UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

CARLOS MIGUEL PRESCENDO TONIN

DESENVOLVIMENTO DE UM SISTEMA DE NAVEGAÇÃO PARA SISTEMAS AGV ATRAVÉS DO MÉTODO SLAM

PORTO ALEGRE-RS
2018

CARLOS MIGUEL PRESCENDO TONIN

DESENVOLVIMENTO DE UM SISTEMA DE NAVEGAÇÃO PARA SISTEMAS AGV ATRAVÉS DO MÉTODO SLAM

Trabalho de Conclusão do Curso de Engenharia de Controle e Automação da Universidade Federal do Rio Grande do Sul, apresentado à Banca Julgadora como pré-requisito para aprovação na atividade.

Orientador: Prof. Dr. Heraldo José de Amorim

Co-orientador: Prof. Dr. Flávio José Lorini

PORTO ALEGRE - RS
2018

CARLOS MIGUEL PRESCENDO TONIN

DESENVOLVIMENTO DE UM SISTEMA DE NAVEGAÇÃO PARA SISTEMAS AGV ATRAVÉS DO MÉTODO SLAM

Trabalho de Conclusão do Curso de Engenharia de Controle e Automação da Universidade Federal do Rio Grande do Sul, apresentado à Banca Julgadora como pré-requisito para aprovação na atividade.

PORTO ALEGRE, 25 DE JUNHO DE 2018

Banca Examinadora

_____

Prof. Dr. Mario Roland Sobczyk Sobrinho

_____

Prof. Dr. Eduardo André Perondi

_____

Prof. Dr. Renato Ventura Bayan Henriques

_____

Prof. Dr. Heraldo José de Amorim

“Dedico este trabalho a minha vó que sempre me inspirou a dar meu melhor”

AGRADECIMENTOS

Gostaria de agradecer, sobretudo, aos meus pais Leonel e Geni por todo o apoio durante o curso, por toda a paciência e compreensão ao longo dos anos durante as maratonas de estudo que se convertiam em meses sem visitas e ligações não atendidas, não há formas de descrever o quão sou grato pela presença de vocês. Em especial agradeço à minha avó Albina, a qual sempre depositou uma fé incalculável em mim e foi a fonte de forças quando os problemas pareciam insuperáveis.

Ao meu orientador, Heraldo Amorim, por ter acreditado no trabalho e pelo suporte e parceria ao longo da realização do mesmo.

À empresa Spark e aos colegas que ali trabalham, pelo apoio e auxílio para o desenvolvimento deste trabalho sem os quais o mesmo não teria sido possível.

Aos meus colegas Bruno Exner, João Vitor Assmann, Rafael Pergher e Thiago Companhoni, por todo o apoio e ajuda desinteressada ao longo de toda a graduação, este trabalho certamente só existe graças a toda parceria de vocês, e por isso sempre serei grato. Agradeço também aos meu colegas Thomás Schulze e Giovana Bandinelli, por terem sido de fundamental importância para a manutenção da minha sanidade durante as partes mais desgastantes do curso.

Aos meus colegas da Equipe Pampa de Aerodesign, que me proporcionaram um aprendizado incalculável sobre o significado de trabalho de equipe e da enorme capacidade de um pequeno grupo de pessoas que realizar grandes feitos.

Agradeço à Universidade Federal do Rio Grande do Sul e a todos os servidores e professores, em especial aos professores do curso de Engenharia de Controle e Automação, pela oportunidade de aprendizado ímpar.

Por fim, gostaria de agradecer a todos que de alguma forma me ajudaram, sendo através de um incentivo ou um puxão de orelha, estando longe ou perto.

“When something is important enough,
you do it even if the odds are not in your favor.”

(Elon Musk)

RESUMO

Veículos guiados automaticamente (AGVs – *Automated Guided Vehicles*) representam uma solução popular para a movimentação de materiais em ambientes industriais devido à sua alta flexibilidade. No entanto, os sistemas de orientação mais utilizados para estes veículos demandam a instalação e manutenção de infraestrutura adicional, o que acaba por reduzir significativamente a atratividade da introdução de AGVs em uma planta. Este trabalho descreve a integração de um algoritmo para localização e mapeamento simultâneos (SLAM – *Simultaneous Localization And Mapping*) a um AGV industrial, tendo como foco o desenvolvimento e adaptação de algoritmos de modo a integrar os diferentes sistemas presentes, tanto para a operação do veículo, como para a execução dos algoritmos referidos. O método para solução do problema SLAM foi selecionado após a análise das técnicas clássicas e dos recursos disponíveis para realização dos mesmos, sendo então realizada a instalação de sensores e o desenvolvimento dos algoritmos demandados para a comunicação entre os diferentes componentes do AGV. Os mapas gerados para o ambiente de testes refletiram o espaço navegado, assim como as posições obtidas para a localização do veículo apresentaram desvios considerados aceitáveis para o caso. Entretanto, a etapa de navegação livre não teve sucesso em manter o veículo em sua trajetória. A análise dos testes de validação indica que a utilização do algoritmo SLAM para navegação em "tempo real" requer profundas alterações nos algoritmos selecionados visando melhorar o desempenho dos mesmos. Por outro lado, a metodologia adotada se mostrou viável, desde que o sistema seja capaz de suportar o intenso processamento de dados.

Palavras-chave: SLAM, AGV, posicionamento, mapeamento, navegação.

ABSTRACT

Automated guided vehicles (AGVs) are a popular solution for material handling in industrial environments due their high versatility and flexibility. However, most usual guidance systems require the installation and maintenance of additional hardware or infrastructure, which is a hinderance to the AGV's versatility. This work describes the integration of a simultaneous localization and mapping (SLAM) algorithm to an industrial AGV, focusing on the development and adaption of algorithms in order to link both the vehicle's and the SLAM's computer systems. The methodology adopted for solving the SLAM problem was selected after an analysis of the classic techniques along with the available resources, following by the installation of additional hardware to the AGV and the development of the necessary software. The mapping process allowed accurate representations of the testing environment, as well as relatively precise position values generated by the localization branch of the algorithm. However, the navigation algorithm developed was unable to keep the AGV on the stablished path. Analysis of experimental data indicates that the SLAM algorithm adopted demands strong adaptations to allow the achievement of industry-compatible performance in "real-time" navigation. Nevertheless, the adopted methodology proved to be viable provided the hardware is able to cope with the data processing demands of the algorithm.

Keywords: SLAM, AGV, positioning, mapping, navigation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFRGS – Universidade Federal do Rio Grande do Sul

AGV – *Automated Guided Vehicle*

SGV – *Self Guided Vehicle*

SLAM – *Simultaneous Localization and Mapping*

ROS – *Robot Operating System*

EKF – *Extended Kalman Filter*

USB – *Universal Serial Bus*

CAN – *Controller Area Network*

TCP – *Transmission Control Protocol*

IP – *Internet Protocol*

GPU – *Graphics Processing Unit*

LISTA DE SÍMBOLOS

x_t	Pose do robô no instante *t*.
m	Matriz que representa o mapa do ambiente.
$z_{1:t}$	Dados provenientes do sensor de distância, do instante 1 até *t*.
$u_{1:t}$	Dados referentes à movimentação das rodas do robô, do instante 1 até *t*.
$x^{[j]}$	Pose do robô contida na partícula *j*.
$\omega^{[j]}$	Peso associado à partícula *j*.
j	Número correspondente à partícula.
$\omega_t^{[j]}$	Peso associado à partícula *j* no instante *t*.
$x_t^{[j]}$	Pose do robô contida na partícula *j* no instante *t*.
n_{eff}	Número de partículas efetivas.
m_t	Matriz que representa o mapa do ambiente no instante *t*.
x_t^*	Dados de odometria aprimorados por *Scan Matching*.

SUMÁRIO

CAPITULO 1 – INTRODUÇÃO

A utilização de veículos guiados automaticamente, ou AGVs (*Automated Guided Vehicles*), em ambientes como fábricas ou depósitos é uma tendência. Essa tecnologia tem sua operação baseada no princípio popularmente denominado como “seguidor de linha”, o qual demanda a instalação de guias que o AGV utiliza como referência para se movimentar entre os diferentes destinos ao longo de seu trajeto. Esta abordagem, entretanto, tem como limitação básica o trajeto estabelecido durante a instalação do AGV: uma vez que as guias são instaladas, sua alteração é um processo dispendioso e, muitas vezes, demorado, além de demandar a parada da operação. Sabendo-se que um dos principais motivadores para o uso de um sistema AGV é sua produtividade elevada, a necessidade de paralisar operações para ajustes do mesmo é indesejada (SCHULZE, BEHLING e BUHRS, 2008).

Com a crescente demanda por produtos customizados e a maior necessidade de flexibilização dos sistemas produtivos, ambientes industriais estão sujeitos a mudanças constantes. Por exemplo, centros de distribuição podem sofrer alterações na distribuição física de suas prateleiras e *racks*, devido à necessidade de acomodar novos produtos, à variação nas demandas, ou a alterações operacionais diversas. Por conseguinte, é desejável que a solução para movimentação de materiais adotada seja capaz de acompanhar tais alterações sem requerer tempo adicional para alterações de trajetória.

Uma das soluções que visam a atender esta necessidade é o processo conhecido como localização e mapeamento simultâneos, ou SLAM (*Simultaneous Localization and Mapping*). Este tipo de sistema utiliza uma representação virtual (mapeamento) do ambiente em que o sistema AGV irá operar em conjunto com dados provenientes de sensores, que permitem a localização do veículo neste ambiente. O mapeamento do sistema é feito através de dados obtidos pelo o uso do conjunto de sensores específicos e provenientes do próprio AGV, como

de *encoders*. Em resumo, busca-se criar uma representação virtual, ou um mapa, do ambiente no qual o AGV irá operar e, de posse deste mapa, determinar com precisão a posição do veículo, com auxílio dos dados obtidos durante sua operação. Desta forma um sistema supervisório é adicionado ao processo, agregando maior conhecimento sobre o estado da planta em termos das posições dos AGVs. Isso facilita o controle do sistema e permite melhor atribuição de tarefas e trajetos aos veículos, ao mesmo tempo em que é mantido um comportamento similar ao seguimento de linha em termos de navegação, porém ocorrendo a transição do meio físico para o virtual (DURRANT-WHYTE e BAILEY, 2006).

