores utilizados na localização e no mapeamento que fazem com que o SLAM seja necessário. Por isso, esses erros foram modelados, tendo como base um robô real (o próprio *Magellan Pro*) e incorporados às medições realizadas pelo seu sistema de odometria e de lasers (tanto o superior quanto o inferior), de maneira a torná-los imprecisos. Nessa modelagem foram utilizadas distribuições gaussianas cuja média é igual ao valor fornecido pelo cálculo determinístico e o desvio padrão se relaciona com a precisão inerente ao sensor. Os parâmetros utilizados nessa etapa não são utilizados durante o tratamento de erros (que também foram modelados como distribuições gaussianas), pois em aplicações reais não há como conhecê-los. A Figura 5.3 mostra o efeito que a introdução desses erros possui sobre os resultados finais alcançados com a utilização direta da informação coletada pelos sensores.



(a) *(b)* *(c)*

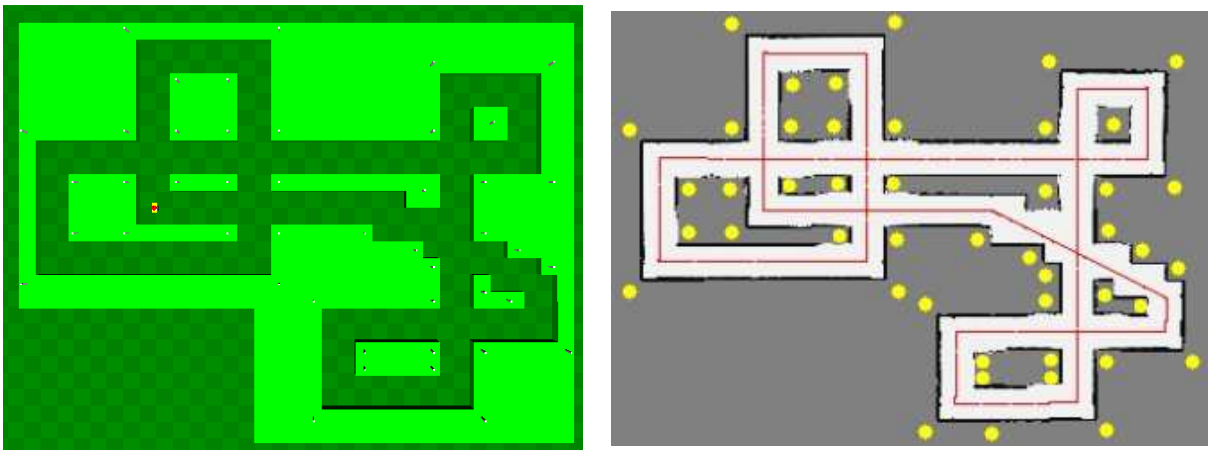
Figura 5.3 - Modelo de erros em ambientes virtuais (Guizilini et al; 2007).
(a) Localização e mapeamento sem erros (b) Localização com erros (c) Mapeamento com erros

Pode-se perceber que a utilização direta da informação obtida pelos sensores já não é suficiente para se obter resultados precisos de localização e mapeamento, devido ao acúmulo de erros que a navegação iterativa acarreta e que se reflete em ambas as estimativas. Nesse ambiente impreciso já existe a necessidade de um algoritmo de SLAM como uma maneira de se eliminar esses erros antes que eles se acumulem a ponto de invalidar o resultado final.

5.1.2. FastSLAM

Dentro do ambiente *Gazebo* foi construído um percurso fechado (Figura 5.4a) para a navegação do robô, composto por diversos corredores e com postes espalhados pelo ambiente que agem como marcos pré-estabelecidos, cujas posições não são conhecidas pelo robô no início da navegação. A Figura 5.4b mostra o resultado de localização e mapeamento quando o mo-

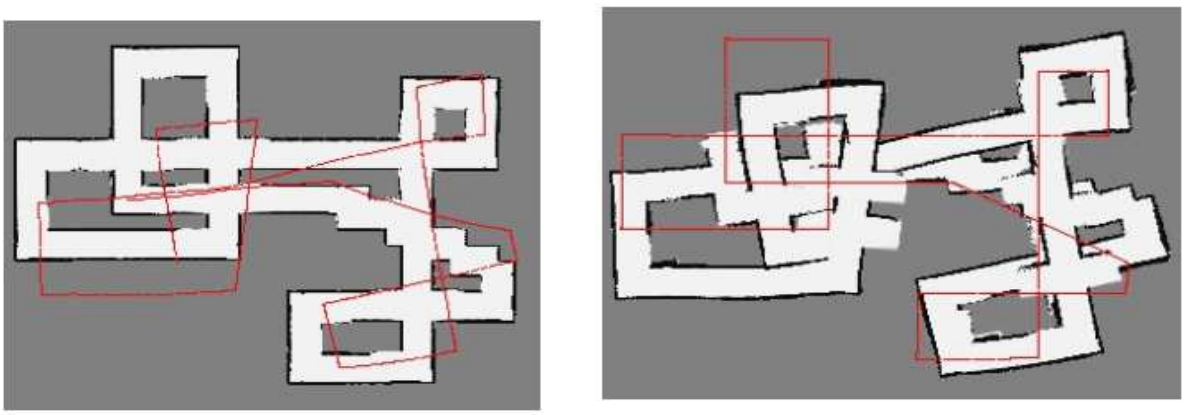
delo de erros não é utilizado, ou seja, quando os sensores fornecem resultados exatos. O robô iniciou a navegação no ponto indicado na Figura 5.4a e se deslocou inicialmente para a direita, percorrendo os corredores até retornar a posição inicial, em um percurso aproximado de 150 metros. O mapa métrico é construído a partir das informações do laser inferior e o mapa de características (que representa os marcos como círculos amarelos) é construído a partir das informações do laser superior.



(a) *(b)*

Figura 5.4 - Primeiro percurso virtual.
(a) Mapa original (b) Localização e mapeamento sem erros

Como já era de se esperar, os resultados de localização e mapeamento obtidos se aproximaram muito da realidade, limitados apenas pelos erros de aproximação que tanto o simulador como o algoritmo possuem. Em seguida, o modelo de erros dos sensores (odometria e laser) foi adicionado e o robô percorreu novamente a trajetória definida. Durante essa navegação foi utilizada a informação exata de localização de forma a impedir a colisão e permitir que o percurso fosse completado, enquanto a odometria era submetida ao acúmulo de erros.

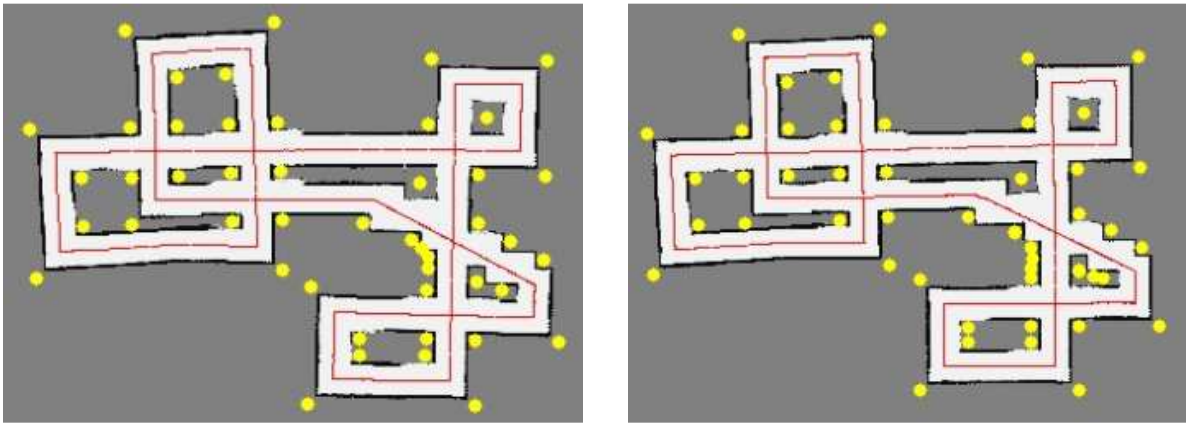


(a) *(b)*

Figura 5.5 - Localização e mapeamento com erros no primeiro percurso virtual.
(a) Localização com erros (b) Mapeamento com erros

Pode-se perceber claramente a influência que os erros possuem sobre os resultados finais de localização e mapeamento. O acúmulo de erros de odometria se reflete na incapacidade do robô de retornar até o ponto inicial depois de completar a trajetória proposta (Figura 5.5a). A parcela dos erros de mapeamento referente à imprecisão do laser, por não ser cumulativa, pôde ser eliminada pela abordagem probabilística utilizada no mapa métrico, gerando uma alta definição nos contornos das paredes que foram mapeadas. Contudo, a parcela dos erros de mapeamento referente à imprecisão de localização do robô não pôde ser eliminada, e os erros de localização se propagaram para o mapa, refletindo-se na falta de paralelismo entre corredores e repetição de estruturas (Figura 5.5b).

Esse mesmo percurso foi realizado novamente, utilizando inicialmente o FastSLAM 1.0 e depois o FastSLAM 2.0, e os resultados são mostrados na Figura 5.6. Em ambos os algoritmos foi incorporado também um sistema de eliminação de marcos gerados por erros de correspondência que atribui um índice de existência para cada um dos marcos armazenados. Sempre que um marco é observado o seu índice aumenta, e quando a sua posição se encontra dentro da área dos sensores e ela não é observada o índice diminui. Caso se torne negativo o marco é excluído do mapa e não é mais utilizado no FastSLAM.

