3
Implementação dos Algoritmos no Sistema Experimental

3.1
SLAM

3.1.1
Filtragem das Varreduras

Um problema crucial na *Correspondência de Varreduras* é como selecionar pontos das Varreduras que são úteis para a correspondência. Por que uma Varredura inclui centenas de pontos, muitos deles podem ser redundantes e afetar a eficiência e precisão dos algoritmos, em especial quando a forma do meio ambiente é simples. Além disso, o valor da Equação (2-26) é influenciado pela densidade das leituras. Regiões com maior densidade de leitura são produzidos quando o robô móvel está perto de uma parede, assim a Equação (2-26) resulta em valores mais elevados nestas regiões, veja a Figura 3.1. O algoritmo de Correspondência de Varreduras pode dar resultados inadequados de sobreposição, como na Figura 3.2. Para superar esta situação, este trabalho utiliza dois métodos, uma reamostragem uniforme utilizada por Burguera [31] e Ynoquio [2], e um reamostragem destacando as características mais relevantes (*Reamostragem baseada em Saliências* [35].

3.1.1.1
Reamostragem Uniforme

Este filtro é capaz de criar uma nova amostra da Varredura com uma perda mínima de informação e custo computacional baixo. A ideia por trás deste filtro é mover uma janela circular sobre a Varredura e substituir os pontos dentro da janela com seu centro de gravidade. A Figura 3.3 exemplifica o efeito do passo de reamostragem. O Filtro tem o efeito de alisamento da distribuição de pontos ao longo da Varredura e também reduz o número de pontos da Varredura, sem perder muita informação.

O raio da janela define a distância mínima entre os pontos na Varredura filtrada, ver Figura3.3 (b). Este raio tem de ser definido experimentalmente. Os

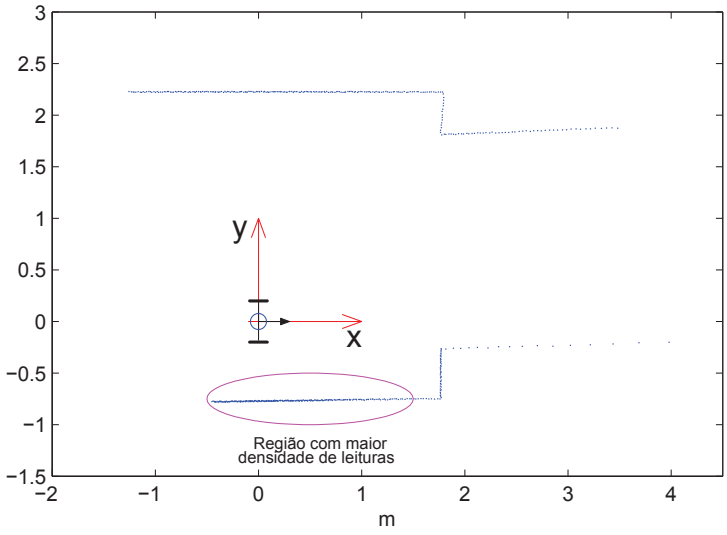


Figura 3.1: Regiões de alta densidade, produzida por um robô móvel situado perto da parede

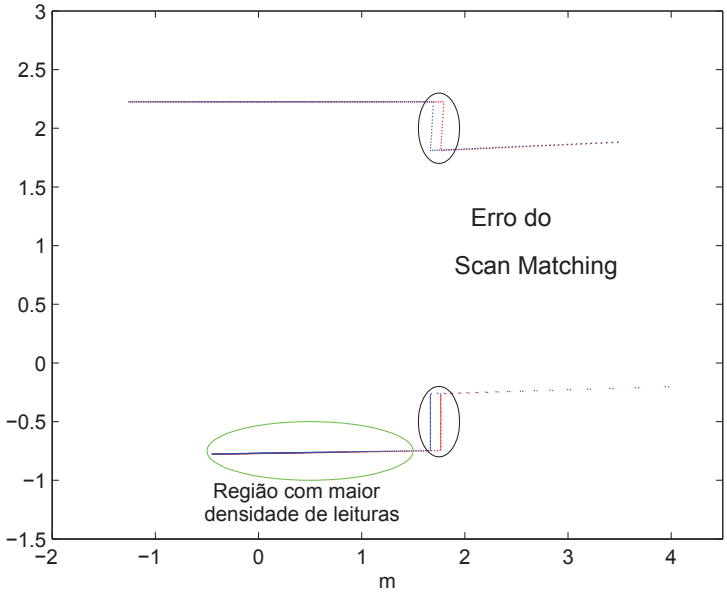
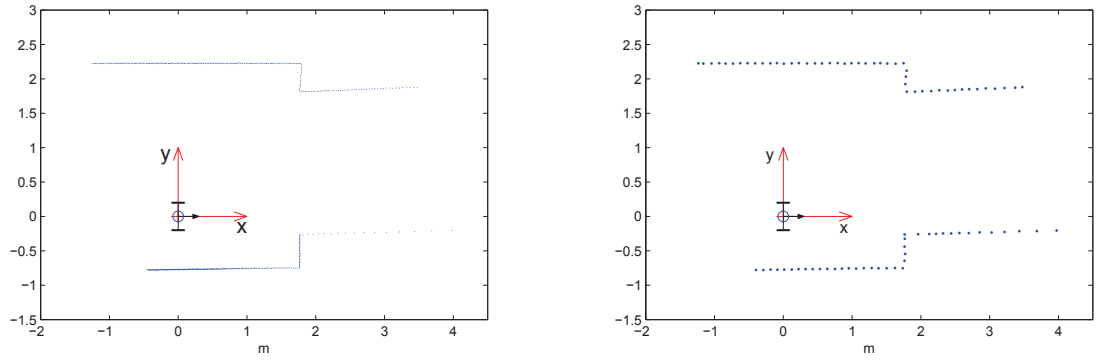