O problema SLAM é conhecido como um dos exemplos de problemas que têm em sua essência um paradoxo. A tradução do problema para a realidade SLAM implica que a localização requer a existência prévia de um mapa, porém, para que o mapeamento seja realizado corretamente, é necessário o conhecimento da localização. Soluções clássicas para tal problema consistem na aplicação do filtro de Kalman, de modo que tanto as informações referentes à medição do sensor, como os dados de odometria, sejam incorporados, de forma a determinar a posição do robô ao mesmo tempo que o mapa do ambiente é construído. Esta abordagem, entretanto, foi aprimorada com o implementação de filtros de partículas e abordagens baseadas em grafos, de forma a melhor representar o comportamento tipicamente não linear que surge quando o algoritmo de SLAM é implementado em ambientes com grande ambiguidade, ou seja, ambientes em que não apresentam grandes distinções entre um setor e outro como, por exemplo, o corredor de um hotel.

A implementação do conceito SLAM, presumindo que o mesmo apresente desempenho similar a outras implementações típicas, busca reduzir drasticamente tanto o tempo de instalação inicial como subsequentes alterações de trajeto. A manutenção do circuito também se reduz de forma drástica, uma vez que o mesmo passa a ser virtual. Aplicações que utilizam fitas magnéticas ou faixas pintadas requerem constante atenção e manutenção, pois as faixas ou fitas estão sujeitas a desgaste, o que pode causar a completa parada da operação do sistema AGV. No caso de guias ativas, há o custo mais elevado de instalação e manutenção, além da maior dificuldade em estender a malha (DISSANAYAKE *et al.*, 2001).

Tendo em vista os benefícios do uso de sistemas SLAM (ou os problemas decorrentes do uso de fitas pintadas ou coladas), o presente trabalho busca realizar a integração de um sistema SLAM a um AGV industrial típico, cuja navegação será desenvolvida utilizando princípios do seguimento de trajetória.

CAPITULO 2 – REVISÃO BIBLIOGRÁFICA

2.1 AGV

De acordo com Groover (2008), veículos guiados automaticamente (AGVs) são sistemas de manipulação de materiais que empregam veículos operados de forma autônoma ao longo de caminhos pré-estabelecidos. A implementação de AGVs como sistemas de movimentação de materiais apresenta vantagens quando comparada a sistemas tradicionais, como esteiras de transporte, uma vez que seus caminhos permanecem livres na ausência do veículo. Além disso, um único veículo em um sistema AGV é capaz de transportar uma grande gama de produtos diferentes a diversos pontos, não estando limitado a um único circuito.

Os sistemas de orientação são os métodos através dos quais AGVs são capazes de seguir os trajetos pré-determinados. Segundo Groover (2008), tais sistemas de orientação podem ser classificados em sistemas de cabos embutidos (ativos), faixas pintadas (passivos) ou veículos autoguiados (SGV). Sistemas passivos, como fitas magnéticas e sistemas que utilizam faixas pintadas são sistemas ditos reativos (SACCHETIN, 2005), ou seja, são capazes de detectar discrepâncias imediatas entre a posição desejada e a real, reagindo de acordo. No entanto, tais sistemas não têm qualquer informação sobre o trajeto à frente, dependendo de algum outro sistema para receber qualquer informação referente ao trajeto futuro. Adicionalmente, tais sistemas requerem infraestrutura adicional para operação do veículo, como pintura do piso e instalação de fitas, de modo que qualquer alteração de *layout* do circuito requer alterações físicas, exigindo a parada da planta.

Diferente dos sistemas descritos anteriormente, sistemas que empregam veículos autoguiados não demandam trajetos fisicamente definidos; o caminho que o veículo deve seguir é definindo computacionalmente (GROOVER, 2008), sendo o método de operação variante de acordo com o sensor empregado no veículo e o software utilizado. Tais sistemas exigem me-

nores tempos de instalação por não demandarem alterações na infraestrutura do ambiente, sendo que, em geral, as rotas são definidas com antecedência, caso exista um mapa pré-definido do ambiente. Adicionalmente, pelo fato de o veículo possuir informação referente ao trajeto completo, a produtividade pode ser melhorada através do aumento da velocidade, uma vez que o sistema permite o cálculo da distância até curvas ou pontos de parada. Bifurcações e junções, pontos tipicamente problemáticos em sistemas reativos, também não representam um problema para sistemas autoguiados, tendo em vista que os veículos possuem somente informações referentes ao seu trajeto atual, não sendo afetado por trajetos adicionais.

2.2 SLAM

A localização e mapeamento simultâneos (*Simultaneous Localization and Mapping*) é um dos problemas mais fundamentais da robótica (THRUN, BURGARD e FOX, 2005). O problema SLAM surge quando o robô não possui conhecimento prévio do ambiente no qual irá operar e não possui acesso direto às suas próprias poses. Como alternativa, os dados a que tem acesso são as medidas de distância $z_{1:t}$, provenientes de sensores, e as variáveis de controle $u_{1:t}$, correspondentes aos deslocamentos das rodas. Em resumo, o termo SLAM se refere ao processo de obter o mapa de um ambiente ao mesmo tempo em que o robô se localiza neste mapa.

Existem duas variantes principais do problema SLAM: *Online SLAM* e *Full SLAM* (THRUN, BURGARD e FOX, 2005). *Online SLAM* implica determinar a localização atual do robô juntamente com o mapa. A distribuição representante desta variante é apresentada na Equação (1):

$$p(x_t, m | z_{1:t}, u_{1:t}) \tag{1}$$

onde x_t representa a pose do robô no instante t, m representa o mapa e $z_{1:t}$ e $u_{1:t}$ representam respectivamente as medições e os sinais de controle. Neste formato, dados sobre poses passadas são desconsiderados para a obtenção da pose atual. A segunda variante, *Full SLAM*, busca obter a pose atual do robô ao longo de toda sua trajetória $x_{1:t}$, juntamente com o mapa, cuja distribuição que a representa é dada pela Equação (2):

$$p(x_{1:t}, m | z_{1:t}, u_{1:t}) \tag{2}$$

2.2.1 Filtros de Kalman

O mais antigo e mais influente algoritmo para a resolução do problema SLAM é baseado no filtro de Kalman estendido, ou EKF - *Extended Kalman Filter*. Basicamente, a abordagem que implementa o EKF utiliza a associação de dados por máxima verossimilhança para a variante *Online SLAM*. Dessa forma, tal solução é sujeita a uma série de aproximações e suposições limitantes. A primeira destas limitações é a necessidade da utilização de mapas baseados em características (*Feature-based maps*). Mapas no EKF são compostos de uma série de marcos pontuais. Por motivos computacionais, o número de marcos é geralmente menor que 1000 (BAILEY e DURRANT-WHYTE, 2006), além de demandar que os marcos apresentem baixa ambiguidade. Por esta razão, implementações baseadas no *EKF* requerem significante desenvolvimento de detectores de características, de forma que os dados obtidos durante o deslocamento de um robô ou veículo possam identificar os elementos de interesse correspondentes ao modelo virtual.

A segunda limitação do uso de sistemas baseados em EKF é que uma das suposições fundamentais deste filtro é a existência de ruído que pode ser modelado por uma distribuição Gaussiana. A incerteza presente na estimativa da pose deve ser pequena quando comparada às tolerâncias necessárias, pois, caso contrário, a linearização presente no EKF tende a introduzir erros acima dos níveis aceitáveis.

Por fim, o algoritmo baseado no EKF se limita a medições positivas, o que significa que ele somente pode processar avistamentos positivos de marcos. Informações negativas, provenientes da ausência de marcos, não podem ser processadas. Sendo assim ambientes ambíguos podem gerar erros de posicionamento devido à não detecção da falta marcos distintos para as diversas posições.

Filtros de Kalman operam em essência, através de uma estimação de estados, sendo consideradas as variáveis de pose do robô (x, y, θ) e as posições de cada um dos marcos (m_x, m_y), juntamente a uma matriz de covariância, que descreve a incerteza associada à composição das posições.

Para que seja computacionalmente viável realizar o processamento no tempo necessário, a matriz que representa o espaço de estados deste sistema não pode ser demasiadamente grande, o que significa que o número máximo de marcos deve ser definido de forma criteriosa. Assim, variantes da implementação básica com o filtro de Kalman estendido buscam reduzir o

número de marcos necessários para que o robô tenha valores satisfatórios de precisão em sua localização.

2.2.2 Filtros de Partículas

Uma alternativa popular às técnicas Gaussianas, como filtro de Kalman, são os filtros não paramétricos. Estes filtros não dependem de uma forma fixa para a estimativa da posterior (probabilidade condicional associada a um evento após a análise de eventos passados). Ao invés disso, eles aproximam a posterior por um número finito de valores. Uma destas implementações é o filtro de partículas. O filtro de partículas é uma implementação alternativa do filtro de Bayes (THRUN, BURGARD e FOX, 2005) cuja ideia básica é que a pose final e o mapa correspondente são representados através de um conjunto de amostras. A Equação (3) relaciona as hipóteses de estado ($x^{[j]}$) aos pesos associados com cada hipótese ($\omega^{[j]}$), onde cada combinação de *x* e ω é chamada de partícula, sendo *j* correspondente ao número da partícula. Esta representação é uma aproximação não-paramétrica, podendo representar uma gama muito maior de distribuições do que Gaussianas, por exemplo (LEONARD, DURRANT-WHITE, 1991).

$$x = \left\{\langle x^{[j]}, \omega^{[j]}\rangle\right\}_{j=1,\ldots,J} \tag{3}$$

O peso de cada hipótese determina a probabilidade de que esta corresponda ao estado real do robô. A obtenção dos pesos de cada hipótese é realizada por meio de um processo denominado amostragem por importância, onde uma distribuição proposta é comparada com a distribuição alvo segundo pesos, calculados segundo a Equação (4) (THRUN, BURGARD e FOX, 2005).

$$\omega_t^{[j]} = \frac{alvo\left(x_t^{[j]}\right)}{proposta\left(x_t^{[j]}\right)} \tag{4}$$

A etapa fundamental do algoritmo que implementa o filtro de partículas é conhecida como reamostragem ou (*resampling*) (THRUN, BURGARD e FOX, 2005). Nesta etapa, partículas são selecionadas do conjunto temporário $\overline{X_t}$, e a probabilidade de que cada partícula seja selecionada é dada pelo peso de sua importância. Algumas das técnicas utilizadas para reamostragem são o método da roleta (NEULAND, 2014) e a amostragem estocástica uniforme (MARQUES, 2014). Esta etapa não é obrigatória na implementação clássica do filtro de partículas, porém, uma vez que o algoritmo opera com um número limitado de partículas, tal

processo evita que muitas partículas representem estados pouco prováveis. O referido processo, porém, acarreta riscos como, por exemplo, a eliminação de amostras relevantes, fenômeno conhecido como esgotamento de partículas (*particle depletion*). Buscando evitar que isto ocorra, a reamostragem geralmente é condicionada a indicadores como o número efetivo de partículas (*Number of effective particles*) (GRISETTI, STACHNISS e BURGARD, 2005), que consiste em uma medida empírica de quão bem a distribuição alvo é aproximada pelas amostras das propostas, sendo dada pela Equação (5).

$$n_{eff} = \frac{1}{\sum_i \left(\omega_t^{[i]}\right)^2} \tag{5}$$

onde n_{eff} descreve "a variância dos pesos normalizados das partículas" (GRISETTI, STACHNISS e BURGARD, 2005), sendo que a operação requer que os pesos $\omega_t^{[i]}$sejam normalizados de forma a se obter um valor com significado. Assim a reamostragem ocorre apenas quando n_{eff} se reduzir a um valor menor que o limite estabelecido, o qual representa a proporção de partículas que devem ser mantidas, um valor típico a ser utilizado é $N/2$, onde N representa o número total de partículas. Neste caso, 50% das partículas irão "sobreviver" à etapa de reamostragem.