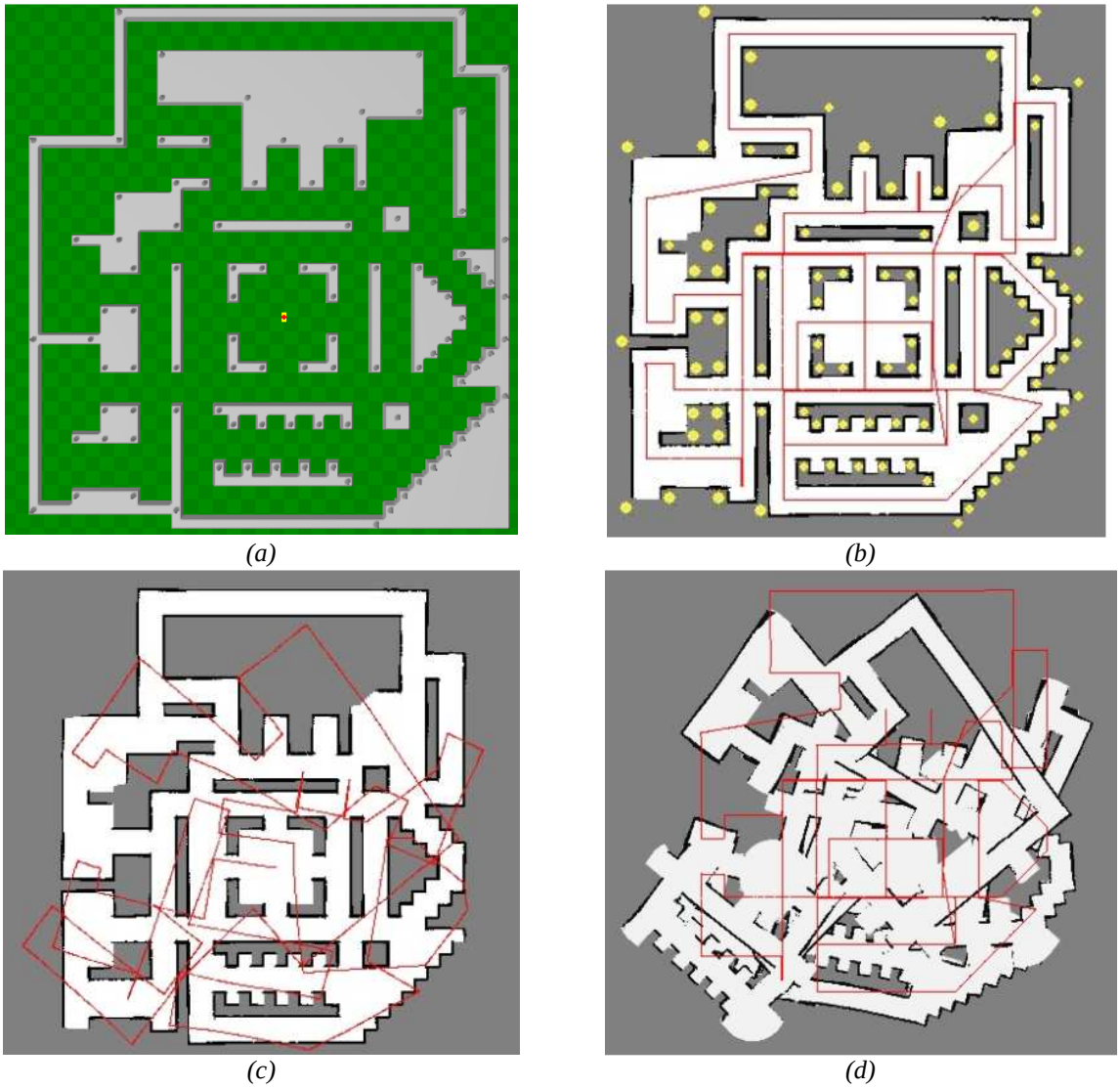


(a) *(b)*

Figura 5.6 - Resultados do FastSLAM no primeiro percurso.
(a) FastSLAM 1.0 (50 partículas) (b) FastSLAM 2.0 (10 partículas)

A melhoria nos resultados obtidos na Figura 5.6 é substancial, mostrando que um algoritmo de SLAM é realmente necessário em tarefas de localização e mapeamento simultâneos que apresentam erros que não podem ser modelados. O mapa resultante e a trajetória se mostram muito próximos da realidade (Figura 5.4b) e podem ser utilizados em aplicações posteriores que requerem resultados precisos. Em especial, pode-se perceber que o robô começou a percorrer o corredor inferior esquerdo de maneira desalinhada, mas tão logo reconheceu um

marco ele conseguiu recuperar a precisão na sua localização e se alinhar com o corredor já mapeado. Como esperado o FastSLAM 2.0 não aumentou a qualidade dos resultados, mas sim permitiu que uma mesma qualidade pudesse ser obtida com menos partículas. Testes realizados com conjuntos maiores de partículas em ambos os algoritmos mostraram resultados semelhantes, indicando que a Figura 5.6 é a melhor estimativa que o FastSLAM consegue fornecer para a localização e mapeamento de acordo com as condições de erro estabelecidas e a maneira como ele aborda o problema.



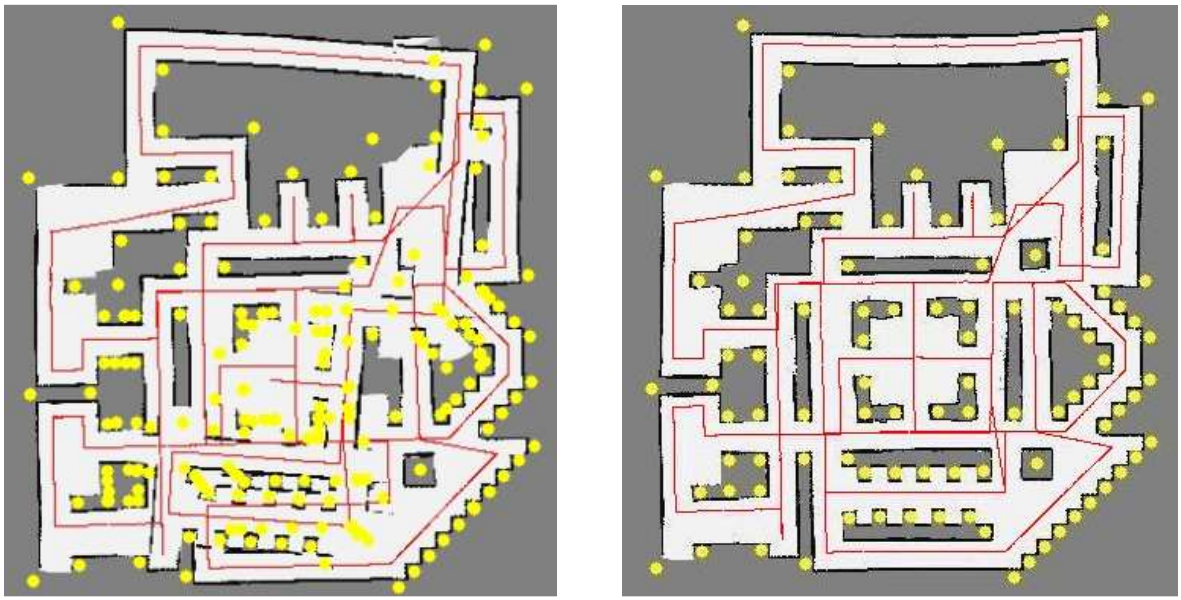
(a) *(b)* *(c)* *(d)*

Figura 5.7 - Segundo percurso virtual.
(a) Mapa original (b)Localização e mapeamento sem erros
(c) Localização com erros (d) Mapeamento com erros

Para testar os limites do FastSLAM foi construído (Figura 5.7) um outro ambiente virtual, com um percurso de navegação aproximadamente dez vezes maior (1.5 km) e com significativamente mais marcos. O diâmetro de cada um dos postes utilizados como marcos também

foi aumentado, gerando uma maior imprecisão quanto à sua posição ao ser detectada em diferentes ângulos. Adicionalmente, como um desafio para o SLAM foi incluído um corredor (extremo superior, com entrada pelo lado esquerdo) onde não há nenhum marco para ser detectado. Enquanto está atravessando esse corredor nenhuma informação será incorporada às estimativas do robô, e os erros de localização e mapeamento se acumularão até que o corredor termine e ele possa voltar a detectar marcos e refinar suas estimativas. Espera-se que nesse momento o robô consiga recuperar a precisão na sua localização e também corrigir as estimativas das posições dos observados antes dessa convergência ocorrer.

Novamente, os resultados de localização e mapeamento obtidos com o uso direto da informação coletada pelos sensores do robô servem apenas para reiterar a necessidade de se utilizar algoritmos de SLAM nessas situações. O mesmo percurso foi realizado novamente, agora com o FastSLAM 1.0 e 2.0, e os resultados são mostrados na Figura 5.8.



(a) *(b)*

Figura 5.8 - Resultados do FastSLAM no segundo percurso.
(a) FastSLAM 1.0 (50 partículas) (b) FastSLAM 2.0 (50 partículas)

Pode-se perceber que até o início do corredor os erros referentes às estimativas de localização e mapeamento se mantinham estáveis e de acordo com aqueles obtidos no percurso anterior, mesmo levando em consideração uma navegação mais prolongada. Contudo, ao sair do corredor o FastSLAM 1.0 (Figura 5.8a) não foi capaz de recuperar a precisão de localização a partir dos marcos observados devido à baixa caracterização que eles possuem no sistema de detecção utilizado. Essa caracterização é feita tendo como base a posição do marco no ambiente relativamente ao robô, e por isso erros tanto de localização quanto de mapeamento

dificultam a correspondência. Ao sair do corredor a localização do robô havia acumulado tantos erros que ele não conseguiu reconhecer os marcos observados, e ao invés disso começou a gerar um outro conjunto de marcos, eliminando aqueles que estavam corretos através do sistema de verificação de existência e passando a se orientar a partir das correspondências erradas. Esse erro então se consolidou e se propagou pelo restante da navegação, criando uma inconsistência entre os resultados obtidos antes e depois do corredor ter sido atravessado, que se reflete na repetição de estruturas observadas nesses dois instantes.

Já o FastSLAM 2.0 conseguiu recuperar a precisão (Figura 5.8b) depois que o robô atravessou o corredor, e com isso manter a consistência de resultados durante todo o período de navegação. Isso se deve à sua maior eficiência na representação da distribuição de probabilidades de localização do robô, que fez com que ele conseguisse propagar o estado correto a partir de um pequeno conjunto de partículas que aleatoriamente se mantiveram dentro da tolerância de reconhecimento de marcos estabelecida. Essas partículas conseguiram reconhecer os marcos observados e refinar as suas hipóteses de localização e mapeamento, e ao receberem pesos maiores se propagaram para as iterações posteriores enquanto aquelas com correspondências erradas eram eliminadas. A utilização de um maior número de partículas aumenta a probabilidade de restar um conjunto dentro dessa região após a navegação cega e o acúmulo de erros, e testes realizados mostraram que o FastSLAM 1.0 consegue manter a precisão com conjuntos de pelo menos 200 partículas.

Os resultados obtidos nessa etapa validam o FastSLAM como uma solução para o problema de se incorporar as informações obtidas a partir dos sensores do robô às suas estimativas de localização e mapeamento de maneira a eliminar os erros acumulados durante a navegação, e permitiram a sua utilização no algoritmo final proposto. A substituição do sistema de sensoriamento a laser pelo sistema de visão omnidirecional trouxe uma série de benefícios que eliminou grande parte das dificuldades encontradas nos testes preliminares aqui apresentados, permitindo a obtenção de resultados mais precisos em um conjunto maior de situações e aplicações. Um sistema de visão permite a obtenção de marcos naturais, eliminando a necessidade de posicionar estruturas específicas no ambiente previamente à navegação e aumentando a quantidade de informações que o robô coleta a cada iteração. A caracterização de marcos que uma imagem proporciona permite uma correspondência mais robusta que não depende das estimativas de localização e mapeamento, e com isso o robô torna-se capaz de se recuperar mais facilmente de situações de alta imprecisão.

5.2. Auxílio Visual Omnidirecional

Os primeiros testes realizados com o SIFT foram feitos tendo como base imagens convencionais, dado que esse algoritmo foi inicialmente desenvolvido para esse tipo de imagem e depois modificado para outros tipos de geometria. Foram realizados testes de extração, descrição e correspondência de características, tendo como base os parâmetros apresentados por (Lowe; 2004), onde são escolhidos por gerarem resultados eficientes e estáveis em uma grande gama de aplicações. Posteriormente os mesmos testes foram realizados nas imagens omnidirecionais que seriam utilizadas na solução final, e esses parâmetros foram modificados visando a obtenção de melhores resultados nessas condições. Os algoritmos de triangulação e determinação de marcos, por dependerem diretamente da geometria do sistema de visão, foram implementados diretamente em imagens omnidirecionais já visando a sua utilização nos testes finais.

