SLAM baseado em Scan-Matching com Otimização por Enxame de Partículas

Pedro Jorge de Albuquerque de Oliveira[1], Nadia Nedjah[1], Luiza de Macedo Mourelle[2]

[1] Departamento de Engenharia Eletrônica e Telecomunicações, Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Brasil
[2] Departamento de Engenharia Eletrônica e Telecomunicações, Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Brasil
pejalb@gmail.com, (nadia | ldmm)@eng.uerj.br

Abstract. This paper presents a method for solving the problem of Simultaneous Localization and Mapping (SLAM) based on the application of Particle Swarm Optimization (PSO) to the estimation of occupancy grid maps from laser scan data. The registration task is cast in the form of a nonlinear optimization problem and used to get an adequate approximation of the relative motion of the robot in between data frames. The method presented has the advantage of requiring no landmarks and being fast enough to perform online. Furthermore, we elaborate on how to tune the parameters of the algorithm to obtain satisfactory results.

1 Introdução

O problema conhecido como SLAM (*Simultaneous Localization And Mapping*) lida com as tarefas geminadas de estimação da trajetória de um agente móvel e do mapa do ambiente por onde ele se locomove. Essa área de pesquisa recebeu grande atenção nas ultimas décadas por parte de pesquisadores nas áreas de robótica, inteligência artificial, visão computacional e correlatas. Ao longo dos anos, uma miríade de formulações para o problema foram propostas. Os motivos por trás dessa abundância de de formulações é em grande parte devida ao desejo de cada pesquisador de aplicar determinadas técnicas visando a maximização da performance de dada solução, em seu nicho particular de aplicação, ou talvez em carácter exploratório permitindo a análise do comportamento específico de determinados métodos. Nesse trabalho, propomos um método de resolução para o SLAM baseado num procedimento de *scan matching* calcado na meta-heurística de otimização conhecida como *Particle Swarming Optimization* (otimização por enxame de partículas).

No caso do SLAM o mapa assume o papel de modelo do ambiente, e sua estimação está conjugada à inferência da trajetória do autômato no ambiente que o circunda. O mapa pode apresentar-se de multiplas formas, tais como mapas de marcadores (*feature maps*), mapas de ocupação (*Occupancy Grid Maps*, nuvens de pontos (*Point Clouds*) ou formulações topológicas [1, 2] Independentemente

de sua forma, a construção de um mapa coerente gera a necessidade de uma ampla coleta de dados acerca do ambiente que se deseja modelar. Com o propósito de analisar as relações existentes entre as informações disponíveis, cada conjunto de medidas[3] pode ser considerado como uma entidade pertencente a um *data frame*, em que cada *data frame* pode ser, por exemplo, um conjunto de distâncias obtidas por meio de um *Laser Range Finder*, uma varredura de sonar, um grupo de fotografias obtidas do mesmo ponto de vista, ou qualquer outro tipo de dado que o sensor forneça. A aquisição desses *data frame*, entretanto, leva a outro tipo de desafio, haja vista que os dados coletados estão sempre expressos com relação ao sistema de coordenadas local de uma entidade que se encontra em movimento e cuja posição posição é, *a priori*, desconhecida. No caso dos dados oriundos do sensor a laser, como o que foi utilizado no presente trabalho, obtém-se, a cada varredura um conjunto de distâncias aos obstáculos mais próximos presentes em um setor angular em torno do robô. Cabe ao *back end* do sistema de SLAM pegar essa informação e da maneira mais eficiente e precisa possível retornar um *data frame* que permita atualizar o modelo de mundo, *i.e.* o mapa e a trajetória. Da diferença dos sistemas de referência dos dados obtidos advém a necessidade de se utilizar um sistema capaz de obter o alinhamento deles [3]. O processo de alinhamento de duas imagens, obtidas de duas imagens, obtidas em instantes diferentes ou de pontos de vista diferentes, que representem o mesmo objeto ou lugar, é conhecido como *image registration*. Os processos de *image registration* e o de *scan registration*, que doravante serão referidos por registro de imagem e registro de varredura, respectivamente, são uma peça fundamental em diversos dos métodos desenvolvidos para a resolução de várias tarefas como o SLAM, *Structure From Motion*, reconhecimento de objetos, fusão de imagens e modelagem tridimensional. No contexto do SLAM, procedimentos dessa sorte são empregados objetivando recuperar o movimento relativo entre dois *data frames*. No caso específico do uso de dados provenientes de LRFs, o problema de alinhamento de *data frames* é usualmente denominado *scan alignment* ou *scan matching*.

O alinhamento de varreduras apresenta, por conseguinte, uma enorme variação no que tange tanto a quantidade quanto a forma da informação utilizada para determinar a correspondência necessária à determinação do estado do sistema, como um todo, tal qual especificado anteriormente. Dissertemos então sobre os impactos de escolher realizar uma correspondência varredura a mapa ao invés de uma do entre varreduras, tendo essas duas formas sido investigadas na elaboração do presente artigo. Dire-mo-vos que quanto maior a "quantidade de informações" utilizada para estimar a transformação supracitada, maior será o esforço computacional necessário mas, em contra partida, maior será a possibilidade de alcançar um resultado mais preciso. Nesse ponto, que fique claro que nos referimos à informação diretamente processada para que se faça a inferência

[3] *N.B.*: Por conjunto de medidas referi-mo-nos a uma agregação de informações obtidas numa única varredura. Considera-se que o sensor se encontra parado em cada varredura,e os dados desse modo obtido formam, portanto, um conjunto do ponto de vista da temporalidade.

pretendida, *i.e.* a quantidade de informação embutida em algum modelo que, porventura se utilize, desenvolvido, ou treinado, *a priori*, não influencia, evidentemente, o esforço computacional de modo direto.