2.2.3 Rao-Blackwellization

Uma das desvantagens mais significativas presentes nos filtros de partículas é que a amostragem em grandes espaços dimensionais pode ser ineficiente, sendo que a alternativa em tais situações requer que algumas das variáveis sejam marginalizadas. Uma de tais técnicas é referida como *Rao-Blackwellization* (DOUCET *et al.*, 2000), por ser relacionada ao teorema de Rao-Blackwell. Dado que uma distribuição de probabilidade conjunta de duas variáveis dependentes pode ser descrita de acordo com a Equação (6).

$$p(a,b) = p(b|a)p(a) \tag{6}$$

onde $p(a,b)$ representa a probabilidade da ocorrência de duas variáveis, $p(b|a)$ representa a probabilidade da ocorrência da variável b dado a e $p(a)$ representa a probabilidade da ocorrência de a.

De acordo com o teorema de Rao-Blackwell, se existir alguma maneira em que os parâmetros da distribuição $p(b|a)$ podem ser computados de forma fechada, através de um número finito de operações, então $p(a)$ nunca será um estimador pior do que $p(a,b)$, porém, frequentemente, é um estimador melhor, ou seja, possui variância menor ou igual.

A consequência para o problema SLAM é que considerando a formulação do problema *Full SLAM* conforme a Equação (7), ao aplicar a regra da cadeia juntamente com as propriedades de Markov se pode obter a expansão correspondente à Equação (8).

$$p(x_{0:t}, m_{1:M}|z_{1:t}, u_{1:t}) \tag{7}$$

$$p(x_{0:t}|z_{1:t}, u_{1:t})p(m_{1:M}|x_{0:t}, z_{1:t}) \tag{8}$$

A distribuição $p(m_{1:M}|x_{0:t}, z_{1:t})$ pode ser estimada, uma vez que haja uma estimativa da pose $x_{0:t}$ em conjunto com os dados de medição $z_{1:t}$ os quais são obtidos diretamente do sensor. Desta forma, a trajetória $p(x_{0:t}|z_{1:t}, u_{1:t})$ pode ser estimada implementando um filtro de partículas similar ao algoritmo de localização de Monte Carlo (*Monte Carlo Localization*) (THRUN, BURGARD e FOX, 2005), onde a distribuição proposta consiste do modelo de movimentação, Equação (9), e a correção consiste do modelo de observação, Equação (10).

$$x_t^{[j]} \sim p(x_t|x_{t-1}, u_t) \tag{9}$$

$$\omega_t^{[j]} = \frac{alvo}{proposta} \propto p(z_t|x_t, m) \tag{10}$$

2.2.4 Scan-Matching

Conforme estabelecido pelo processo de *Rao-Blackwellization*, o ponto de partida para o processo de localização consiste na estimativa da localização atual, baseado na medição do sensor $z_{1:t}$ e dos dados de odometria $u_{1:t}$, os quais tipicamente possuem alto nível de ruído associado a seus valores, uma vez que se baseiam na movimentação das rodas do robô, que, devido a fatores como terreno irregular ou escorregamento, podem não corresponder ao deslocamento real. A consequência direta disto, é que se faz necessário um alto número de partículas, de forma a modelar suficientemente o ruído associado à movimentação. Pelo fato de os mapas que representam o ambiente serem tipicamente grandes, o aumento do número de partículas gera um impacto computacional, tanto no tempo de processamento, como no consumo de memória, fazendo-se necessária, assim, a implementação de um processo capaz de aprimorar os dados de odometria antes da aplicação do filtro de partículas.

Uma das técnicas aplicadas para melhorar a precisão dos dados de odometria é conhecida como *scan-matching* (OLSON, 2009), a qual busca maximizar a verossimilhança entre a pose a mapa atuais relativos aos anteriores conforme a Equação (11).

$$x_t^* = argmax\{p(z_t|x_t, m_{t-1})p(x_t|u_t, x_{t-1}^*)\} \tag{11}$$

onde x_t^* representa os dados de odometria aprimorados através da técnica *scan-matching*. Tal implementação proporciona uma correção para a pose localmente consistente, tendo como resultado a redução no número de partículas necessárias uma vez que o erro presente nos dados de entrada é reduzido.

2.3 ESTADO DA ARTE

As soluções clássicas para o problema SLAM podem ser classificadas como filtragem, na qual o problema é abordado através de uma estimação de estados na qual o estado do sistema consiste na posição atual do robô e de seu mapa correspondente. Alternativamente, existem abordagens classificadas como suavização, na qual, a trajetória completa do robô é estimada utilizando todas as medições realizadas e, tipicamente, depende de técnicas de mínimos quadrados para minimização de erro.

A abordagem mais recente é baseada em grafos (*graph-based*), e envolve a construção de um grafo cujos vértices representam poses ou marcos observados pelo robô, e as arestas entre dois vértices codificam uma medição do sensor, que restringe a conexão entre as poses (XIANG *et al.*, 2015). Desse modo o problema de estimação é simplificado por intermédio da abstração das medições brutas dos sensores, sendo que tais medições são substituídas pelas arestas no grafo, que podem ser definidas como “medições virtuais” (ANNAIYAN, OLIVARES-MENDEZ e VOOS, 2017).

O paradigma baseado em grafos divide-se em duas etapas fundamentais: a construção do grafo e a determinação da distribuição de poses mais prováveis, dadas as medições. A etapa de construção é tipicamente denominada *front-end*, enquanto que a segunda etapa é referida como *back-end*. Este paradigma consiste em um conjunto complexo de algoritmos de otimização que buscam associar corretamente a medição mais recente (vértice) com a aresta que apresenta a maior probabilidade de representar o deslocamento entre o instante anterior e o atual (GRISETTI *et al.*, 2010). Tal abordagem é recente, uma vez que as formulações iniciais eram de uma complexidade comparavelmente alta, devido à necessidade da minimização do erro, por meio de técnicas clássicas.

Métodos para SLAM ainda buscam obter robustez de forma a serem difundidos especialmente em aplicações industriais (CADENA *et al.* 2016). As diversas etapas para a implementação de um algoritmo SLAM são foco de uma grande quantidade de pesquisas, sendo

abordagens utilizando filtros de Kalman e filtros de partículas já consolidadas e implementadas em análises tanto de desempenho como de aplicabilidade, (MEGALINGAM *et al.* 2018), (PAJAZITI e AVDULLAHU, 2014), (PEI, WU e ZHANG, 2014).

CAPITULO 3 – ESTUDO DE CASO

Este capítulo busca expor o ponto de partida para o desenvolvimento do trabalho, analisando os diferentes métodos para a solução do problema SLAM, apresentando os recursos selecionados tanto referentes a *software* quanto a *hardware*.

3.1 SELEÇÃO DO MÉTODO DE SOLUÇÃO DO PROBLEMA SLAM

A resolução do problema SLAM, em essência, prega que não existe uma solução ideal para todos os casos. A escolha do método mais apropriado é largamente dependente da aplicação, sendo que a solução ótima geralmente só pode ser determinada após o teste prático de cada uma das diferentes técnicas, de forma a determinar qual apresenta o melhor desempenho, sendo a própria definição de desempenho variante de acordo com os equipamentos disponíveis, o local de testes e o objetivo final.

Um dos pontos de maior destaque quando se aborda o problema SLAM é a linearidade do sistema no qual o algoritmo será implementado (MONTEMERLO *et al*., 2002). Algoritmos baseados em filtros de Kalman são preferidos quando o sistema pode ser suficientemente descrito por funções lineares, cujas incertezas também podem ser adequadamente representadas por distribuições Gaussianas. Sistemas cujos comportamentos possuem não-linearidades muito expressivas, requerem abordagens não-paramétricas tais como filtros de partículas, de modo que a distribuição das incertezas e as funções que descrevem o sistema são estimadas através do próprio filtro podendo representar melhor sistemas não lineares.

O fator que foi considerado como decisório, para seleção de um filtro, foi o tipo de mapa gerado pela etapa de mapeamento. Algoritmos que têm como base a utilização do filtro de Kalman geram mapas baseados em características (ou *Feature-based*) (SKRZYPCZYNSKI,

2009). Tais mapas são formados registrando a posição de uma série de marcos detectados no ambiente mapeado, gerando mapas similares ao que pode ser observado na Figura 1.

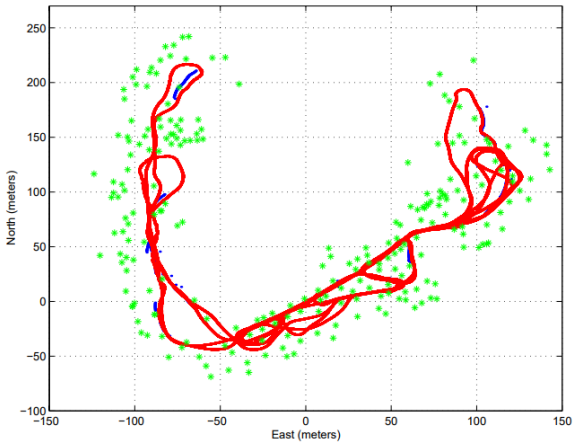


Figura 1: Exemplo de mapa baseado em características (NIETO *et al.*, 2003).

A segunda modalidade de mapa é denominada volumétrica (YATIM e BUNIYAMIN, 2015). Um exemplo de tal mapa é apresentado na Figura 2. Estes são mapas que se baseiam na criação de uma grade com células de um determinado tamanho as quais são avaliadas por meio das medições dos sensores. Este mapeamento é baseado em filtros não-paramétricos como filtros de partículas e filtros de histogramas.