5.2.1. Imagens Convencionais

Os testes iniciais de detecção e extração de características em imagens convencionais foram realizados com o auxílio de imagens de figuras geométricas, como uma forma de se reconhecer se as características estavam sendo detectados de maneira correta. Nessas figuras simples é possível prever onde elas se localizarão, e com isso julgar se o resultado final é consistente. Um exemplo de extração de características em figuras geométricas é mostrado a seguir (Figura 5.9), onde os pontos vermelhos indicam as características extraídas.

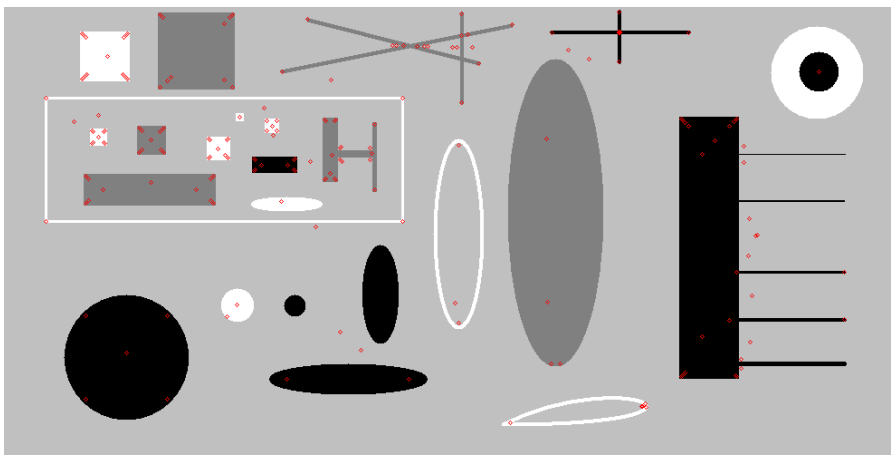
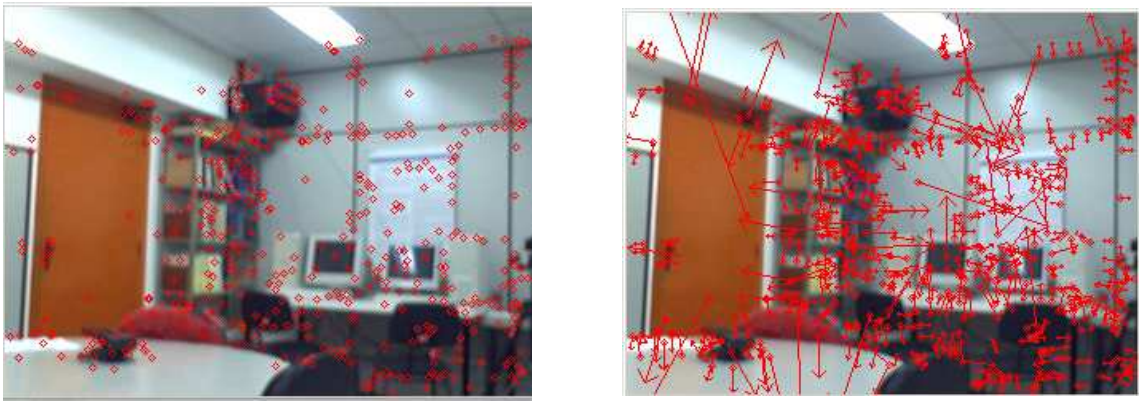


Figura 5.9 - Extração de características em formas geométricas.

.

É possível perceber a consistência entre as características que compõem cada um dos objetos da imagem. Círculos são caracterizados unicamente pelo seu centro, e conforme se alonga em uma elipse passa a ser caracterizado por dois pontos, os seus focos. Quadrados e retângulos são definidos pelos seus cantos e centro e retângulos muito alongados são tratados como elipses, com o surgimento de dois "focos" ao invés de um único centro. Segmentos de reta são caracterizados em suas extremidades e intersecções. Regiões vazias localizadas entre dois objetos podem ser caracterizadas devido à suavização gaussiana que mistura dois objetos em escalas mais elevadas. Contudo, testes de correspondência entre características coletadas nesse tipo de imagem não se mostraram promissores, devido à sua escassez e à semelhança entre as características de dois objetos geométricos simples, o que gera uma grande quantidade de correspondências erradas.

Para a obtenção de imagens reais do ambiente para testes foi utilizada uma webcam comum, que coletava imagens de tamanho 320x240 pixels utilizadas diretamente pelo SIFT, mantendo os mesmos parâmetros utilizados anteriormente. A Figura 5.10 mostra as etapas de extração de características e atribuição de orientação e magnitude (o sentido das setas indica a orientação e o tamanho, a magnitude). Pontos com mais de uma seta representam regiões que se multiplicaram durante a etapa de atribuição de orientação realizada com o seu histograma local, dando origem a mais de uma característica.



(a) *(b)*

Figura 5.10 - Extração de características em imagens convencionais.
(a) Características extraídas (b) Magnitude e orientação

Cada uma dessas características foi descrita para correspondência posterior, e como durante a navegação supõe-se que os objetos permanecerão imóveis e apenas o robô se movimentará, a maioria dos testes foi realizada em ambientes estáticos, movimentando apenas a câmera de forma a obter imagens dos mesmos objetos em diferentes pontos de vista. A Figura 5.11 mos-

tra o resultado de correspondência para alguns objetos escolhidos manualmente na imagem à esquerda (características limitadas pelos retângulos azuis), com as linhas verdes indicando as correspondências realizadas com a imagem à direita. Todo o processo de extração, descrição e correspondência foi realizado em aproximadamente 1.2 segundos, e é possível perceber que o SIFT consegue resultados precisos nessa tarefa.

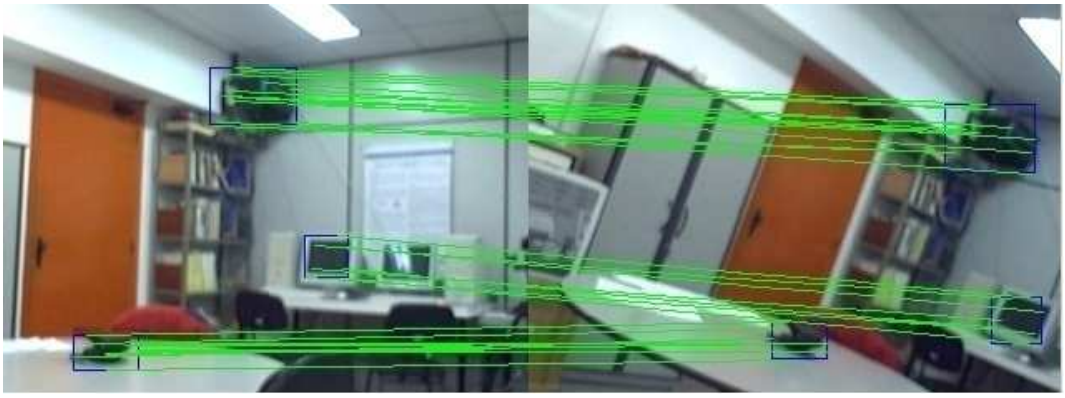


Figura 5.11 - Correspondência entre objetos.

Outros testes de correspondência de objetos foram realizados, agora considerando efeitos como oclusão e mudança de disposição dos objetos, como mostrado na Figura 5.12. Inicialmente o algoritmo recebeu uma imagem isolada de cada um dos objetos que ele deveria procurar e extraiu as características que os compõem, agrupando-os em conjuntos que representam cada um dos objetos em específico.

Figura 5.12 - Reconhecimento de objetos.

Dessa forma, ao receber outra imagem do ambiente o SIFT pode procurar por esse padrão e reconhecer cada um dos objetos de uma forma muito mais precisa do que seria feito caso ele estivesse correspondendo características isoladamente. O algoritmo final utilizou um processo semelhante, com a diferença que não há nenhuma informação inicial referente à natureza dos objetos que serão utilizados, eles são determinados a partir das próprias imagens coletadas de maneira a eliminar a necessidade de conhecimento prévio do ambiente e aumentar a genericidade do algoritmo. Ao corresponder uma determinada quantidade de características (três, como proposto por Lowe) o objeto é considerado correspondido e a sua forma passa por uma transformação que visa orientá-lo de acordo com sua projeção na imagem naquele instante, dando origem aos quadriláteros que delimitam a sua posição na imagem. Características não correspondidas pertencentes ao objeto que são mantidas para correspondências posteriores e são utilizadas para o cálculo espacial da deformação.

5.2.2. Imagens Omnidirecionais

O próximo passo foi a implementação do algoritmo em imagens omnidirecionais, obtidas a partir de um sistema de visão (Figura 5.13a) que utiliza um espelho hiperbólico usinado em um torno de precisão (Figura 5.13b). O sistema é montado sobre o robô de maneira a coincidir com o seu eixo de rotação, ou seja, a câmera não se desloca quando o robô rotaciona, apenas gira junto com ele. A geometria do espelho permite que o robô observe objetos arbitrariamente distantes, sendo limitado apenas por regiões de oclusão e pela resolução que o sistema omnidirecional apresenta e que se degrada conforme se aproxima das bordas.

(a) *(b)*

Figura 5.13 - Sistema de visão omnidirecional (Grassi Jr., Okamoto Jr.; 2006).
(a) Sistema montado (b) Espelho hiperbólico utilizado

Esse sistema foi calibrado a partir da determinação de pontos na imagem cujas posições no ambiente são conhecidas e procurando-se minimizar a distância entre as posições calculadas e as reais através da modificação dos parâmetros relevantes. A Figura 5.14 mostra um resultado de calibração usando como base o raio do robô, o raio máximo do espelho hiperbólico e o limite inferior de alguns objetos espalhados no ambiente a distâncias conhecidas.

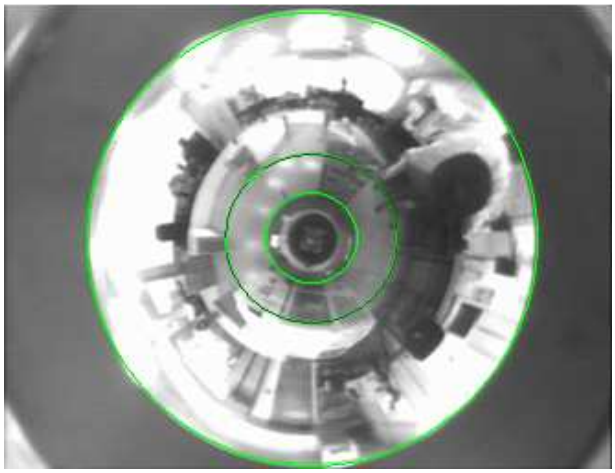
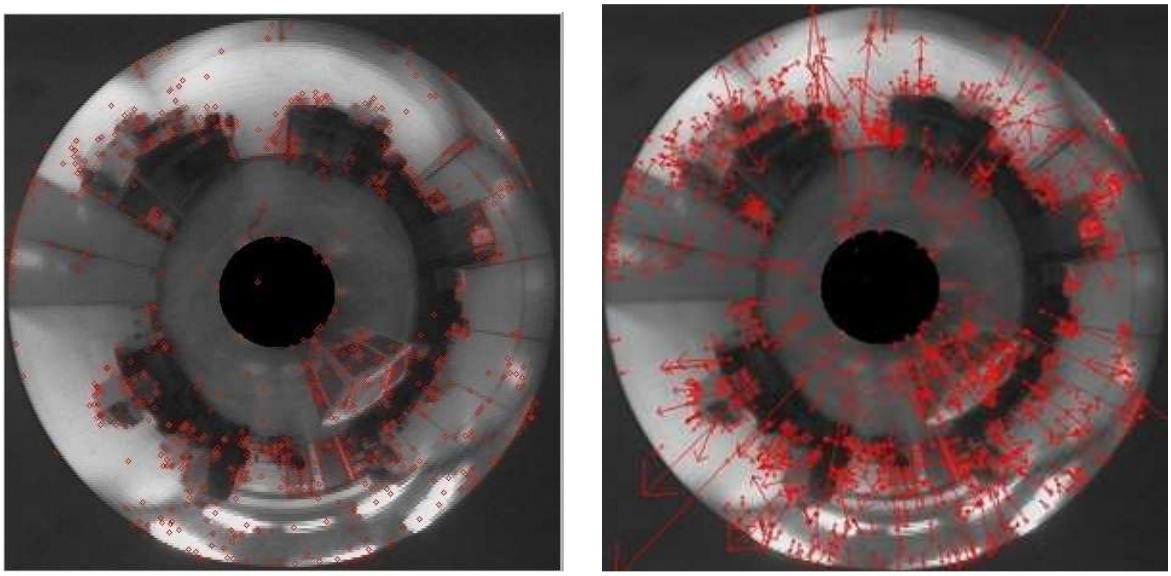


Figura 5.14 - Calibração do espelho.