Figura 3.2: Erro na sobreposição das Varreduras (círculos pretos), a primeira Varredura (pontos vermelhos) e a segunda Varredura (pontos azuis) apresentam regiões de alta densidade (círculo verde) de pontos

valores baixos deste parâmetro não resolvem a influência da densidade das leituras, enquanto um valor elevado pode causar perda de informações da Varredura. Burguera[31] experimentalmente testou o parâmetro a um valor de $5cm$ para o sensor utilizado. Mas, neste trabalho, este parâmetro é definido a $10cm$, devido ao tipo de sensor adotado, menos preciso.



(a) (b)

Figura 3.3: Reamostragem Uniforme da Varredura. (a) Varredura original com 594 pontos (b) Varredura depois da reamostragem com 85 pontos.

3.1.1.2 Reamostragem Baseada em Saliências

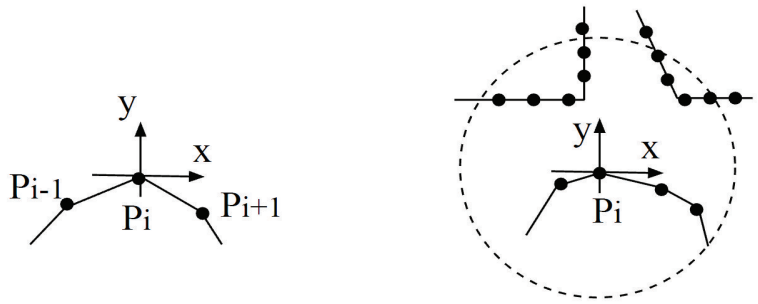
O método calcula a *Saliência* de cada ponto da Varredura de acordo com a quantidade de informação que tem o ponto, ou o comprimento do segmento de linha em que se encontra o ponto. Com base na *Saliência*, o método determina se cada ponto da Varredura deve ser reamostrado ou não. Pontos do Varredura redundantes são descartados e pontos que apresentam características relevantes são preservados, desse modo a eficiência e precisão do algoritmo de Correspondência de Varreduras é melhorada.

Reamostragem baseada em saliências não usa os recursos explícitos como linha e canto. Em vez disso, utiliza um ponto da Varredura com a sua direção tangente, que é referido como *Ponto Direcionado*, como é mostrado na Figura3.4(a). O sistema de coordenada local de um *Ponto Direcionado* é tal que a origem está localizada no ponto da Varredura e o eixo x é a tangente do ponto indicado [30]. Também define-se a *Assinatura* de um *Ponto Direcionado* P_i como um conjunto P_j de outros *Pontos Direcionados* em torno P_i. Transformando a coordenada de cada P_j para o sistema de coordenada local de P_i, é possível fazer a *Assinatura* invariante à translação e rotação de P_i.