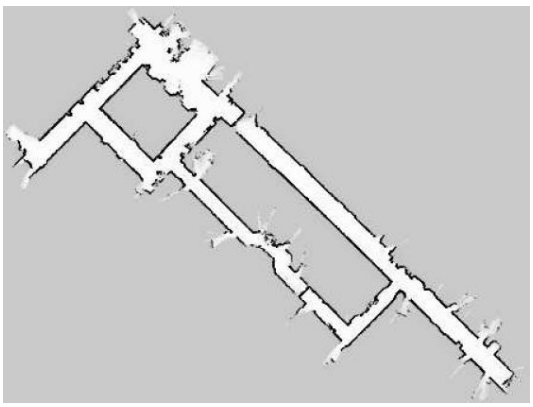


Figura 2: Exemplo de mapa volumétrico (THRUN, BURGARD e FOX, 2005).

Devido à motivação deste trabalho ser a busca pela navegação do AGV, sem a necessidade de instalação de infraestrutura adicional para orientação, mapas volumétricos apresentam melhor solução, uma vez que determinam se cada célula se encontra ocupada ou não, sendo, desta forma, possível estabelecer um caminho livre de obstáculos entre dois pontos, enquanto que, em mapas baseados em características, não se pode garantir que não haja obstáculos entre dois ou mais marcos.

3.2 SELEÇÃO DO SENSOR

As principais soluções aplicadas para o mapeamento do ambiente incluem sensores laser, sonares e câmeras. As primeiras soluções para o problema SLAM aplicavam sensores sonares. Tais sensores têm como princípio de operação a medição do tempo entre a emissão e recepção de ondas sonoras como forma de determinação de distâncias. Tais sensores, apesar de seu baixo custo, apresentam um alto nível de ruído (RIISGAARD e BLAS, 2005). Como consequência, o tempo necessário para que as medições com tal sensor fossem capazes de produzir um mapa com fidelidade suficiente para a correta localização do robô era proibitivamente grande, além de apresentar sérias restrições quanto à operação em ambientes ruidosos.

Atualmente, o tipo de sensor mais popular para experimentos envolvendo SLAM são câmeras, uma vez que o custo de dispositivos com alta resolução e alta taxa de quadros por segundo vêm se tornando cada vez menor. A medição de distâncias com a utilização de imagens, entretanto, não é direta, de modo que são necessários algoritmos de processamento de imagem para que tais dados possam ser obtidos. Este processamento adicional impacta na taxa de atualização, que pode ser obtida com *hardware* limitado, o que, por sua vez, pode tornar proibitiva a operação em “tempo real” (CHONG *et al*., 2015).

Como previamente mencionado, os algoritmos empregados para o mapeamento e localização possuem um elevado custo computacional, tanto em tempo de processamento como consumo de memória, ao ponto de que a operação em tempo real de veículos tem sua máxima velocidade limitada pela capacidade do computador de calcular a posição atual e gerar as correções necessárias. Sensores a laser ou sonares não demandam capacidade computacional para pré e pós-processamento, uma vez que os dados fornecidos já correspondem às distâncias medidas. Desse modo, em contrapartida ao custo mais acessível de sistemas de imagem quando comparado a sensores laser, existe um aumento na complexidade do algoritmo de processamento de dados, e de seu inerente custo computacional.

Apesar de possuírem um custo elevado, sensores a laser são os que melhor se ajustam à solução do problema SLAM, uma vez que aliam a elevada precisão à baixa sensibilidade a ruídos. Dessa forma, para os propósitos deste trabalho, o único ponto de desvantagem do sensor a laser é seu preço.

O sensor modelo Tim 571 é um *scanner* laser opto-eletrônico, que é capaz de esquadrinhar seu entorno com um passo angular definido. As medidas de distância utilizam coordena-

das polares num plano de duas dimensões (2D). De acordo com a documentação fornecida pelo fabricante SICK, o alcance máximo é de 25 metros, dentro de uma zona de 270°. A medição é realizada através da determinação do tempo entre a emissão e detecção da reflexão do feixe laser (*time-of-flight measurement*), o mesmo princípio empregado por radares e sonares. A frequência de medição é de 15 Hz, sendo a resolução angular de 0,3°.

3.3 ALGORITMO IMPLEMENTADO

Tendo como base o método a ser implementado e o sensor de localização selecionado, o passo seguinte incluiu a escolha da plataforma utilizada para o desenvolvimento do algoritmo. A plataforma ROS destaca-se pelo seu amplo emprego em pesquisa e desenvolvimento no ramo de robótica, possuindo uma série de bibliotecas de código aberto destinadas a acelerar o desenvolvimento de plataformas de teste. Uma das bibliotecas disponíveis se chama gmapping®, mantida pelo projeto Open SLAM. Tal biblioteca, de acordo com sua página de distribuição, consiste de uma implementação em ROS do algoritmo SLAM proposto por Grisetti, Stachniss e Burgard (2005; 2007). Esta tem como base a utilização de um filtro de partículas que implementa o teorema de Rao-Blackwell (*Rao-Blackwellized Particle Filter*), técnicas adicionais como *Scan-Matching* são implementadas buscando reduzir o número de partículas necessárias, de forma que o algoritmo tenha um melhor desempenho tanto em termos de tempo de execução como consumo de memória. Técnicas para a reamostragem condicional também são implementadas com o objetivo de reduzir o problema do esgotamento de partículas.

3.4 ROS

ROS (*Robot Operating System*) é definido de acordo com sua página introdutória (ROS.org, 2018), como um sistema operacional de código aberto, que visa a simplificar a implementação de experimentos em robôs. O foco principal da plataforma ROS é prover uma grande seleção de bibliotecas e pacotes generalistas, que buscam oferecer abstração de *hardware,* comunicação entre diferentes serviços, transmissão assíncrona e armazenamento de dados em uma espécie de banco de dados embutido. Tais aspectos facilitam a implementação de algoritmos, uma vez que o desenvolvimento de aplicações similares demanda a integração de uma grande quantidade de bibliotecas e métodos que, em essência, não têm nenhuma relação com o objetivo da aplicação como por exemplo, a transmissão de informações entre dife-

rentes serviços, a qual demanda experiência significativa em tais implementações para que isto seja feito de forma eficiente.

A utilização de algoritmos desenvolvidos para operação dentro do ROS, está sujeita a algumas particularidades. Diferentemente do que ocorre normalmente com a execução de algoritmos independentes em um sistema operacional típico, quando executados simultaneamente, dados podem ser disponibilizados por meio de variáveis acessíveis a todos os programas que operam em conjunto, sem a necessidade de desenvolvimento adicional. O ato de disponibilizar os dados nestas "variáveis globais" é denominado *publicação*, e o ato de consultar tais variáveis é denominado *assinatura*. Tais termos serão utilizados no restante deste trabalho para descrever tais atos.

Os diversos módulos que operam de forma independente ou em conjunto com os demais são denominados *nós*. Por exemplo o pacote *gmapping*®, contém unicamente o nó responsável pelos cálculos referentes ao processo SLAM, porém, os dados demandados para operação devem ser publicados por nós especificamente desenvolvidos de acordo com o *hardware* disponível. Assim, a operação do AGV utilizando o processo SLAM não consiste na execução de um único algoritmo, mas na correta integração de uma grande quantidade de programas, ou nós.

3.5 AGV

A plataforma de testes utilizada para o teste de operação do algoritmo SLAM foi o AGV modelo *myPresto* fabricado pela empresa *Spark*, Figura 3. Este AGV é um modelo de rodas centrais, que se movimenta através de um acionamento diferencial. Tal modelo possui *encoders* ligados diretamente aos eixos de cada uma das duas rodas responsáveis pela tração do mesmo.

Figura 3 - AGV modelo myPresto (Spark, 2018).

CAPITULO 4 – ADAPTAÇÃO DO VEÍCULO

Este capítulo dedica-se às várias etapas demandadas para a adaptação do AGV, a fim de operar utilizando o algoritmo SLAM. Dentre estas fases, incluem-se a instalação e a configuração de sensores, em conjunto com qualquer *hardware* adicional demandado, e também os algoritmos desenvolvidos com a finalidade de efetuar a comunicação ou realizar o processamento dos dados provenientes dos sensores.

4.1 INSTALAÇÃO

4.1.1 Sensor Laser

Para a execução dos testes, foi instalado em um veículo AGV modelo *myPresto* um sensor modelo *Tim571* (Figura 4), alimentado pela própria bateria do veículo, que fornece uma tensão nominal de 24 V. Este sensor utiliza para transmissão de dados uma conexão Ethernet, que foi ligada diretamente ao computador responsável pela operação do ambiente ROS, doravante denominado “computador principal”.

A instalação do sensor requer o ajuste de uma série de parâmetros para que os dados sejam obtidos por um nó ROS. Seguindo o procedimento de ajuste descrito no manual do fabricante (SICK, 2018), e com o auxílio do *software* fornecido para o ajuste do sensor, as configurações de endereçamento de transmissão de dados foram alteradas para a operação em conjunto com o computador.

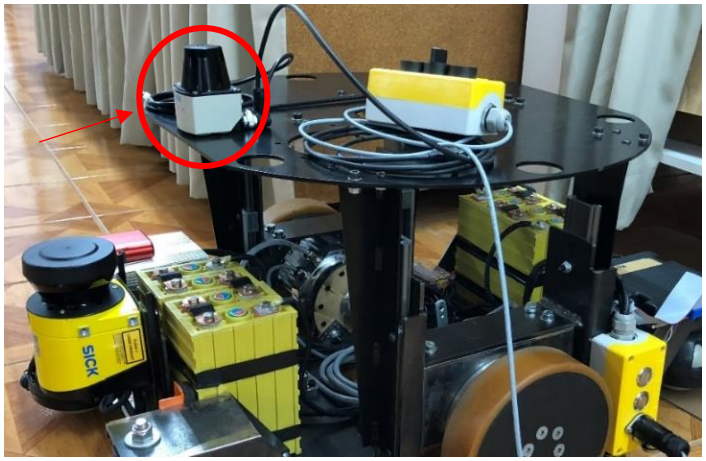


Figura 4 - Sensor Tim571 instalo ao AGV.

4.1.2 *Encoders*

Cada um dos motores presentes no AGV contém um *encoder* diretamente ligado ao seu eixo, fornecendo valores referentes ao deslocamento dos mesmos. Tais valores, no entanto, não são atualmente utilizados diretamente na operação do AGV, sendo unicamente necessários para o controlador dos motores realizar a sequenciamento das bobinas para o correto acionamento dos mesmos.