Os resultados de extração de características e atribuição de magnitude e orientação em imagens omnidirecionais obtidas a partir desse sistema são mostrados na Figura 5.15. As regiões em preto internas e externas representam regiões não relevantes da imagem (a reflexão da câmera e o suporte do espelho, respectivamente) que foram removidas para evitar o surgimento de características que não representam o ambiente.



(a) *(b)*

Figura 5.15 - Extração de características em imagens omnidirecionais.
(a) Extração de características (b) Orientação e magnitude

Na Figura 5.15a é possível perceber que uma imagem omnidirecional continua gerando uma grande quantidade de características para cada um dos objetos que a compõem, e eles se distribuem de uma maneira similar ao que acontece em imagens convencionais. A diferença que uma imagem omnidirecional gera sobre o algoritmo de SIFT em relação a outros tipos de imagem está principalmente na orientação das suas características, que possuem uma tendência a apontar radialmente em relação ao centro da imagem, seja para fora ou para dentro, como mostrado na Figura 5.15b.

Na Figura 5.16 é apresentado um resultado de correspondências entre imagens omnidirecionais, da mesma forma que foi feito na Figura 5.11. O ambiente permaneceu estático e a câmera se movimentou paralelamente à caixa, percorrendo aproximadamente 3 metros e mudando em 90º o seu ângulo de visão em relação a ela. Todas as características extraídas da imagem da esquerda foram comparadas com todas aquelas obtidas na imagem da esquerda em busca de correspondência, e foram escolhidas algumas regiões (delimitadas pelos quadrados azuis na imagem à esquerda) para representar as correspondências realizadas (indicadas pelas linhas verdes que conectam características). Como é possível perceber, a qualidade dos resultados de correspondência entre objetos em imagens omnidirecionais é similar àquela obtida quando são utilizadas imagens convencionais.

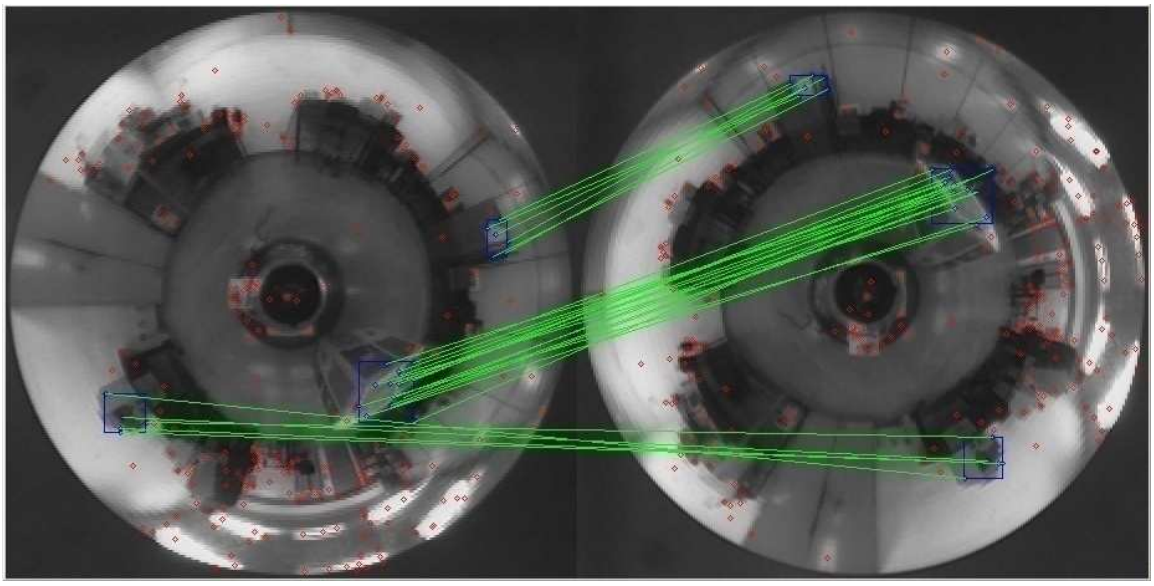


Figura 5.16 - Correspondência entre objetos em imagens omnidirecionais.

A Figura 5.17 apresenta os resultados obtidos com o algoritmo de determinação de marcos que agrupa as características extraídas da imagem em busca de grupos que possam ser correspondidos conjuntamente, visando diminuir a probabilidade de erros. Os círculos verdes

indicam as características extraídas da imagem que foram correspondidas com aquelas armazenadas no mapa do robô, e os círculos amarelos são marcos adicionados ao mapa naquele instante. Os retângulos verdes indicam os conjuntos de marcos que pertencem a um mesmo "objeto", determinado pelo robô a partir de informações de contraste e distância entre os seus pixels. As imagens mostradas estão separadas por um intervalo de 2 segundos (outras imagens foram obtidas e processadas nesse intervalo).

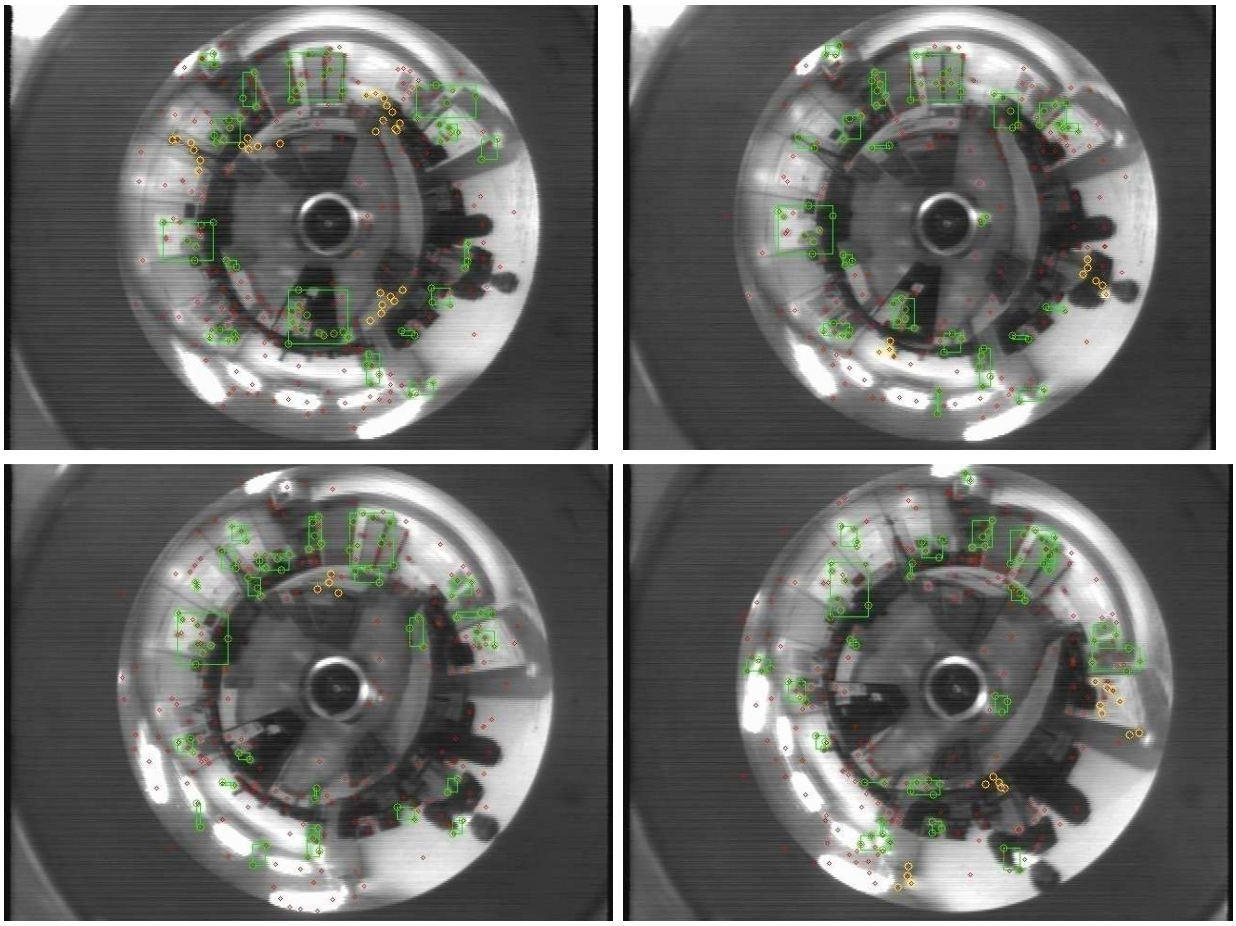


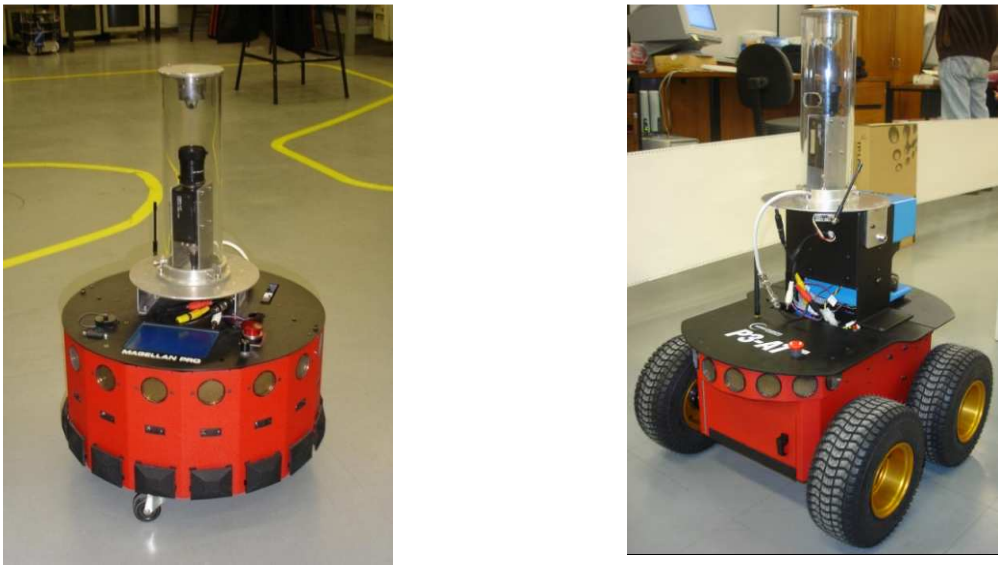
Figura 5.17 - Detecção de marcos em imagens seqüenciais (intervalo de 2 segundos entre frames).