Para comprimir a *Assinatura*, aplica-se a transformação de Hough para cada P_j da *Assinatura* e projeta-se no espaço de Hough (ρ, θ):

$$\rho(\theta) = x_o cos(\theta) + y_o sin(\theta) \tag{3-1}$$

Isto gera um padrão no espaço de Hough. Em seguida, uma partição do espaço



(a) (b)

Figura 3.4: Representação da *Saliência*

de Hough em quadrículas é obtida para gerar uma tabela, Se selecionam os pontos Hough que são máximos locais em termos do número de ocorrências. O tamanho da quadrícula aqui adotada é $5[cm]$ x $5[graus]$ [30]. Assim a *Assinatura* de P_i é definida como:

$$G_i = \{(q_{ij}, w_{ij})/1 \leq j \leq N_i\} \tag{3-2}$$

onde q_{ij} é um ponto Hough (ρ_{ij}, θ_{ij}), w_{ij} é o número de votos de q_{ij} e N_i é o número de valores máximos locais.

O método define a $Salincia\ L_i$ de um *Ponto Direcionado* P_i com base na quantidade de informação da sua *Assinatura* G_i.

$$L_i = -\sum_{j}^{N_i} log(P(q_{ij})) \tag{3-3}$$

onde $-log(P(q_{ij}))$ é a quantidade de informação q_{ij}, e $P(q_{ij})$ é a probabilidade que q_{ij} aparece em uma *Assinatura*.

A Equação (3-3) fornece uma boa medida para a *Saliência*, mas consome tempo de computação, porque as *Assinaturas* de todos os *Pontos Direcionados* devem ser calculadas antes da reamostragem deles. Para evitar isto, a transformação de Hough é aplicada a cada P_i da Varredura, está transformação cria uma tabela, através de uma aproximação para a *Saliência* definida como:

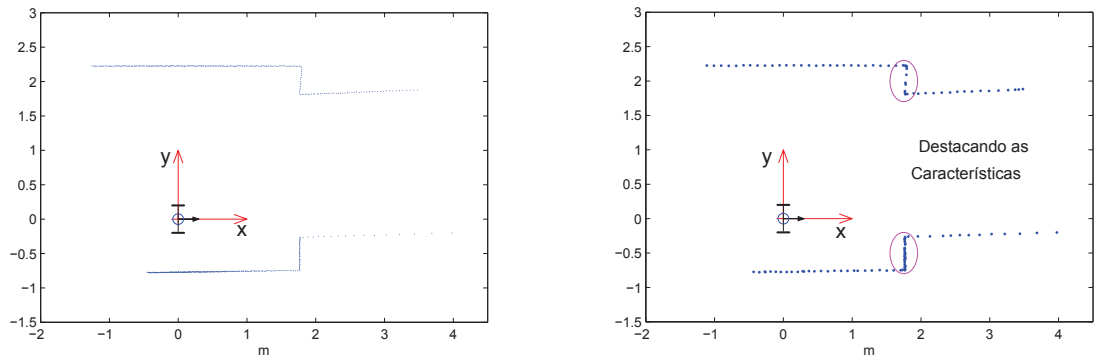
$$L_i = \{\frac{u_i}{N}\}^{-1} \tag{3-4}$$

onde u_i denota o número de votos da quadrícula da tabela de Hough que corresponde a um *Ponto Direcionado* P_i, e N é o número total de pontos P_i da Varredura.

A reamostragem baseada em Saliências consiste em:

- Reamostrar os pontos da Varredura uniformemente com um intervalo de $10cm$ (Reamostragem Uniforme).
- Reamostrar um ponto da Varredura que tem uma *Saliência* L_i elevada.

A Figura 3.5 mostra os resultados para uma Varredura adquirida em um corredor. O número de pontos da Varredura é 594. A reamostragem reduz o número de pontos para 134. Como pode ser observado, todos os pontos em torno das quinas são preservados, e os pontos nas paredes são redefinidos quase uniformemente.



(a) (b)

Figura 3.5: Reamostragem baseada em saliências. (a) Varredura original com 594 pontos (b) Varredura depois do reamostragem com 134 pontos.

3.2 Algoritmo de Correspondência de Varreduras

O algoritmo de Correspondência de Varreduras é importante para realizar um mapeamento e localização do robô móvel utilizando um LRF sem sensores de odometria. Como a informação fornecida pelo LRF é composta de Varreduras em diferentes posições do robô móvel dentro do ambiente, com esta informação podemos substituir o sensor de odometria.

Assim, para substituir um sensor de odometria, usam-se duas Varreduras consecutivas consecutivos para fazer uma sobreposição deles usando o algoritmo de Correspondência de Varreduras. O resultado desta sobreposição é o deslocamento entre duas Varreduras consecutivas. Se o LRF é montado no centro do robô móvel, o deslocamento é igual ao deslocamento do robô móvel.