O AGV possui como meio de comunicação entre seus componentes uma rede CAN bus (*Controller Area Network*). Dessa forma, o controlador do AGV teve de ser programado de forma a transmitir os valores dos *encoders* na rede CAN. Tendo como base os valores máximo e mínimo de ± 2147483648 contagens (32 bits), conforme indicado pelo manual do controlador dos motores, a codificação dos valores dos *encoders* para transmissão foi feita conforme ilustrado pela Figura 5. Sendo utilizado o protocolo CANopen para a comunicação.

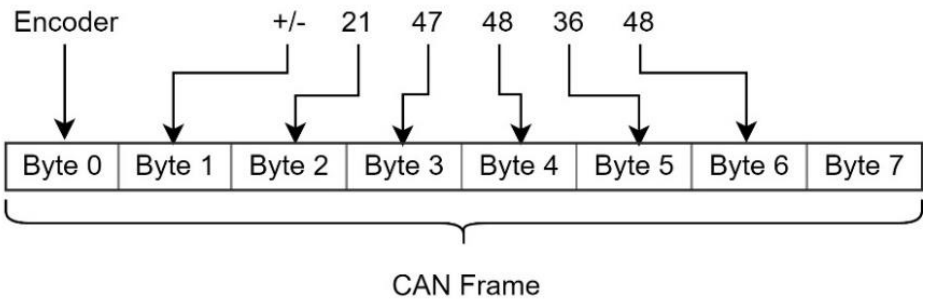


Figura 5 - Codificação para transmissão CAN.

Visando a simplificar tanto do processo de codificação, quanto a decodificação dos valores, foi desenvolvida uma mensagem de 7 bytes. O primeiro Byte determina se o valor corres-

ponde ao *encoder* da direita ou esquerda, o segundo byte define se o valor é positivo ou negativo e cinco Bytes adicionais contêm o valor absoluto do respectivo *encoder*, onde cada byte está duas ordens de grandeza distante do seguinte, restando ainda um Byte não utilizado.

Como o computador principal não possui uma interface para conexão direta à rede CAN do AGV, um computador adicional, modelo Raspebrry Pi 3 B, foi escolhido. O Rapberry Pi 3 modelo B é um computador que possui dimensões similares às de um cartão de crédito. Conforme presente nas especificações fornecidas pelo fabricante, a plataforma emprega um processador ARMv8 de quatro núcleos com capacidade de processamento de 1,2 GHz. A fim de adicionar uma interface CAN ao Raspberry Pi um *shield* CAN foi conectado através dos pinos digitais. Como tal *shield* originalmente foi densenvolvido para a utilização em conjunto com microcontroladores Arduino e seu respectivo *hardware*, foram necessárias alterações no *Kernel* do sistema operacional para que a interface fosse reconhecida.

O computador Raspberry Pi representa, portanto, uma interface intermediaria, lendo os valores dos *encoders* diretamente da rede CAN do AGV, decodificando os mesmos e então retransmitindo-os para o computador principal através de uma conexão TCP/IP por intermédio de uma rede *wireless*.

4.2 INTEGRAÇÃO DE DADOS

4.2.1 Sensor Laser

O sensor Tim 571 foi diretamente ligado ao computador principal, sendo, portanto, necessário o monitoramento da conexão para leitura dos dados e decodificação dos mesmos. Para isso, foram consultados os manuais técnicos fornecidos pelo fabricante (SICK) nos quais são descritos os diversos “telegramas” utilizados para troca de informações entre o sensor e o computador ao qual está conectado.

A fim de realizar a conexão ao endereço IP atribuído ao sensor, transmitir os telegramas responsáveis por iniciar a transmissão de dados e receber e decodificar as valores das medições, foi necessário o desenvolvimento de um pacote ROS. Cada medida realizada pelo sensor laser representa um ângulo e uma distância em coordenadas polares. O sensor possui uma resolução de 0,3° e uma faixa de medição de 270°, resultando num total 900 valores a cada varredura sendo que cada varredura ocorre à uma taxa de atualização do sensor de 15 Hz.

Por fim, de modo a disponibilizar os valores para o nó *gmapping*®, os dados do escaneamento foram publicados no tópico *scan* de acordo com o recomendado pela referência de navegação ROS. De forma a avaliar se os dados provenientes do sensor eram coerentes foi utilizado o *software* Rviz, capaz de monitorar informações publicadas nos tópicos ROS. A Figura 6 apresenta o resultado de uma varredura do sensor laser Tim 571 na área de testes da empresa Spark, empresa na qual o autor atua como estagiário. onde foram realizados os experimentos. A comparação entre os pontos medidos e a planta baixa do local, permitiu uma primeira avaliação da qualidade das medidas obtidas pelo sensor laser, considerado adequado à utilização proposta.

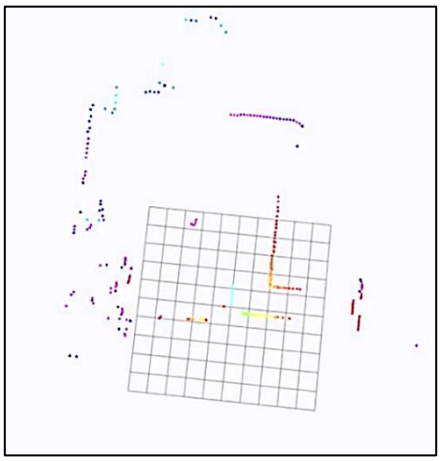


Figura 6 - Medições correspondentes a uma varredura do sensor laser Tim 571.

4.2.2 Odometria

Os valores de contagem de pulsos dos *encoders* são obtidos pelo controlador dos motores, e transmitidos em uma rede CAN, sendo, por fim, retransmitidos por um computador Raspberry Pi através de uma rede *wireless* pelo protocolo TCP/IP para o computador responsável pelo processamento do algoritmo SLAM. Assim, de modo a receber tais valores, foi desenvolvido um pacote ROS capaz de se conectar ao Raspberry Pi e, deste modo, recuperar a contagem de pulsos de cada um dos *encoders*.

A etapa de conexão utiliza o mesmo cliente TCP já desenvolvido para leitura dos dados do sensor laser. Os dados de odometria são baseados nas contagens de pulsos dos *encoders*, movimentos no sentido horário dos motores geram contagens positivas, enquanto que movimentos anti-horários geram contagens negativas. Os valores requeridos consistem dos deslocamentos nos eixos x e y, em conjunto com a orientação e também das velocidade nos eixos

x, y e velocidade angular. Utilizando como referência o plano de coordenadas do AGV, o eixo x configura movimentos frontais e traseiros, enquanto que o eixo y representa movimentos laterais. Como o AGV não é capaz de movimentos laterais, o deslocamento no eixo y pode ser desconsiderado, sendo avaliados assim os deslocamentos lineares, correspondentes ao eixo x, e angulares do veículo correspondentes ao ângulo θ.

De forma a determinar os deslocamentos, foi primeiramente necessário estabelecer a relação entre a quantidade de pulsos detectados pelos *encoders* e a distância percorrida. Para tanto, foram realizados testes onde o AGV era movimentado manualmente a uma distância de 0,5 m ± 1 mm, sendo registrada a diferença entre o valor apresentado por cada *encoder* antes e após a movimentação. Após 10 repetições deste teste, foi estabelecida uma razão de 64246 pulsos por metro em média, para ambos motores, com uma variação de 0,2% entre a maior e menor contagem de pulsos.

De posse da razão entre a quantidade de pulsos e o deslocamento, o algoritmo responsável pela obtenção dos dados diretamente do Raspberry Pi foi adaptado. Neste ponto, foi possível determinar a distância total percorrida, pelas rodas esquerda e direita. Assim, para obter o deslocamento linear (Δ_x), foi realizado o cálculo presente na Equação (12) .

$$\Delta_x = \frac{\Delta_e + \Delta_d}{2} \tag{12}$$

onde Δ_e e Δ_d representam as distancias percorridas pelas rodas da esquerda e direita respectivamente, enquanto que o deslocamento angular (Δ_θ) é obtido através da Equação (13).

$$\Delta_\theta = \frac{\Delta_e - \Delta_d}{l_{AGV}} \tag{13}$$

onde l_{AGV} representa a largura do AGV, o deslocamento obtido é dado em radianos.

De posse dos deslocamentos, as velocidades lineares e angulares são facilmente obtidas, uma vez que o algoritmo registra o tempo decorrido entre o ciclo atual e o anterior, e, de forma similar, registra o valor mais recente dos *encoders*, de modo a ser comparado com a próxima leitura. Logo, similarmente ao realizado com os dados provenientes do sensor laser, os dados de odometria foram publicados no tópico *odom* de acordo com o recomendado pela referência ROS.

4.2.3 Planos de referência

A execução do algoritmo SLAM depende do correto relacionamento entre os diferentes planos de referência de cada um dos sensores. O plano de referência ao qual as medições devem se relacionar é o plano do AGV, o qual é denominado *base_link* dentro da "árvore" de relações entre planos. Como, uma vez instalados, os sensores têm suas posições fixas em relação ao centro do AGV, foi decidida pela execução de transformadas fixas entre os diferentes planos.

Devido ao fato de o AGV possuir rodas centrais, o centro do plano de referência de odometria coincide com o plano de referência do veículo. Assim, a transformada estática consiste simplesmente de distâncias e rotações nulas em relação aos três eixos, x, y e z.

O sensor laser Tim 571 se encontra instalado em uma posição deslocada em relação ao centro do veículo. Após a realização de medições, pode ser determinado que o plano de referência do sensor laser possuía um deslocamento de 0,24 m no eixo x e 0,30 m no eixo z. Tais valores foram assim utilizados na transformação estática estabelecendo a relação entra os planos distintos.

Por fim, o mapeamento do ambiente produz um plano de referência adicional correspondente ao mapa em si. Todavia, o AGV movimenta-se em relação ao mapa de modo que não exista uma transformação fixa entre tais planos de referência. Tal relação só pode ser determinada tendo posse a posição do veículo. O próprio algoritmo SLAM relaciona o mapa gerado e os dados de odometria, por meio dos quais se obtém uma relação indireta entre o veículo e o mapa.

4.3 TESTES OPERACIONAIS

Após o desenvolvimento de todos os pacotes adicionais demandados para a obtenção dos dados necessários para a execução do algoritmo SLAM, o pacote gmapping® foi executado, utilizando as configurações padrões. O AGV foi operado utilizando um *joystick*, uma vez que o objetivo era a validação das etapas de localização e mapeamento.

O processo de mapeamento pode ser observado em seus estágios iniciais na Figura 7. Pode-se notar que os espaços vazios são preenchidos por um tom cinza claro, enquanto que obs-

táculos são representados por pontos ou linhas pretas. Pontos coloridos representam medições do sensor laser. O AGV se encontra na base da seta vermelha na parte inferior da imagem. O mapa é representado no *software* Rviz em escala 1:1 porém a imagem foi afastada de modo à englobar todo mapa.