A correspondência entre objetos obtida na Figura 5.16 e a determinação de marcos apresentada na Figura 5.17 validam a utilização do SIFT como um método de se extrair e corresponder marcos no ambiente que possam ser utilizados pelo robô durante a navegação. Esse mesmo sistema de visão foi utilizado nos testes finais, não sendo necessária nenhuma alteração. Durante a navegação as imagens omnidirecionais obtidas estarão vinculadas a uma localização (a do robô no instante em que a imagem foi coletada), e o conhecimento dessa localização aliado com a informação da orientação do marco em relação a essa localização permitem a triangulação, e com isso a estimação da posição do marco no ambiente e da localização do robô.

5.3. SLAM com Auxílio Visual Omnidirecional

A última etapa de implementação realizada nesse trabalho constituiu-se na união dos dois algoritmos apresentados anteriormente, o FastSLAM e o SIFT, em um único algoritmo capaz de cumprir todas as tarefas necessárias para se resolver o problema do SLAM (obter informações e incorporá-las às estimativas) da maneira como apresentado no capítulo anterior. Foram utilizados dois robôs, um Magellan Pro (Figura 5.18a) e um Pioneer 3AT (Figura 5.18b), que navegaram percursos distintos. Procurou-se com isso testar a capacidade do algoritmo de lidar com diferentes situações sem a necessidade de alterações em seu código, visando a obtenção de uma solução verdadeiramente genérica para o problema do SLAM que não dependa da plataforma utilizada.

Sobre ambos os robôs foi montado o mesmo sistema de visão omnidirectional, a ambos possuem também um sistema de odometria que fornece estimativas incrementais de localização. O mapeamento métrico no Magellan Pro foi feito com sensores de sonar (16 sensores espalhados igualmente ao redor do robô com um alcance máximo de 4 metros) enquanto o Pioneer 3AT utilizou separadamente sensores de sonar (12 espalhados igualmente nas partes frontal e traseira e 2 em cada lado, todos com alcance máximo de 5 metros) e um sensor laser (varredura de 180º à frente do robô, com resolução de 0.5º e alcance máximo definido de 30 metros).



(a) *(b)*

Figura 5.18 - Robôs utilizados nos experimentos.
(a) Magellan Pro (b) Pioneer 3AT

Os testes foram realizados em percursos montados dentro do LPA (Laboratório de Percepção Avançada), ao qual pertencem os robôs e onde esse trabalho foi conduzido. O processamento

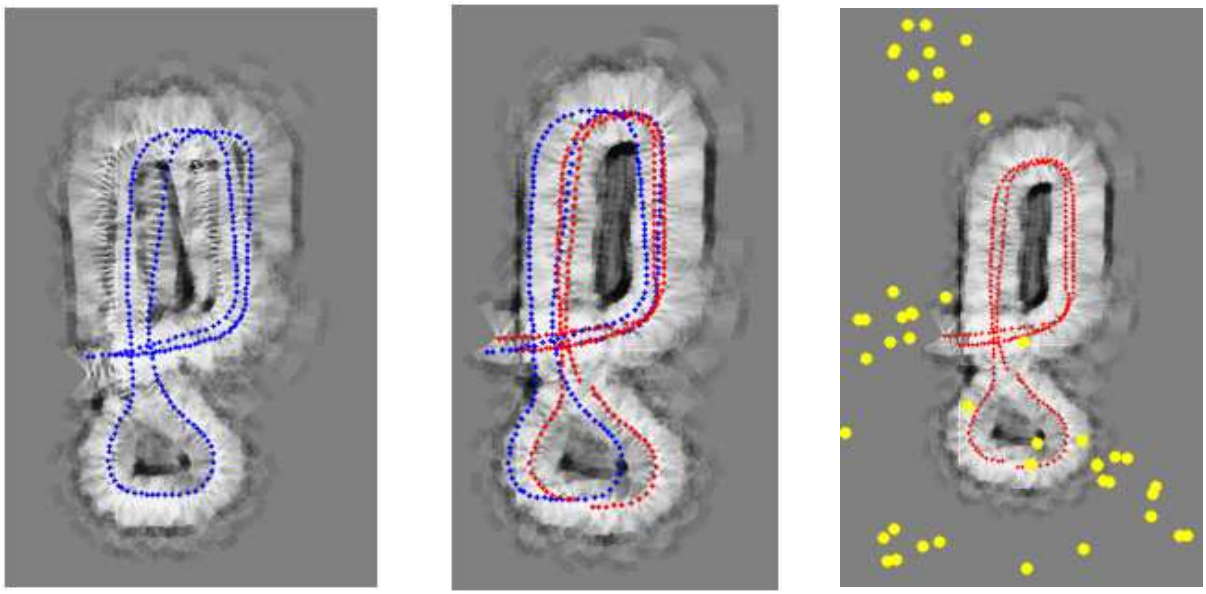
foi realizado em um computador externo aos robôs (um Pentium IV 2.0 GHz Core 2 Duo com Ubuntu 7.10) e a comunicação foi feita a partir de uma rede wireless (responsável pelo envio e recebimento de dados) e um transmissor via rádio (responsável pelo envio de imagens em uma estrutura CORBA). A navegação foi feita manualmente nas situações em que o algoritmo de SLAM não era utilizado, devido ao acúmulo de erros odométricos que impedia uma navegação segura, e de maneira autônoma nas situações em que esses erros eram tratados pelo algoritmo proposto. O processamento de imagens e construção de mapas foi feito com o auxílio das bibliotecas do OpenCV, e a comunicação entre programa e robô foi feita através do *Player*, instalado nos computadores embarcados do robô e acessado externamente pela rede de comunicação.

Os primeiros testes foram feitos com o MagellanPro, e para isso foi construída uma pista de testes (Figura 5.19) por onde o robô navegou enquanto eram coletados em intervalos regulares dados fornecidos pelo seu sistema de visão, odometria e sonares. Em um percurso de aproximadamente 40 metros com uma velocidade máxima de 0.2 m/s foram coletados 542 conjuntos de informação. Para testar a consistência de localização e mapeamento em diferentes instantes o robô atravessou várias vezes os mesmos corredores, gerando diferentes conjuntos de dados para uma mesma região que deveriam ser correspondidos. Os objetos que compunham a pista de testes e o laboratório (e que seriam utilizados pelo sistema de visão para a determinação de marcos) permaneceram estáticos durante a navegação, mas permitiu-se que houvesse movimentação natural de pessoas.

(a) *(b)*

Figura 5.19 - Pista de testes para o Magellan Pro.
(a) Foto (b) Mapa métrico da pista

Esse banco de dados de informações construído foi processado *off-line*, e tentou-se manter o tempo de processamento o mais próximo possível do tempo de navegação, de maneira a permitir que o algoritmo pudesse ser usado *on-line* durante a movimentação do robô pelo ambiente. O tempo necessário para o robô completar o percurso foi de aproximadamente 300 segundos, e o tempo de processamento do código (levando em consideração a paralelização de algoritmos que fez com que a informação de marcos não fosse utilizada em todas as iterações) foi de 268 segundos, indicando que seria possível processá-lo iterativamente durante a navegação. A Figura 5.20 apresenta os resultados obtidos nesse processamento de dados.



(a) *(b)* *(c)*

Figura 5.20 - Localização e mapeamento no primeiro percurso real.
(a) Sem SLAM (b) Com SLAM (c) SLAM e Mapeamento por Características

A Figura 5.20a mostra os resultados de localização e o mapa métrico construído a partir dos dados coletados quando o algoritmo de SLAM proposto não é utilizado. Pode-se perceber que os erros se acumulam de maneira que, quando o robô atravessa novamente os corredores superiores ele já não consegue reconhecê-los e os repete no mapa, assim como torna a caixa central superior irreconhecível. A utilização desses dados também faria com que o robô não conseguisse completar o percurso de maneira autônoma, pois ele colidiria com essa caixa como é indicado pela sua odometria. A resolução do mapeamento métrico se deve à utilização de sensores de sonar, que possuem uma alta dispersão angular e são imprecisos radialmente, criando uma grande quantidade de falsos negativos que impede a definição correta de bordas e regiões transversais ao movimento do robô, como é o caso dos cantos do percurso.

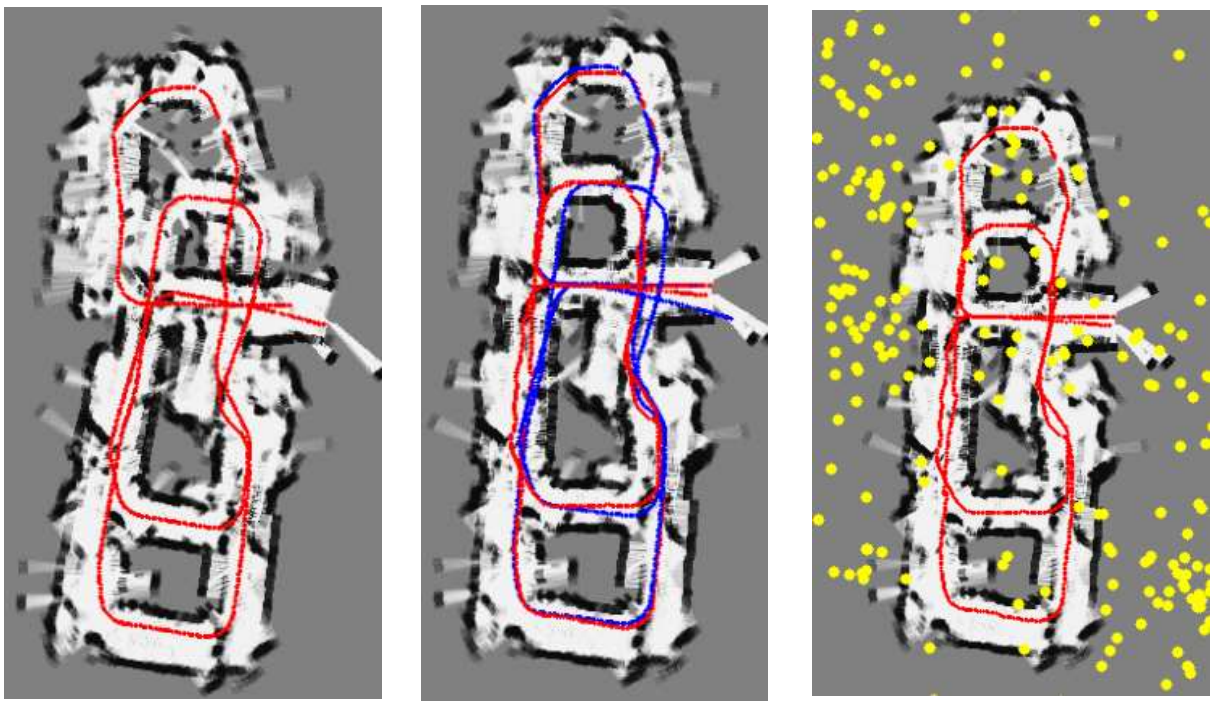
A Figura 5.20b mostra os resultados de localização (linha vermelha) e o mapa métrico construído a partir deles quando o algoritmo de SLAM proposto é utilizado, comparando-os com a localização obtida originalmente (linha azul). Pode-se perceber que o alinhamento de trajetórias e corredores é muito mais pronunciado, indicando que o robô foi capaz de reconhecer as regiões por onde já passou e com isso manter a consistência de seu mapeamento durante toda a navegação. As alterações bruscas de localização se devem ao acúmulo de incertezas durante os intervalos em que não há informação visual, que faz com que as partículas se espalhem por uma área maior do ambiente. Quando a informação visual é obtida os pesos das partículas são alterados e pode ser que uma partícula distante seja eleita a mais provável, fazendo com que o robô altere bruscamente a sua estimativa de localização. A Figura 5.20c apresenta o mapeamento por características dos marcos utilizados pelo robô, indicados por círculos amarelos (são apresentados apenas os marcos que foram correspondidos durante a navegação). É possível perceber claramente alguns aglomerados de marcos em determinadas regiões do ambiente onde existem estruturas, mas não foi possível determinar exatamente a quais objetos pertencem, pois o robô decide quais estruturas utilizar.