O algoritmo de Correspondência de Varreduras utilizado neste trabalho foi descrito no capítulo 2. O algoritmo de Correspondência de Varreduras por NDT fornece uma boa função para representar uma sobreposição de duas leituras consecutivas do LRF pela Equação (2-26). O algoritmo NDT original utiliza como método de minimização o algoritmo de Newton [17]. Mas, neste trabalho, como método de otimização é usado evolução diferencial ED [2].

O algoritmo de evolução diferencial é utilizado no processo de otimização para tornar o algoritmo de Correspondência de Varreduras mais robusto. A optimização através de algoritmos genéticos foi detalhada no capítulo 2.

3.3 Parâmetros e Considerações

3.3.1 O Sensor

Talvez o mais conhecido LRF em robótica é o LRF da família SICK [2]. Mas o LRF utilizado neste trabalho é da família URG, um sensor com uma faixa de medição muito menor que os sensores da família do SICK.

O modelo utilizado é $URG-04LX-UG01$. A Tabela 3.1 mostra as suas características mais importantes. Uma característica importante a considerar

Tabela 3.1: Características do sensor LRF modelo $URG-04LX-UG01$.

Especificações	
Faixa de Detecção	$20mm$ a $4000mm$
Área de Varredura	240^o
Frequência de Varredura	$10Hz$ $(100msec/scan)$
Resolução	$1mm$
Precisão	$20mm-1000mm$: $\pm 30mm$ $20mm-4000mm$: $\pm 3\%$ da medição
Resolução Angular	0.36^o

é a velocidade máxima linear e angular do robô móvel, a fim de garantir que seu movimento não afeta a leitura do LRF. As Equações (3-5) e (3-6) calculam a velocidade máxima mediante uma função da frequência da varredura do sensor.

$$v_{max} = \xi_t \cdot f_s \tag{3-5}$$

$$w_{max} = \xi_r \cdot f_s \tag{3-6}$$

onde f_s é a Frequência da varredura, e ξ_r e ξ_t são o erro máximo introduzido pelo movimento do robô. Assim, por exemplo, para erros $\xi_r = 1^o$ e $\xi_t = 10mm$, as velocidades máximas do robô móvel precisam ser:

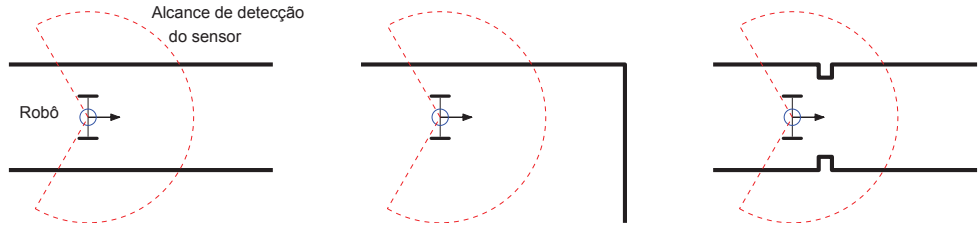
$$v_{max} = 0.1m/s$$

$$w_{max} = 0.18rad/s$$

3.3.2 O Ambiente

A faixa de medição do sensor utilizado é de até $4m$, implicando que o ambiente a ser mapeado precisa possuir variedade de características numa seção inferior a $4m$. Por exemplo, longos corredores com paredes paralelas, ou lugares espaçosos cujas paredes estão fora da faixa de detecção do LRF, produzem varreduras sucessivas que o algoritmo de Correspondência de Varreduras resolve erradamente, pois a correspondência consome muito tempo por causa de sua ambiguidade. Em particular, se o S_{ref} é constituído por uma ou duas linhas paralelas, infinitas possibilidades seriam obtidas. Essas varreduras são comuns em um ambiente real. A Figura 3.6 mostra exemplos de ambiguidade; nos casos (a) e (b) a exata correspondência é impossível na teoria.

Em alguns casos, varreduras tem partes salientes pequenas, como mostra a Figura 3.6 (c), fornecendo poucas informações para fazer a correspondência. Para lidar com este problema, é necessário utilizar filtros que destaquem pontos com características distintas e que descartem pontos redundantes.



(a) Paredes paralelas (b) Parede única (c) Paredes com pilares

Figura 3.6: Exemplos de degeneração.