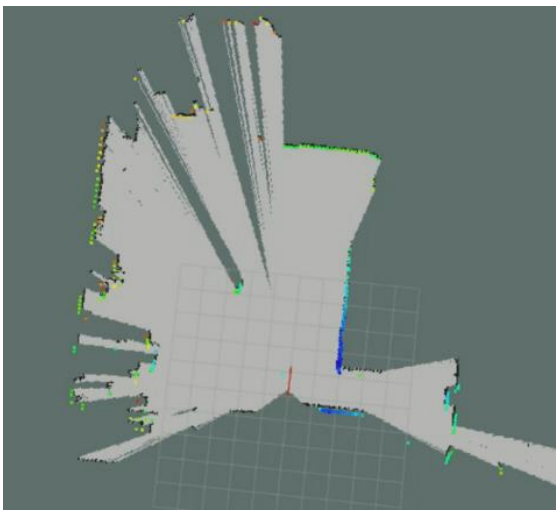


Figura 7 - Mapeamento do ambiente.

4.4 NAVEGAÇÃO

4.4.1 Trajetória

A execução do algoritmo SLAM indica a posição com base em coordenadas x e y tendo como base o mapa bidimensional gerado. Essas coordenadas podem assim ser utilizadas de modo a navegar o AGV segundo uma trajetória pré-determinada.

O objetivo consiste em testar a capacidade da localização por meio do método SLAM para a navegação, sendo assim desenvolvido um algoritmo capaz de determinar o erro de posicionamento do AGV para trajetórias compostas por linhas retas, ou curvas semicirculares. O princípio seguido por tal algoritmo foi que os movimentos retilíneos são realizados unicamente nos eixos x ou y. Assim, o erro de posição consiste no desvio apresentado no eixo perpendicular ao de movimentação. Por exemplo, considerando a origem como partida e uma movimentação unicamente no eixo x, as coordenadas do eixo y descrevem diretamente o erro de posição em relação à referência que deve ser seguida.

Ambas as coordenadas apresentam variação para as curvas semicirculares, de modo que foi necessário o desenvolvimento de uma metodologia de cálculo capaz de determinar a dis-

tância do AGV ao ponto mais próximo na curva. Tal cálculo foi baseado na Equação (14), que determina a distância entre dois pontos no plano xy.

$$(x-a)^2 + (y-b)^2 = d^2 \tag{14}$$

onde x e y representam as coordenadas do AGV, a e b representam respectivamente as coordenadas x e y do centro da circunferência e d representa a distância até o centro circunferência, sendo assim possível determinar esta distância. Uma vez que uma linha reta entre um ponto qualquer e o centro da circunferência intercepta a periferia da mesma basta deduzir o valor do raio, sendo obtido assim o erro de posição, ou a distância entre o ponto esperado da trajetória e o efetivamente obtido.

O algoritmo foi desenvolvido para operar em trechos, ou seja, são informados pontos de origem e destino, juntamente com o tipo do movimento, linear ou curva, e o programa determina o trajeto entre os diversos pontos. Os dados consultados seguem o formato apresentado na Tabela 1, os valores possuem unidade de metro.

Tabela 1 - Tabela de pontos intermediários para planejamento de trajetória.

índice	tipo	distancia	início x	início y	fim x	fim y	raio	centro x	centro y
1	reta	2	0	0	2	0	-	-	-
2	reta	2	2	0	4	0	-	-	-
3	curva	2	4	0	4	2	1	4	1

Entretanto o valor calculado referente ao erro é realizado de forma absoluta. Assim, os sinais são avaliados separadamente. Tendo como referência o AGV, existem unicamente dois sentidos para o erro: esquerda e direita. Foi estabelecido como padrão para operação, valores positivos para erros à direita e negativos para erros à esquerda. A determinação do sinal, contudo, deve levar em consideração o sentido de deslocamento do AGV, pois o plano de referência do mapa é fixo, enquanto que o do AGV é variável. Uma vez que cada trecho fornece as informações de origem e destino, foram estabelecidas funções que avaliam o sinal que deve ser atribuído ao erro, considerando o movimento realizado e os valores das posições inicial, atual e final, esta última correspondendo ao destino pré-estabelecido.

O algoritmo de avaliação da trajetória e determinação do erro de posição é executado em um pacote ROS, o qual é capaz de receber os dados de posição do AGV avaliados pelo pacote SLAM.

4.4.2 Transmissão de dados

De forma a transmitir as correções necessárias para a manutenção da trajetória ao controlador dos motores presente no AGV, foi desenvolvido um processo análogo ao já empregado para a transmissão dos dados obtidos pelos *encoders*, porém seguindo um fluxo inverso.

Primeiramente, foi estabelecida uma conexão TCP entre o computador principal e o Raspberry Pi presente no AGV. De forma a não afetar a transmissão dos dados de odometria, foi criada uma conexão separada, sendo transmitido o valor do erro de posição acompanhado de seu respectivo sinal.

No Raspberry Pi, foi então desenvolvido um novo programa, responsável por receber os valores de erro obtidos do computador principal e então os retransmitir na rede CAN, a codificação para transmissão nesta rede foi simplificada, tratando-se de apenas 5 algarismos e um sinal. Foram, portanto, designados um algarismo para cada Byte, utilizando um total de 6 bytes. Tal codificação é ilustrada pela Figura 8 para um erro arbitrário de 1,2345 m.

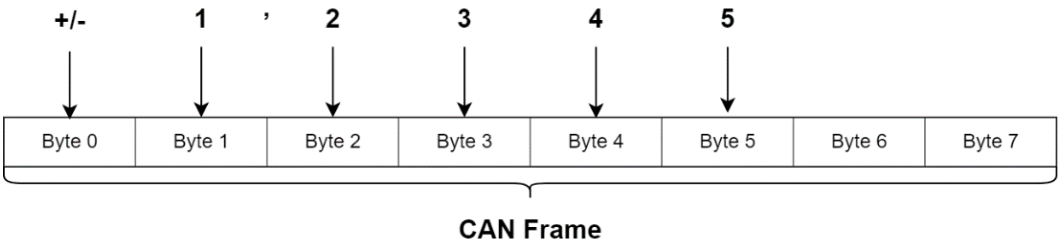


Figura 8 - Codificação para transmissão do erro na rede CAN.

Tal codificação opera com a premissa de que o erro nunca será maior que 9,9999 m pois um valor com dois algarismos antes da vírgula apresentaria uma codificação errônea. Entretanto, um erro com valor superior a 10 m representa um valor de erro irrecuperável para o algoritmo SLAM em um ambiente típico. Sendo, então, a operação restrita a erros inferiores a 10 m.

CAPITULO 5 – RESULTADOS

Este capítulo apresenta os resultados observados, tanto no que se refere ao emprego do algoritmo SLAM, quanto à navegação do AGV.

O algoritmo SLAM foi configurado de forma que cada pixel do mapa, ou cada valor da matriz de ocupação, representava uma área de 1 mm². Desse modo, valores inferiores a 1 mm não são avaliados pelo algoritmo de mapeamento.

Todas as medições de distância foram realizadas utilizando o sensor medidor de distância laser Fluke 424D, sobre uma superfície nivelada e mantendo a mesma posição entre medições. Nas condições em que foram realizados os testes, este sensor apresenta uma incerteza de medição típica de ± 1 mm e máxima de ± 2 mm segundo o fabricante. De modo que todos os valores tanto médias, como desvios padrão possuem uma incerteza associada de ± 3mm.

5.1 LOCALIZAÇÃO E MAPEAMENTO

De forma a validar primeiramente a etapa de localização do algoritmo SLAM, foi realizada a movimentação do AGV utilizando seu *joystick*, o qual foi configurado para unicamente enviar comandos para movimentos lineares para frente ou para trás, de modo que comandos acidentais do operador não resultassem em uma trajetória diagonal. Para cada trajeto foram realizadas 10 medições.

Tendo como base unicamente os valores de posição nos eixos x e y, o AGV foi movido no eixo x até uma posição de 1,008 m segundo o algoritmo SLAM, com o auxílio do medidor de distância laser foi medida uma alteração de 1,001 m entre as posições original e final do AGV, resultando em um erro de medição de 7 mm.

Medições subsequentes para um mesmo deslocamento de 1 m apresentaram erros de medição entre a posição indicada pelo algoritmo e o sensor, variando entre 5 e 62 mm, sendo que o valor médio calculado entre a totalidade de medições foi de 21 mm (σ = 17,707 mm). Avaliações para uma distância de 3,00 m entre as posições iniciais e finais apresentaram erros de medição entre 11 e 87 mm, apresentando um valor médio de 28 mm (σ = 22,925 mm).

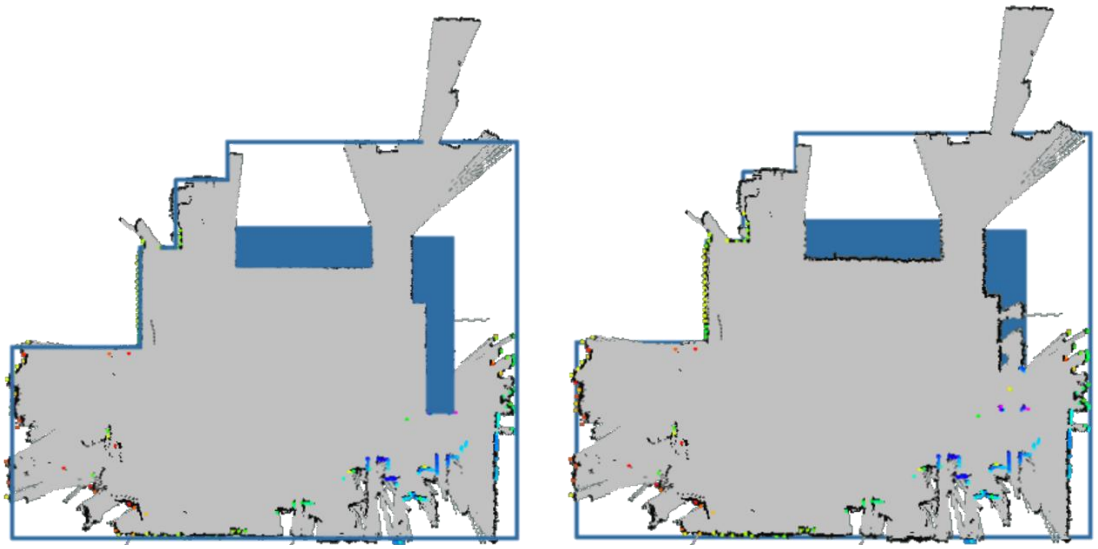
A determinação da variação entre as posições virtuais e reais para trajetórias curvas foi realizada movimentando o AGV, em uma trajetória arbitrária, do ponto de origem até o ponto (1,500 ; 1,500), ou seja, um deslocamento de 1,5 m tanto no eixo x como no eixo y. O deslocamento foi calculado empregando o deslocamento linear diagonal entre a origem e o destino para o valor virtual. O medidor de distância, por sua vez, foi empregado de modo a medir diretamente o deslocamento diagonal, sendo demarcado um ponto sobre o eixo central do AGV como "alvo" de forma a minimizar variações de medida devido à alteração na pose do AGV.