Em seguida foram realizados testes com o Pioneer 3AT, para o qual foi montada uma segunda pista de testes que foi percorrida de diversas maneiras em trajetórias de aproximadamente 70 metros. Nesses testes o algoritmo de SLAM foi utilizado simultaneamente à navegação e os resultados obtidos a cada instante eram utilizados pelo robô no cálculo da sua trajetória.

(a) *(b)*

Figura 5.21 - Pista de testes para o Pioneer 3AT.
(a) Foto (b) Mapa métrico da pista

Nesses testes o processamento dos dados foi feito *on-line*, ou seja, não houve armazenamento dos dados para utilização posterior, as informações fornecidas pelo sistema de odometria, visão e sensores de distância eram coletadas, processadas pelo algoritmo de SLAM e descartadas, enquanto as estimativas obtidas eram utilizadas pelo robô para orientar a sua navegação. Inicialmente foram utilizados os sensores de sonar para o mapeamento métrico, e os resultados são indicados na Figura 5.22. Pode-se perceber claramente o desalinhamento entre as regiões superiores e inferiores do percurso quando a navegação ocorreu sem o auxílio visual. Esse desalinhamento foi praticamente eliminado com o algoritmo de SLAM proposto, que manteve a consistência das estimativas durante toda a navegação e permitiu que o robô retornasse para o ponto de partida com uma maior precisão. O mapeamento por características mostra diversos aglomerados de marcos ao redor do percurso e em alguns dos obstáculos utilizados na sua construção, indicando uma consistência entre a posição dos marcos extraídos e as estruturas existentes no ambiente.



(a) *(b)* *(c)*

Figura 5.22 - Localização e mapeamento no segundo percurso real (sonar).
(a) Sem SLAM (b) Com SLAM (c) SLAM e Mapeamento por Características

Em seguida, foram realizados testes de mapeamento métrico com o sensor laser montado sobre o Pioneer 3AT. A utilização desse tipo de sensor permitiu uma definição muito maior das paredes que compunham o percurso, melhorando consideravelmente a qualidade do mapa métrico final obtido (Figura 5.23). O desalinhamento entre estruturas observadas em instantes

diferentes de navegação mostram o acúmulo de erros de odometria que impede o robô de retornar à sua posição inicial depois de cumprir o percurso. Especialmente na região superior do mapa, pode-se perceber o desalinhamento da parede diagonal à direita e da parede vertical à esquerda, e a dificuldade em se caracterizar as caixas que se encontram no meio do percurso devido ao acúmulo de informações conflitantes. Uma navegação autônoma baseada nessas informações novamente terminaria em colisão, impedindo que o percurso fosse completado.

(a) *(b)* *(c)*

Figura 5.23 - Localização e mapeamento no segundo percurso real (laser).
(a) Sem SLAM (b) Com SLAM (c) SLAM e Mapeamento por Características

A utilização do algoritmo de SLAM gerou um grande ganho de qualidade nas estimativas utilizadas durante a navegação, praticamente eliminando qualquer tipo de desalinhamento e fazendo com que a trajetória do robô fosse consistente durante todo o percurso. Em especial, pode-se perceber que o robô foi capaz de retornar exatamente para o ponto inicial, indicando que ele manteve a precisão de localização e de mapeamento durante todo o período de navegação. Novamente, os saltos bruscos nas estimativas de localização se devem a correções de erros acumulados nos instantes em que a informação visual não pôde ser obtida. O mapeamento por características gerou uma grande quantidade de marcos em objetos posicionados dentro do percurso, indicando consistência de resultados também nesse tipo de mapa.

6. Conclusão

Esse trabalho apresentou uma solução para o problema do SLAM que faz uso de um sensor de visão omnidirecional para detectar e reconhecer marcos no ambiente que o robô possa utilizar como pontos de referência para auxiliar a sua navegação por ambientes desconhecidos. Procurou-se com isso eliminar os erros de localização e mapeamento gerados por imprecisões nos sensores utilizados, que de outra forma se acumulariam e invalidariam os resultados finais obtidos. Um sensor de visão omnidirecional traz uma série de benefícios que foram explorados visando a obtenção de uma solução robusta que possa ser utilizada em qualquer tipo de ambiente, sem a necessidade de alterações prévias ou o conhecimento de suas estruturas.

A obtenção de marcos no ambiente é feita através do algoritmo de SIFT utilizado diretamente sobre as imagens omnidirecionais, gerando características que são agrupadas em conjuntos não-semânticos que permitem uma maior estabilidade nos resultados de correspondência e um menor custo computacional. A incorporação dessa informação nas estimativas de localização e mapeamento do robô é feita através do FastSLAM, cujo método de armazenagem de marcos foi modificado visando uma maior eficiência na manipulação da grande quantidade de informação que uma imagem omnidirecional pode gerar. O conhecimento da posição do robô no instante em que cada imagem foi obtida permite a triangulação das informações obtidas na correspondência e o cálculo da posição tridimensional da posição dos objetos ao seu redor.

As ferramentas utilizadas nessa solução foram implementadas inicialmente em ambientes controlados procurando com isso validar a capacidade de cada uma delas em resolver o problema ao qual se propõem. O método de extração de marcos apresentado foi implementado inicialmente em imagens convencionais e posteriormente adaptado para o sistema omnidirecional final. O algoritmo de SLAM proposto foi implementado em um simulador tridimensional e foram realizados testes que validam a sua capacidade de eliminar os erros acumulados de localização e mapeamento em situações semelhantes àquelas encontradas em situações reais de navegação. Para se obter resultados *on-line* o processamento do SIFT foi realizado paralelamente ao processo de estimação dos estados do robô, e seus resultados são incorporados sempre que possível para eliminar os erros acumulados. Essa estrutura permite a incorporação imediata de novas informações obtidas a partir de outros sensores, aumentando a quantidade de dados que o robô possui para resolver o problema do SLAM sem

comprometer a sua capacidade de navegação. Outra possibilidade é a fusão sensorial entre diferentes sensores, criando vínculos entre suas informações de maneira proporcionar uma melhor caracterização do ambiente.

Os testes finais com os robôs apresentados mostram que a utilização de sensores de visão omnidirecional é uma solução atraente para a obtenção da informação necessária para se resolver o problema do SLAM em diferentes situações por períodos de navegação arbitrariamente grandes. A premissa básica de que não há nenhum conhecimento prévio do ambiente foi mantida, e o algoritmo proposto nesse trabalho pode, em princípio, ser utilizado em qualquer tipo de ambiente. As restrições utilizadas no processo de determinação de marcos a partir de características podem ser livremente alteradas, tanto em relação aos seus parâmetros quanto à sua natureza. Diferentes tipos de câmera, geometria ou ambiente podem possuir determinados padrões que serão explorados de maneira mais eficiente com o auxílio de parâmetros específicos, aumentando a qualidade dos resultados finais.

7. Referências Bibliográficas

Akihiko, T.; Imiya, A. **A Panoramic Image Transform of Omnidirectional Images Using Discrete Geometry Techniques.** *Second International Symposium on 3D Data Processing, Visualization and Transmission (DPVT).* 2004.

Altermatt, M.; Martinelli, A.; Tomatis, N.; Siegwart, R. **SLAM With Corner Features Based on a Relative Map.** *Proceedings of IEEE, International Conference on Intelligent Robots and Systems.* Vol: 2. Pags: 1053-1058. 2004.

Andreasson, H.; Duckett, T. **Topological Localization for Mobile Robots Using Omnidirectional Vision and Local Features.** *Proceedings of 5th Symposium on Intelligent Autonomous Vehicles.* Julho 2004.

Araneda, A. **Statistical Inference in Mapping and Localization for Mobile Robots.** *PhD Thesis. Dept. Statistics, Carnegie Mellon University.* 2004.

Arya, S.; Mount, D. **Approximate Nearest Neighbour Queries in Fixed Dimensions.** *Fourth Annual Symposium on Discrete Algorithms (SODA).* Pags: 271-280. 1993.

Asada, H.; Brady, M. **The Curvature Primal Sketch.** *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol: 8. Num: 2. 1986.

Asmar, D.; Zelek, J.; Abdallah, S. **Tree Trunks as Landmarks for Outdoor Vision SLAM.** *Conference on Computer Vision and Pattern Recognition Workshop.* Vol: 17. Num: 22. Pags: 196-206. Junho 2006.

Avots, D.; Thibaux, R.; Thrun, S. **A Probabilistic Technique for Simultaneous Localization and Door State Estimation with Mobile Robots in Dynamic Environments.** *Proceedings of the Conference on Intelligent Robots and Systems (IROS).* Lausanne, Suiça, 2002.

Bailey, T.; Nieto, J.; Nebot, E. **Consistency of the FastSLAM Algorithm.** *Proceedings of IEEE, International Conference on Robotics and Automation (ICRA).* Pags: 424-429. Maio 2006.

Bailey, T. **Mobile Robot Localization and Mapping in Extensive Outdoor Environments.** *PhD Thesis. University of Sydney.* Sydney, Austrália, 2002.

Baker, S.; Nayar, S. **A Theory of Catadioptric Image Formation.** *International Conference for Computer Vision.* Pags: 35-42. 1998.

Balch, T.; Arkin, R. **Avoiding the Past: A Simple but Effective Strategy for Reactive Navigation.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Atlanta, Maio 1993.

Ballard, D. **Generalizing the Hough Transform to Detect Arbitrary Shapes.** *Pattern Recognition.* Vol: 13. Num: 2. Pags: 111-122. 1981.

Beis, J.; Lowe, D. **Shape Indexing Using Approximate Nearest-Neighbour Search in High Dimensional Spaces.** *Conference on Computer Vision and Pattern Recognition.* Pags: 1000-1006. Porto Rico, 1997.

Bekris, K.; Glick, M.; Kavraki, L. **Evaluation of Algorithms for Bearing-Only SLAM.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Pags: 1937-1943. Maio 2006.

Betke, M.; Gurvits, L. **Mobile Robot Localization Using Landmarks.** *Proceedings of IEEE, Transactions on Robotics and Automation.* Vol: 13. Num: 2. Abril 1997.