3.3.3 Robô Móvel

Os experimentos utilizaram o robô *iRobot Create* [36] para os testes de SLAM. Esse modelo de robô (ver Figura 3.7) é didático e amplamente utilizado por estudantes e pesquisadores. O *iRobot Create* é um dispositivo que recebe comandos pela porta serial e os executa, ou também pode ser controlado remotamente. Ao robô é acoplado a LRF modelo $URG-04LX-UG01$ e um

Figura 3.7: O Robô *iRobot Create* acoplado a um LRF ($URG-04LX-UG01$)

microcomputador portátil da classe netbook, modelo Asus Eee PC 1000HEB, com processador Intel Aton N280 de $1,66Ghz$, responsável por enviar comandos de movimentos e capturar dados do LRF.

Aqui, listamos algumas características importantes que justificam o uso do *iRobot Create* nos testes:

- Possui amplo espaço para instalação de hardware adicional.
- Pode usar um cabo serial para enviar comandos a partir de um PC.
- Uma série de protocolos totalmente documentados, proporciona acesso total aos sensores, atuadores e funcionalidade do robô móvel.
- Autonomia de 90 a 120 minutos por carga, dependendo do piso. Carregamento da bateria independente.
- Movimento de translação e rotação independente.

- A velocidade que o robô pode atingir é entre -500 e 500 mm/s.

Uma das características do robô é ser capaz de receber comandos para realizar seu movimento a partir de um computador. Além disso, Microsoft Robotics Developer Studio tem bibliotecas especializadas para o *iRobot Create*. A assim, mediante a linguagem de programação VPL (Visual Program Lenguage), é possivel criar um painel de controle para seu movimento, ver Figura 3.8. O painel

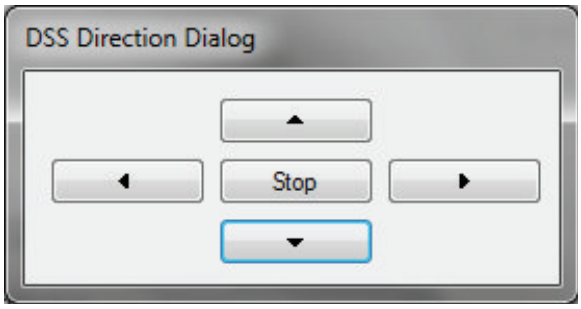


Figura 3.8: Controle Mediante Microsoft Robotics Developer Studio

de controle de movimento (Figura 3.8), envia o movimento a ser realizado pelo *iRobot Create*: seguir em frente, para trás, vire à esquerda ou à direita. A Figura 3.9 mostra o esquema de programação em VLP (Microsoft Robotics Developer Studio). Nos experimentos, os comandos são enviados por um teleoperador, ou

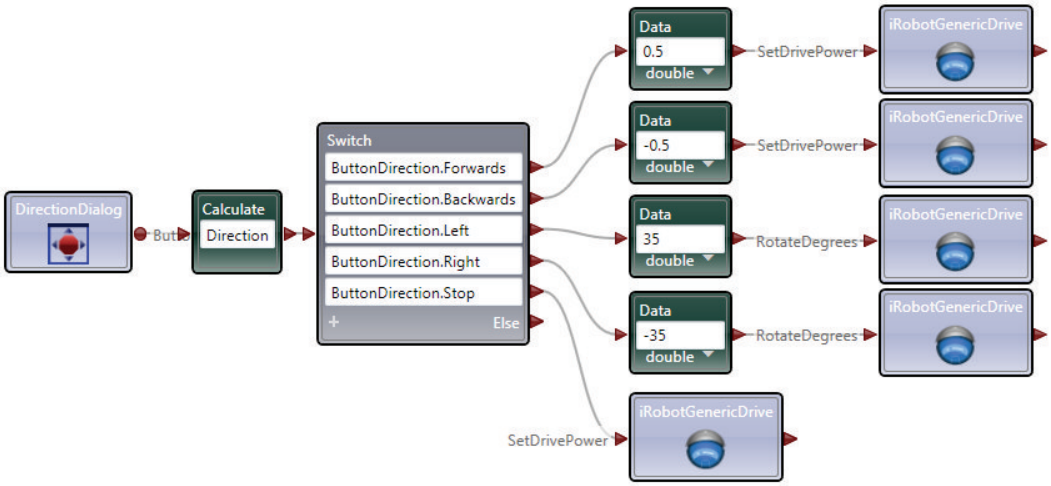


Figura 3.9: Esquema do Controle em Microsoft Robotics Developer Studio

pré-programados no microcomputador. Note, no entanto, que os algoritmos de SLAM implementados não fazem uso destes comandos, gerando os mapas e localizando o robô apenas a partir de varreduras sucessivas pelo LRF.

No próximo capítulo, algumas simulações são apresentadas.