Os erros de medição obtidos para o deslocamento em ambos eixos cartesianos apresentaram valores variando entre 64 mm e 253 mm com um valor médio de 116 mm (σ = 60,225 mm).

Simultaneamente à avaliação da posição do AGV, é realizada a construção do mapa virtual do ambiente. De forma a estabelecer a correspondência entre a representação virtual e o ambiente físico, foram realizadas comparações entre as dimensões de paredes e objetos imóveis presentes no local.

Inicialmente, foi obtida a planta do local de forma a empregar as dimensões presentes nesta como referência. Entretanto, após a verificação através de medidas, foram observadas variações de até 370 mm entre o apresentado no projeto e o determinado. Foram, então, medidas as dimensões do ambiente, assim como as dimensões das bancadas fixas presentes no local. Sobreposições dos mapas obtidos por medição e pelo algoritmo SLAM são apresentadas na Figura 9.

Por se tratar de um ambiente ocupado por uma série de obstáculos, é notável a quantidade de posições cuja "visão" do sensor estava bloqueada. Tendo como referência principal o setor noroeste do mapa, foram selecionadas as dimensões das paredes ali presentes para comparação. Adicionalmente, as bancadas presentes na região central do local foram usadas para comparação.



(a) (b)

Figura 9 - Sobreposição dos mapas: (a) o "mapa medido" em primeiro plano; (b) "mapa SLAM" em primeiro plano.

O *software* Rviz permite que sejam realizadas medidas diretamente no mapa virtual entre dois pontos selecionados, sendo assim empregado para a avaliação das dimensões geradas pelo algoritmo SLAM. As discrepâncias entre as dimensões apresentaram valores variando entre 6 mm e 51 mm, resultando como valor médio 22 mm (σ = 14,987 mm). A similaridade entre os erros de medição observados para os deslocamentos lineares de 1 m, 3 m e nas medições de elementos do mapa sugere a existência de uma fonte de erro sistemático.

5.2 NAVEGAÇÃO

A etapa de navegação envolveu o estabelecimento em *software* de uma trajetória a ser seguida pelo AGV com uma velocidade de 0,4 m/s. Inicialmente, o trajeto definido consistiu de uma reta de comprimento 3 m, sendo que neste cenário o algoritmo de navegação realizaria o monitoramento da posição do AGV e assim alteraria a velocidade dos motores buscando manter o AGV em seu trajeto. A Figura 10 apresenta o a trajetória executada pelo AGV, avaliada pelo algoritmo SLAM tendo como destino a posição de 3 m no eixo x e 0 m no eixo y. Pode ser observado nesta ocasião um desvio máximo de aproximadamente 90 mm.

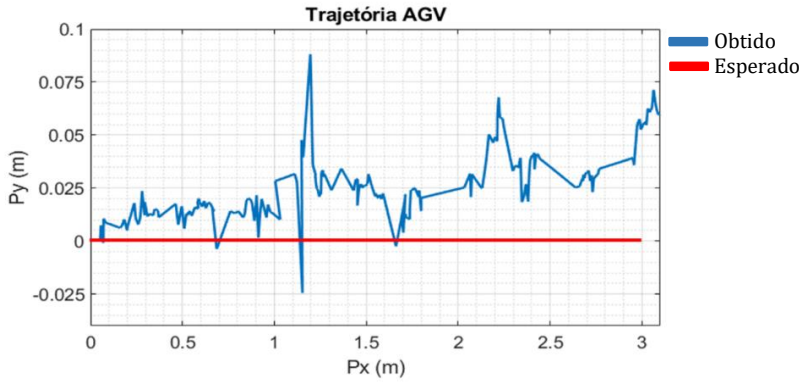


Figura 10 - Trajetória para um deslocamento de 3 m.

O algoritmo de navegação foi, todavia, configurado para não reagir a desvios menores que 5 cm devido a travamentos no algoritmo SLAM que ocorriam a cada atualização realizada no mapa. Individualmente, o algoritmo de localização possui uma alta taxa de atualização (aproximadamente 15 Hz). O mapeamento, por sua vez, ocorre a uma taxa aproximada de 0,3 Hz. No entanto, o processamento da atualização do mapa após cada ciclo de mapeamento causa a interrupção da localização por um período que varia de 0,5 s até 1,2 s dependendo da diferença entre o mapa existente e a observação mais recente do sensor laser. Por esse motivo, a tolerância de 50 mm teve de ser introduzida ao algoritmo de navegação visando a reduzir a frequência das comandos enviados aos motores.

Foi constatado que a posição de parada do AGV através do uso exclusivo do algoritmo de navegação apresentou erros acima dos níveis desejados. No teste apresentado na Figura 10, um deslocamento linear definido de 3,0 m resultou em um deslocamento final de 2,642 m no eixo x e 0,531 m no eixo y, sendo o ponto final da trajetória definido como 3 m em x e 0 m em y. Testes subsequentes apresentaram resultados similares, e nem mesmo a redução de velocidade do veículo para 0,2 m/s foi capaz de reduzir de forma significativa os erros de posicionamento observados.

Um segundo circuito contemplando trajetos curvos foi desenvolvido, possuindo 4 etapas. Inicialmente, o AGV realizaria um movimento linear de 3 m seguido por uma curva semicircular de 3 m de diâmetro, retornando então no sentido oposto para a posição de origem percorrendo 3 m linearmente e por fim realizando a curva final em um semicírculo de 3 m de diâmetro. Assim constituindo, um típico trajeto em formato oval. O registro de posição realizado pelo algoritmo SLAM é apresentado na Figura 11.

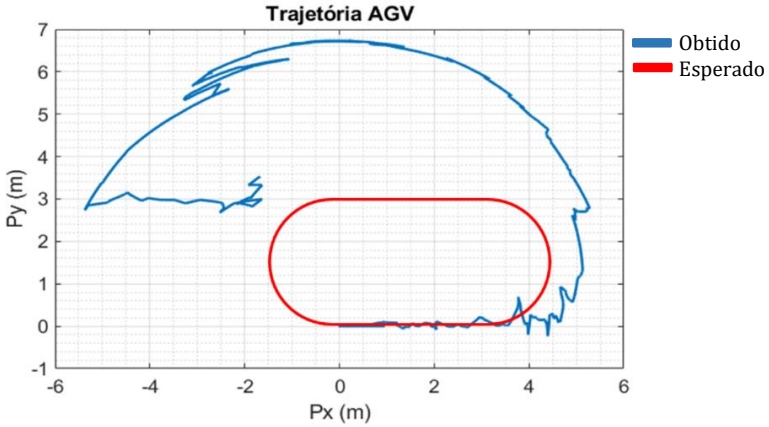


Figura 11 - Trajetória para um deslocamento em curva.

O registro de posição apresenta claramente a incapacidade do algoritmo de navegação em manter sua trajetória uma vez que o movimento angular é realizado, resultando em um desvio de aproximadamente 3 em seu eixo *y* e 2 m em *x*. A medição da posição final do AGV mostrou um deslocamento de 4,500 m no eixo y e 3,1 m no eixo x. Novamente, verificou-se um grande desvio entre o obtido pelo algoritmo e a movimentação real. Pode ser constatado com clareza ao longo da trajetória em curva a presença de um efeito de sub-amostragem decorrente das pausas no algoritmo de localização decorrentes da atualização do mapa; durante as pausas do algoritmos de localização os motores continuaram funcionando com o comando anterior, exigindo correções mais drásticas de trajetória, após a atualização do mapa e resultando em um trajeto bastante irregular.

Apesar da aparente contradição entre a precisão apresentada inicialmente e a observada após a operação autônoma, tais discrepâncias podem ser atribuídas diretamente ao modo de operação. A navegação operada por *joystick* demandava uma quantidade muito menor de correções de modo que o ambiente observado pelo sensor possuía sempre grande similaridade em comparação com o mapa existente, além de serem realizadas pausas em situações de processamento prolongado. A navegação livre, especialmente para a realização de curvas, gerava constantes correções, resultando em variações de observação de ambiente que demandavam elevado processamento, introduzindo atrasos no processamento da etapa de localização, degradando assim o desempenho do algoritmo.

CAPITULO 6 – CONCLUSÕES

Através do trabalho realizado foi possível implementar com sucesso o algoritmo SLAM no veículo em uma condição controlada, executando as etapas de localização e mapeamento simultaneamente. Esta condição controlada consistiu no veículo sendo guiado em baixa velocidade por um operador com o auxílio de um *joystick*.

Os diversos programas desenvolvidos buscando integrar o algoritmo SLAM ao veículo operaram de acordo com o esperado. O principal problema observado foi a capacidade limitada de processamento do computador onde a rotina SLAM (*Gmapping*) foi executada.

O algoritmo de navegação, por sua vez, utilizava-se do controlador presente no AGV, projetado para operação com sensores de elevada taxa de atualização (> 60 Hz), podendo, assim, operar utilizando comandos repentinos de alto valor. Entretanto, por estar sujeito à subamostragem causada pela atualização do mapa, este controle demandou um ajuste para operação suave e, por consequência, não foi capaz de recuperar-se de erros demasiadamente grandes.

A rotina SLAM (*Gmapping)* não foi desenvolvida visando a dividir seu processamento em múltiplos núcleos, estando, assim, limitado à velocidade de 2 GHz de um único núcleo do computador, resultando numa demanda constante de 100% de sua capacidade, de modo que um desenvolvimento tendo como foco o processamento em múltiplos núcleos ou até processamento gráfico através da GPU (*Graphics Processing Unit*), possa eliminar o gargalo causado devido ao computador.

Tendo como principal foco o desenvolvimento de um *software* capaz de integrar o algoritmo SLAM a um AGV comercial genérico, foi demandada atenção a aspectos de comunicação, compatibilidade de *hardware* e integração de sistemas. A etapa de navegação, ainda que

incapaz de apresentar um desempenho compatível com o de um produto comercial, apresentou potencial para melhorias, uma vez que seu comportamento é próximo ao esperado em situações restritas, como trajetos curtos e retilíneos, sendo, assim, uma prova de conceito considerada satisfatória.