Bigun, J.; Buf, J. **Texture Segmentation by Real and Complex Momentos of the Gabor Power Spectrum.** *Progress in Image Analysis and Processing II.* Pags: 191-198. 1992.

Brady, M.; Cameron, S.; Durrant-Whyte, H.; Fleck, M.; Forsyth, D.; Noble, A.; Page, I. **Progress Towards a System that Can Acquire Pallets and Clean Warehouses.** *Fourth International Symposium of Robotics Research.* Pags: 359-374. 1987.

Brezetz, S.; Hebert, P.; Chatila, R.; Devy, M. **Uncertain Map Making in Natural Environments.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Minneapolis, Minnesota, Abril 1996.

Brooks, R. **Symbolic Error Analysis and Robot Planning.** *International Journal of Robotics Research.* Vol: 1. Num: 4. Pags: 29-68. 1982.

Brown, M.; Lowe, D. **Invariant Features from Interest Point Groups.** *British Machine Vision Conference.* Pags: 656-665. Cardiff, País de Gales, 2002.

Burgard, W.; Fox, D.; Thrun, S. **Active Mobile Robot Localization.** *Technical Report D-53117. Dept. of Computer Science, University of Bonn.* 1997.

Buschka, P. **An Investigation on Hybrid Maps for Mobile Robots.** *PhD Thesis. Institutionen för Teknik, Örebro University.* Örebro, Suécia, 2006.

Castellanos, J.; Neira, .; Tardós, J. **Limits to the Consistency of EKF-Based SLAM.** *5th Symposium on Intelligent Autonomous Vehicles (IRAC/EURON). Instituto Superior Técnico.* Lisboa, Portugal, Julho 2004.

Chatila, R.; Laumond, J. **Position Referencing and Consistent World Modelling for Mobile Robots.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Pags: 138-145. St. Louis, 1985.

Chong, K.; Kleeman, L. **Feature Based Mapping in Real, Large Scale Environments Using an Ultrasonic Array.** *International Journal on Robotics Research.* Vol: 18. Num: 1. 1999.

Choset, H.; Nagatani, K. **Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization.** *Proceedings of IEEE, Transactions on Robotics and Automation.* Vol: 17. Num: 2. Abril 2001.

Csorba, M. **Simultaneous Localization and Map Building.** *PhD Thesis. University of Oxford, Robotics Research Group.* 1997.

Davis, M.; Vinter, R. **Stochastic Modelling and Control.** *Monographs on Statistics and Applied Probability.* Maio 1985.

Davison, A.; Murray, D. **Mobile Robot Localization Using Active Vision.** *Proceedings of the 5th European Conference on Computer Vision.* Pags: 809-825. Freiburg, Alemanha, 1998.

Davison, A. **Real-Time Simultaneous Localization and Mapping with a Single Camera.** *International Conference on Computer Vision.* Nice, França, Outubro 2003.

Dellaert, F.; Fox, D.; Burgard, W.; Thrun, S. **Monte Carlo Localization for Mobile Robots.** *Internation Conference on Robotics and Automation (ICRA).* 1999.

Dissanayake, G.; Durrant-Whyte, H.; Bailey, T. **A Computationally Efficient Solution to the Simultaneous Localization and Map Building (SLAM) Problem.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Vol: 2. Pags: 1009-1014. 2000.

Dissanayake, G.; Newman, P.; Clark, S.; Durrant-Whyte, H.; Csorba, M. **A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.** *IEEE Transactions on Robotics and Automation.* Vol: 17. Num: 3. Junho 2001.

Duckett, T.; Marsland, S.; Shapiro, J. **Fast, Online Learning of Globally ConsistentAps.** *Autonomous Robots.* Vol: 12. Num: 3. Pags: 287-300. 2002.

Duda, R.; Hart, P. **Use of the Hough Transformation to Detect Lines and Curves in Pictures.** *Communications of the Association for Computer Machinery.* Vol: 15. Num: 1. Pags: 11-15. 1972.

Durrant-Whyte, H. **Uncertain Geometry in Robotics.** *IEEE Transactions on Robotics and Automation.* Vol: 4. Pags: 23-31. 1988.

Elfes, A. **Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.** *PhD Thesis. Department of Electrical and Computer Engineering, Carnegie Mellon University.* 1989.

Elfes, A. **Sonar-Based Real-World Mapping and Navigation.** *Proceedings of IEEE, International Journal of Robotics and Automation.* Vol: 3. Num: 3. Junho 1987.

Eliazar, A.; Parr, R. **DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks.** *Proceedings of the 16th International Joint Conferenc on Artificial Intelligence (IJCAI).* Acapulco, México, 2003.

Engelson, S.; McDermott, V. **Error Correction in Mobile Robot Map Learning.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Nice, França, Maio 1992.

Ersson, T.; Hu, X. **Path Planning and Navigation of Mobile Robots in Unknown Environments.** *Proceedings of IEEE, International Robots and Systems (IROS).* 2001.

Fitzgibbons, T. **Visual-Based Simultaneous Localization and Mapping.** *PhD Thesis. University of Sydey.* Austrália, Outubro 2004.

Fox, D.; Burgard, W.; Thrun, S. **Markov Localization for Mobile Robots in Dynamic Environments.** *Journal of Artificial Intelligence Research.* Pags: 391-427. 1999.

Friedman, J.; Bentley, J.; Finkel, R. **An Algorithm for Finding Best Matches in Logarithmic Expected Time.** *ACM Transactions on Mathematical Software (TOMS).* Vol: 3. Num: 3. Pags: 209-226. Setembro 1977.

Gaspar, J. **Omnidirectional Vision for Mobile Robot Navigation.** *Tese de Doutorado. Universidade Técnica de Lisboa, Instituto Superior Técnico.* 2002.

Gluckman, J.; Nayar, S.; Thorek, K. **Real-Time Omnidirectional and Panoramic Stereo.** *Proceedings of DARPA Image Understanding Workshop.* Vol: 1. Pags: 299-303. 1998.

Goedemé, T.; Nuttin, M.; Tuytelaars, T.; Gool, L. **Omnidirectional Vision Based Topological Navigation.** *International Journal of Computer Vision. Springer Netherlands.* Vol: 74. Num: 3. Pags: 219-236. Setembro 2007.

Grassi Jr., J.; Okamoto Jr., J. **Development of an Omnidirectional Vision System.** *Journal of the Brazilin Society of Mechanical Sciences and Engineering.* Vol: 28. Num: 1. Pags: 58-68. Rio de Janeiro, 2006.

Guivant, J.; Nebot, E. **Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementations.** *Proceedings of IEEE, Transactions of Robotics and Automation.* Vol: 17. Num: 3. Junho 2001.

Guizilini, V.; Okamoto Jr., J.; Corrêa, F.; Grassi Jr., V. **Implementação do DP-SLAM em Tempo Real para Robôs Móveis Usando Sensores Esparsos.** *8º Simpósio Brasileiro de Automação Inteligente (SBAI).* Florianópolis SC, Outubro 2007.

Guizilini, V.; Okamoto Jr., J. **Visão Estéreo Omnidirecional com Espelho Hiperbólico Duplo.** *13º Simpósio Internacional de Iniciação Cientifca de São Paulo (SIICUSP).* São Paulo, 2005.

Gutmann, J.; Konolige, K. **Incremental Maping of Large Cyclic Environments.** *Proceedings of IEEE, International Symposium on Computational Intelligence in Robotics and Automation.* Pags: 318-315. Califórnia, Novembro 1999.

Haehnel, D.; Burgard, W.; Fox, D.; Thrun, S. **An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements.** *International conference on Intelligent Robots and Systems.* 2003.

Harris, C.; Stephens, M. **A Combined Corner and Edge Detector.** *4th Alvey Vision Conference.* Pags: 147-151. 1988.

Horswill, I. **Polly: A Vision-Based Artificial Agent.** *Proceedings of National Conference on Artificial Intelligence.* Pags: 824-829. Washington DC, 1993.

Hu, H.; Gu, D. **Landmark-Based Navigation of Industrial Mobile Robots.** *International Journal of Industry Robot.* Vol: 27. Num: 6. Pags: 458-467. 2000.

Ikeda, S.; Miura, J. **3-D Indoor Environment Modeling by a Mobile Robot With Omnidirectional Stereo and Laser Range Finder.** *Proceedings of International Conferene on Intelligent Robots and Systems.* Pags: 3435-3440. 2006.

Jensfelt, P. **Approaches to Mobile Robot Localization in Indoor Environments.** *PhD Thesis in Signal, Sensors and Systems. Royal Institute of Technology SE-10044.* Estocolmo, Suécia, 2001.

Jensfelt, P.; Kristensen, S. **Active Global Localization for Mobile Robot Using Multiple Hypothesis Tracking.** *Proceedings of IJCAI, Workshop on Reasoning With Uncertainty in Robot Navigation.* Pags: 13-22. Estocolmo, Suécia, 1999.

Jung, I. **Simultaneous Localization and Mapping in 3-D Environments With Stereo Vision.** *PhD Thesis.* CNRS, Toulouse, 2004.

Kadir, T.; Brady, M.; Zisserman, A. **An Affine Invariant Method for Selecting Salient Regions in Images.** *Proceedings of the 8th European Conference on Computer Vision.* Pags: 345-357. Praga, 2004.

Kaess, M.; Dellaert, F. **A Markov Chain Monte Carlo Approach to Closing the Loop in SLAM.** *Proceedings of IEEE, International Conference on Robotics and Automation (ICRA).* Pags: 643-648. 2005.

Kalman, R. **A New Approach to Linear Filtering and Prediction Problems.** *Transactions of ASME (Journal of Basic Engineering).* Vol: 82. Pags: 35-45. 1960.

Kehagias, A.; Diugash, J.; Singh, S. **Range-Only SLAM With Interpolated Range Data.** *Technica Report CMU-RI-TR-06-26. Robotics Institute, Carnegie Mellon University.* Maio 2006.

Kim, J.; Sukkarieh, S. **Airborne Simultaneous Localization and Map Building.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Taipei, Taiwan, 2003.

Kim, J.; Chung, M. **SLAM With Omnidirectional Stereo Vision Sensor.** *Proceedings of IEEE, International Conference on Intelligent Robots and Systems.* Las Vegas, Nevada, Outubro 2003.

Kleeman, L. **Optimal estimation of Position and Heading for Mobile Robots Using Ultrasonic Beacons and Dead Reckoning.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Pags: 2582-2587. 1992.

Koenderink, J.; Doorn, A. **Representation of Local Geometry in the Visual System.** *Biological Cybernetics.* Vol: 55. Pags: 367-375. 1987.

Kortenkamp, D.; Bonasso, R.; Murphy, R. **AI-Based Mobile Robots: Case Studies of Successful Robot Systems.** *MIT Press.* 1998.

Kwok, N.; Dissanayake, G. **An Efficient Multiple Hypothesis Filter for Bearing-Only SLAM.** *Proceedings of IEEE, International Conference on Intelligent Robots and Systems.* Sendai, Japão, 2004.

Lazebnik, S.; Schmid, C.; Ponce, J. **Sparse Texture Representation Using Affine-Invariant Neighbourhoods.** *Proceedings of the Conference on Computer Vision and Pattern Recognition.* Pags: 319-324. Madison, Wisconsin, USA, 2003.

Ledwich, L.; Williams, S. **Reduced SIFT Features for Image Retrieval and Indoor Localization.** *Australian Conference on Robotics and Automation.* 2004.

Leonard, J.; Feder, H. **Decoupled Stochastic Mapping.** *Technical Report. Massachusetts Institute of Technology, Marine Robotics Laboratory.* 1999.

Leonard, J.; Durrant-Whyte, H. **Directed Sonar Sensing for Mobile Robot Navigation.** *Kluwer Academic.* Boston, MA, 1992.

Leonard, J.; Durrant-Whyte, H.; Cox, I. **Dynamic Map Building for an Autonomous Mobile Robot.** *International Journal of Robotics Research.* Vol: 11. Num: 4. Pags: 89-96. 1992.

Leonard, J.; Rickoski, R.; Newman, P.; Bosse, M. **Mapping Partially Observable Features from Multiple Vantage Points.** *International Journal of Robotics Research.* Vol: 21. Num: 10-11. Pags: 943-975. 2002.

Leonard, J.; Durrant-Whyte, H. **Mobile Robot Localization by Tracking Geometric Beacons.** *Proceedings of IEEE, Transactions on Robotics and Automation.* Vol: 7. Num: 3. Pags: 376-382. 1991.

Lindeberg, T. **Detecting Salient Blob-Like Image Structures and Their Scale With a Scale-Space Primal Sketch: A Method for Focus-of-Attention.** *International Journal of Computer Vision.* Vol: 11. Num: 3. Pags: 283-318. 1993.

Lindeberg, T. **Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales.** *Journal of Applied Statistics.* Vol: 21. Num: 2. Pags: 224-270. 1994.

Liu, J.; Chen, R.; Logvinenko, T. **A Theoretical Framework for Sequential Importance Sampling and Resampling.** *Sequential Monte Carlo in Practice. Springer-Verlag.* Janeiro 2001.

Liu, J.; Chen, R. **Sequential Monte Carlo Methods for Dynamic Systems.** *Journalof the American Statistic Association.* Num: 93. 1998.

Lowe, D. **Distinctive Image Features from Scale-Invariant Keypoints.** *International Journal of Computer Vision.* Vol: 60. Num: 2. Pags: 91-110. 2004.

Lowe, D. **Object Recognition From Local Scale Invariant Features.** *Proceedings of the International Conference on Computer Vision (ICCV).* Pags: 1150-1157. 1999.

Lu, F.; Milios, E. **Globally Consistent Range Scan Alignmnent for Environment Mapping.** *Autonomous Robots.* Vol: 4. Pags: 333-349. 1997.

Maybeck, P. **Stochastic Models, Estimation and Control.** *New York Academic.* Vol: 1. 1979.

Mikolajczyk, K.; Schmid, C. **A Performance Evaluation of Local Descriptors.** *Proceedings of IEEE, Transactions on Pattern Analysis and Machine Intelligenc.* Vol: 27. Num: 10. Pags: 1615-1630. 2002.

Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. **FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem.** *Proceedings of the Association for the Advancement of Artificial Intelligence.* 2002.

Montemerlo, M. **FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association.** *Phd Thesis. Robotics Institute, Carnegie Mellon University.* Pittsburgh PA, 2003.

Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. **FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Probably Converges.** *Proceedings of the 16th International Conference on Artificial Intelligence (IJCAI).* Acapulco, México, 2003.

Moravec, H. **Visual Mapping by a Robot Rover.** *International Joint Conference on Artificial Intelligence.* Pags: 598-600. 1979.

Moutarlier, P.; Chatila, R. **Stochastic Multisensory Data Fusion for Mobile Robot Localization and Environment Modeling.** *5th Simposium on Robotics Research.* Tóquio, 1989.

Murphy, K. **Bayesian Map Learning in Dynamic Environments.** *Neural Information Processing Systems (NIPS).* 1999.

Murray, D.; Little, J. **Using Real-Time Stereo Vision for Mobile Robot Navigation.** *Autonomous Robots.* Vol: 8. Num: 2. Pags: 161-171. 8 2000.

Nalwa, V. **A True Omnidirectional Viewer.** *Technical Report, Bell Lab.* Holmdel, 1996.

Nebot, E.; Masson, F.; Guivant, J.; Durrant-Whyte, H. **Robust Simultaneous Localization and Mapping for Very Large Outdoor Environments.** *Proceedings of the 8th International Symposium on Experimental Robotics (ISER).* 2002.

Negenborn, R. **Robotic Localization and Kalman Filters.** *PhD Thesis. Utrecht University.* 2003.

Nieto, J.; Guivant, J.; Nebot, E.; Thrun, S. **Real Time Data Association for FastSLAM.** *Proceedings of IEEE, International Conference on Robotics and Automation (ICRA).* 2003.

Nieto, J.; Guivant, J.; Nebot, E.; Thrun, S. **A New Solution to the Simultaneous Localization and Map Building (SLAM) Problem - The GPF.** *Department of Mechanical and Mechatronic Engineering. University of Sydney.* Austrália, 2006.

Olson, O. **Selecting Landmarks for Localization in Natural Terrain.** *Autonomous Robos.* Vol: 12. Pags: 201-210. 2002.

Panzieri, S.; Pascucci, F.; Ulivi, G. **Vision Based Navigation Using Kalman Approach for SLAM.** *Technical Report. Universit'a degli Studi "La Sapienza".* 2001.

Peleg, S.; Ben-Erza, M. **Stereo Panorama With a Single Camera.** *Proceedings of Computer Vision and Pattern Recognition.* Pags: 395-401. 1999.

Pierce, D.; Kuipers, B. **Learning to Explore and Build Maps.** *Proceedings of the 12th National Conference on Artificial Intelligence. MIT Press, Menlo Park.* Pags: 1264-1271. Julho 1994.

Prasser, D.; Wyeth, G. **Probabilistic Visual Recognition of Artificial Landmarks for Simultaneous Localization and Mapping.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Vol: 1. Num: 14. Pags: 1291-1296. Setembro 2003.

Press, P.; Austin, D. **Approaches to Pole Detection Using Ranged Laser Data.** *Proceedings of Australasian Conference on Robotics and Automation.* 2004.

Rekleitis, I. **A Particle Filter Tutorial for Mobile Robot Localization.** *International Conference on Robotics and Automation (ICRA).* 2003.

Se, S.; Lowe, D.; Little, J. **Simultaneous Localization and Map-Building Using Active Vision.** *The International Journal of Robotics Research.* Vol: 21. Num: 8. Pags: 735-758. Agosto 2002.

Se, S.; Lowe, D.; Little, J. **Vision-Based Mobile Robot Localization and Mapping Using Scale-Invariant Features.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Pags: 2051-2058. Coréia, 2001.

Shi, J.; Tomasi, C. **Good Features to Track.** *Proceedings of IEEE, Conference on Computer Vision and Pattern Recognition.* Pags: 593-600. 1994.

Simon, J.; Uhlmann, J. **A New Extension of the Kalman Filter to Nonlinear Systems.** *Proceedings of Aerosense, 11th International Symposium on Aerospace/Defense Sensing, Simulations and Controls, Multisensor Fusion, Tracking and Resource Management (SPIE).* 1997.

Smith, R.; Self, M.; Cheeseman, P. **Estimating Uncertain Spatial Relationships in Robotics.** *Autonomous Robot Vehicles. Springer.* Pags: 167-193. 1990.

Smith, R.; Cheeseman, P. **On the Representation and Estimation of Spatial Uncertainty.** *International Journal of Robotics Research.* Vol: 5. Num: 4. Pags: 56-68. 1986.

Smith, S.; Self, M.; Cheeseman, P. **A Stochastic Map for Uncertain Spatial Relationships.** *Fourth International Symposium of Robotics Research.* Pags: 467-474. 1987.

Stentz, A. **Optimal and Efficient Path Planning for Partially-Known Environments.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Vol: 4. Pags: 3310-3317. San Diego CA, USA, Maio 1994.

Sujan, V.; Meggiolaro, M.; Belo, F. **Mobile Robot Localization and Mapping Using Low Cost Vision Sensors.** *Springer Tracts in Advanced Robotics.* 2005.

Svoboda, T.; Pajdla, T. **Epipolar Geometry for Central Catadioptric Cameras.** *International Journal of Computer Vision.* Vol: 49. Num: 1. 2002.

Tardós, J.; Neira, J.; Newman, P.; Leonard, J. **Robust Mapping and Localization in Indoor Environments Using Sonar Data.** *Technical Report.* Num: 4. 2001.

Taylor, R. **A Synthesis of Manipulator Control Programs from Task-Level Specifications.** *Stanford Artificial Intelligence Laboratory Memo.* Num: 282. 1976.

Thorpe, C.; Durrant-Whyte, H. **Field Robots.** *Proceedings of the 10th International Symposium of Robotics Research (ISRR).* Lorne, Austrália, 2001.

Thrun, S. **A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots.** *International Journal of Robotics Research.* Vol: 20. Pags: 335-363. 2001.

Thrun, S.; Montemerlo, M.; Koller, D.; Wegbreit, B.; Nieto, J.; Nebot, E. **FastSLAM: an Efficient Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association.** *Journal of Machine Learning Research.* 2004.

Thrun, S. **Particle Filter in Robotics.** *Proceedings of the 17th Annual Conference on Uncertainty and Artificial Intelligenc (UAI).* 2002b.

Thrun, S.; Fox, D.; Burgard, W. **Probabilistic Robotics.** *MIT Press.* Cambridge MA, 2005.

Thrun, S. **Robotic Mapping: A Survey.** *Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann.* 2002a.

Thrun, S.; Montemerlo, M. **The GraphSLAM Algorithm With Application to Large-Scale Mapping of Urban Structures.** *International Journal of Robotics Research.* Vol: 25. Num: 5-6. Pags: 403-430. 2006.

Trucco, E.; Verri, A. **Introductory Techniques for 3-D Computer Vision.** *Prentice Hall Inc.* 1998.

Tuytelaars, T.; Mikolajczyk, K. **A Survey on Local Invariant Features.** *K.U. Leuven & University of Surrey.* Maio 2006.

Wahlren, C.; Duckett, T. **Topological Map Building for Mobile Robots Using Omnidirectional Vision.** *Proceedings of SWAR, 3th Swedish Workshop on Autonomous Robotics.* Pags: 28-39. Estocolmo, Suécia, 2005.

Welch, G.; Bishop, G. **An Introduction to the Kalman Filter.** *Technical Report, TR 95-041. University of North Carolina, Department of Computer Science.* 1995.

Williams, S. **Efficient Solutions to Autonomous Mapping and Navigation Problems.** *PhD Thesis. Australian Centre for Field Robotics.* 2001.

Witkin, A. **Scale-Space Filtering.** *International Joint Conference on Artificial Intelligence.* Pags: 1019-1022. Karlsruhe, Alemanha, 1983.

Yamauchi, B.; Schultz, A.; Adams, W. **Mobile Robot Exploration and Map-Building with Continuous Exploration.** *Proceedings of IEEE, International Conference on Robotics and Automation.* Leuven, Bélgica, Maio 1998.

Zelinsky, A. **A Mobile Robot Exploration Algorithm.** *Proceedings of IEEE, Transactions on Robotics and Automation.* Vol: 8. Num: 6. Dezembro 1992.

Zhu, Z. **Omnidirectional Stereo Vision.** *Workshop on Omnidirectional Vision. Proceedings of IEEE, 10th International Conference on Advanced Robotics.* Budapeste, Hungria, 2001.