6.1 TRABALHOS FUTUROS

As propostas de aprimoramento, se concentram primordialmente na etapa de navegação sendo a etapa de mapeamento o maior contribuinte para sua falha no seguimento de um trajeto determinado. O presente trabalho empregou o pacote *gmapping*®, o qual não possui foco em navegação, tendo sido desenvolvido com foco unicamente no algoritmo SLAM.

De forma a tornar viável a navegação, esta deve ser precedida pelo processo de mapeamento, sendo realizado assim um "treinamento" prévio. Uma análise do código permitiu determinar que o programa original pode ser adaptado de forma a realizar o mapeamento e, em seguida, armazená-lo, operando subsequentemente apenas a localização, eliminando, deste modo, os gargalos de processamento ocasionados pela atualização do mapa.

Uma alternativa à interrupção da etapa de atualização do mapa é o aumento da área representada por cada pixel do mapa, considerando os erros de posição detectados, pode-se determinar que a utilização de cada pixel do mapa com lado de valor 10 mm reduziria significativamente o custo computacional para a etapa de mapeamento, possivelmente permitindo assim a execução da navegação simultaneamente.

Sugere-se também uma abordagem "híbrida" ao sistema SLAM, sendo integrados marcadores ao longo do trajeto, como etiquetas RFID, que serviriam como pontos de checagem nos quais qualquer desvio entre a posição obtida pelo algoritmo SLAM e a real transmitida pelos marcadores possa ser corrigida, reduzindo a incerteza ao longo do trajeto.

É sugerida que a implementação ROS faça uso do pacote *ros-control*, visando operar os diversos algoritmos em camadas distintas, separando *software* e *hardware* buscando uma operação que mais se aproxime de "tempo real".

REFERÊNCIAS BIBLIOGRÁFICAS

ANNAIYAN, A.; OLIVARES-MENDEZ, M. A.; VOOS, H. Real-time graph-based SLAM in unknown environments using a small UAV In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 1. 2017, Miami, FL, EUA. **2017 International Conference on Unmanned Aircraft Systems (ICUAS)**. Piscataway, NJ, EUA: IEEE Press, 2017. 1118 – 1123.

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous Localization and Mapping: Part II. **IEEE Robotics & Automation Magazine**, Piscataway, v.13, n.3, p. 108-117, ago. 2006.

CADENA, C. *et al.* Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. **IEEE Transactions on Robotics**, Piscataway, v.32, n.6, p. 1309-1332, dez. 2016.

CHONG, T. J. *et al*. Sensor Technologies and Simultaneous Localization and Mapping (SLAM). In: 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015), 1. 2015, Langkawi, Malasia. **IRIS 2015 : 2015 IEEE International Symposium on Robotics and Intelligent Sensors**. Piscataway, NJ, EUA: IEEE Press, 2015. 174 – 179.

DISSANAYAKE, M. W. M. G. *et al*. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. **IEEE Transactions on Robotics and Automation**, Piscataway, v.17, n.3, p. 229-241, jun. 2001.

DOUCET, A. *et al*. Rao-Blackwellized Particle Filtering for Dynamic Bayesian Networks. In: 16th Conference on Uncertainty in Artificial Intelligence, 16. 2000, San Francisco, CA, EUA. **Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence**. San Francisco, CA, EUA: Morgan Kaufmann Publishers Inc., 2000. 176 – 183.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous Localization and Mapping: Part I. **IEEE Robotics & Automation Magazine**, Piscataway, v.12, n.2, p. 99-110, jun. 2006.

FLUKE. **Fluke 424D, 419D and 414D Laser Distance Meters Technical Data**. Disponível em: <http://media.fluke.com/documents/3028711_0000_ENG_E_W.PDF?_ga=2.184839209.1088387187.1528245917-1108942704.1528245917>, Acesso em: 15 jun. 2018.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In: 2005 IEEE International Conference on Robotics and Automation, 1. 2005, Barcelona, Espanha. **Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International**

Conference on Robotics and Automation. Piscataway, NJ, EUA: IEEE Press, 2005. 2432 – 2437.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. **IEEE Transactions on Robotics**, Piscataway, v.23, n.1, p. 34-46, fev. 2007.

GRISETTI, G. *et al*. A Tutorial on Graph-Based SLAM. **IEEE Intelligent Transportation Systems Magazine**, Piscataway, v.2, n.4, p. 31-43, dez – mar 2010.

GROOVER, M. P. **Automation, Production Systems, and Computer-Integrated Manufatcturing.** New Jersey: Pearson Prentice-Hall, 2008.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Simultaneous Map Building and Localization for an Autonomous Mobile Robot In: IEEE/RSJ International Workshop on Intelligent Robots and Systems '91, 1, 1991, Osaka, Japão. **Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91**. Piscataway, NJ, EUA: IEEE Press, 1991. 1442 – 1447.

MARQUES, D. M. **Métodos Estocásticos Aplicados a Definição de Estratégias de Amostragem e Homogeneização.** 2014. 140 f . Tese (Doutorado em Engenharia, Área de concentração: Metalurgia Extrativa e Tecnologia Mineral) – Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e de Materiais (PPGE3M), Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2014.

MEGALINGAM, R. K. *et al*. SLAM – ROS based Autonomous Indoor Navigation Simulation Using SLAM Algorithm. **International Journal of Pure and Applied Mathematics**. v.118, n.7, p. 199-205. 2018.

MONTEMERLO, M. *et al*. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: 18th National Conference on Artificial Intelligence, 18. 2002, Edmonton, Alberta, Canada. **Proceedings of the 18th National Conference on Artificial Intelligence**. Menlo Park, CA, EUA: American Association for Artificial Intelligence, 2002. 593 – 598.

NEULAND, R. C. **Uma Hibridização do Método de Monte Carlo com Técnicas Intervalares para o Problema de Localização Global.** 2014. 100 f . Dissertação (mestrado) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, 2014.

NIETO, J. *et al*. Real Time Data Association for FastSLAM. In: 2003 IEEE International Conference on Robotics and Automation, 1, 2003, Taipei, Taiwan. **Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on Robotics and Automation**. Piscataway, NJ, EUA: IEEE Press, 2003. 412 – 418.

OLSON, E. B. Real-Time Correlative Scan Matching. In: 2009 IEEE International Conference on Robotics and Automation, 1. 2009, Kobe, Japão. **Robotics and Automation, 2009. ICRA '09. IEEE International Conference on Robotics and Automation**. Piscataway, NJ, EUA: IEEE Press, 2009. 4387 – 4393.

OPENSLAM. **GMapping.** Disponível em: <http://openslam.org/gmapping.html>, Acesso em: 10 abr. 2018.

PAJAZITI, A.; AVDULLAHU, P. SLAM – Map Building and Navigation via ROS. **International Journal of Intelligent Systems abs Applications in Engineering**. v.2, n.4, p. 71-75, dez. 2014.

PEI, F.; WU, M.; ZHANG, S. Distributed SLAM Using Improved Particle Filter for Mobile Robot Localization. **The Scientific World Journal**. v. 2014, n.1, p. 10, abr. 2014.

RASPBERRY PI. **Raspberry Pi 3 Model B+ Product Brief**. Disponível em: <https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf>, Acesso em: 8 abr. 2018.

RIISGAARD, S.; BLAS, M. R. **SLAM for Dummies**. 2005. Disponível em: < https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf>, Acesso em: 28 mar. 2018.

ROS.ORG. **ROS Instroduction.** Disponível em: <http://wiki.ros.org/ROS/Introduction>, Acesso em: 10 abr. 2018.

SACCHETIN, M. C. **Análise e implementação de algoritmos para localização e mapeamento de robôs móveis baseada em computação reconfigurável.** 2005. 120 f . Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC, Universidade de São Paulo, São Carlos, SP, 2005.

SCHULZE, L.; BEHLING, S.; BUHRS, S. Automated Guided Vehicle Systems: a Driver for Increased Business Performance. In: International MultiConference of Engineers and Computer Scientists 2008, 2. 2008, Hong Kong. **Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol II**. Newswood Limited, Hong Kong, 2008. 1275 – 1280.

SEEEDSTUDIO. **CAN-BUS Shield V1.2 Schematics**. Disponível em: <https://github.com/SeeedDocument/CAN_BUS_Shield/raw/master/resource/CAN-BUS_Shield_v1.2.pdf>, Acesso em: 8 abr. 2018.

SICK. **TiM55x/TiM56x/TiM57x Ranging Laser Scanner Mounting and Electrical Installation**. Disponível em: <https://sick-virginia.data.continum.net/media/docs/3/33/133/Technical _information_TiM55x_TiM56x_TiM57x_Ranging_Laser_Scanner_en_IM0053133.PDF>, Acesso em: 6 abr. 2018.

SICK. **Telegram Listing Ranging sensors LMS1xx, LMS5xx, TiM5xx, MRS1000, MRS6000, NAV310, LD-OEM15xx, LD-LRS36xx**. Disponível em: <https://sick-virginia.data.continum.net/media/docs/7/27/927/Technical_information_Telegram_Listing _Ranging_sensors_LMS1xx_LMS5xx_TiM5xx_MRS1000_MRS6000_NAV310_LD_OEM15xx_LD_LRS36xx_en_IM0045927.PDF>, Acesso em: 6 abr. 2018.

SKRZYPCZYŃSKI, P. Simultaneous localization and mapping: A feature-based probabilistic approach. **International Journal of Applied Mathematics and Computer Science - Special Section: Robot Control Theory** Walter de Gruyter & Co. Hawthorne, NJ, EUA, v.19, n.4, p. 575-588, dez. 2009.

STACHNISS, C.; BURGARD W. Particle Filters for Robot Navigation. **Foundations and Trends® in Robotics**, Hanover, v.3, n.4, p. 211-282, 2012.

STACHNISS, C. **EKF SLAM**. Robot Mapping Lectures. University of Freiburg, Freiburg, 2013.

THRUN, S.; BURGARD, B.; FOX, D. **Probabilistic Robotics**. Cambridge: MIT Press, 2005.

XIANG, L. *et al*. Robust graph SLAM in dynamic environments with moving landmarks. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1. 2015, Hamburgo, Alemanha. **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, EUA: IEEE Press, 2015. 2543 – 2549.

YATIM, N. M.; BUNIYAMIN, N. Particle Filter In Simultaneous Localization And Mapping (SLAM) Using Differential Drive Mobile Robot. **Jurnal Teknologi (Sciences & Engineering)** 77. 10.11113/jt.v77.6557, 2015.