2 Trabalhos Relacionados

Em [4], que é a referência mais frequentemente citada sobre o algoritmo conhecido como *Iterative Closest Point* (ICP), os autores apresentam diversas fórmulas para o cálculo da distância entre diversas representações geométricas dos dados sendo analisados e prosseguem mostrando de que modo tais medidas de distância podem ser utilizadas dentro do contexto de registro de varreduras pelo uso do ICP. Ainda no mesmo trabalho encontra-se proposta uma variante denominada *Accelerated ICP*(ICP Acelerado) pelos autores que realizam ainda uma análise da convergência de ambos os algoritmos.

Em [3], os autores mostram um extenso trabalho sobre os temas de alinhamento consistente de varreduras a laser e registro(*Registration*) por meio do emprego de grafos de poses e, portanto, fazendo uso do conceito de *Pose Graph Optimization*(Otimização de Grafos de Poses). Os autores afirmam que, por meio da manutenção de um histórico completo de todos os *data frames* adquiridos durante o movimento realizado pelo agente que realiza o mapeamento é possível utilizar a informação que se contida nesse conjunto de dados para construir uma rede de relações entre os conjuntos de dados mensurados e, combinando informações odométricas e restrições obtidas por meio do alinhamento de varreduras, um mapa do ambiente onde esse movimento se sucede. O procedimento de construção do mapa é possibilitado pela hipótese de que a trajetória mais provável é aquela que dá origem a uma rede com o que os autores chamam de *"network energy"*(energia da rede).

Em [5], os autores propõem uma solução para o problema de SLAM por meio do alinhamento de varreduras realizado pelo emprego de algoritmos genéticos. No trabalho deles, os autores aplicaram uma técnica de alinhamento entre os dado de uma varredura a laser e o mapa global de ocupação. Ademais, encontra-se nesse trabalho proposta uma implementação do procedimento baseado em um FPGA (Field Programmable Gate Array), cuja performance os autores comparam à uma solução puramente de software rodando num computador de propósito geral. Os resultados apresentados foram promissores, mas, em geral, na análise dos propositores do método, há duas fraquezas fundamentais na abordagem por eles descrita: a acumulação de erros rotacionais e a dificuldade de determinar de modo correto a transformação rígida pela qual o robô passou se o movimento se der num ambiente sem características visuais salientes.

3 Mapas de Ocupação

Os mapas de ocupação, conhecidos como *occupancy grid maps* na literatura angla, são uma forma de representação espacial baseada na discretização do espaço e na classificação de cada "célula" como *cheia* ou *vazia*. Esse tipo de

representação espacial tem como hipóteses a estaticidade do mundo, a independência entre as variáveis aleatórias que representam o estado de ocupação das células do mapa e o fato de cada célula estar completamente cheia ou vazia. A distribuição de probabilidade do mapa é dada pelo produto das probabilidades de todas as suas células [6]. Para a presente discussão adotaremos a convenção usada em [6] e repetida a seguir. Sejam t o instante atual m o mapa, m_i uma célula do mapa no instante atual, x_t a pose atual e z_t a observação feita no instante atual. É conveniente representar as probabilidades de ocupação no formato de *log-odds*. O cálculo do valor nesse formato pode ser obtido a partir de $p(m_i|z_{1:t}, x_{1:t})$ por meio da Equação 1,

$$l_{t,i} = \log \frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}. \tag{1}$$

Vale ressaltar que com o uso da forma de *log-odds* há tanto um ganho de performance no processo de atualização do mapa, já que são trocados os produtos de valores em ponto flutuante pela adição de seus correspondentes num espaço de logaritmos, quanto um ganho de estabilidade numérica em casos cujos valores de probabilidade envolvidos estejam próximos de 0 ou de 1. A obtenção dos valores de ocupação na forma de probabilidades a partir dos valores na forma de *log-odds* pode ser feita pelo uso da Equação 2,

$$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp l_{t,i}}. \tag{2}$$

O Algoritmo 1 indica de que modo é feito o mapeamento nesse tipo de representação e segue [6].

Algorithm 1 mapa de ocupação

entrada $l_{t,i}, x_t, z_t$
  para todas as células m_i **faça**
    se m_i está no campo de visão de z_t **então**
      $l_{t,i} = l_{t-1,i} + modeloInversoSensor(m_i, x_t, z_t) - l_0$
    senão
      $l_{t,i} = l_{t-1,i}$
    fim se
  fim para
  retorna $l_{t,i}$

Em que l_0 representa a probabilidade de ocupação *a priori* sob a forma de *log-odds* conforme definido na Equação 1. O modelo inverso de sensor[4] implementa relação dada pela Equação 3

$$modeloInversoSensor(m_i, x_t, z_t) = p(m_i|z_t, x_t). \tag{3}$$

[4] O modelo de sensor é denominado inverso pois fornece a probabilidade das causas condicionada em relação os efeitos

A importância de considerar um modelo de sensor adequado está no fato de que sensores, por maior que seja a sua precisão, geram medições espúrias em certas circunstâncias. A título de exemplo, poderíamos mencionar as reflexões de raios laser em superfícies vítreas, que pode levar à detecção de obstáculos inexistentes. Além de reflexões, medidas realizadas com distancias próximas do maior afastamento que um LRF (Laser Range Finder) é capaz de fazer tendem a incluir erros, já que o sensor detectará a fronteira da sua região de alcance como um obstáculo.

Ess

4 Otimização de Enxame

A inteligência de enxame é um campo da inteligência artificial que se inspira no comportamento coletivo de seres como formigas, abelhas, vermes, revoadas de pássaros, e cardumes. Muito embora os indivíduos sejam, usualmente, pouco sofisticados em seu *modus operandi*, a aparente coordenação de seus comportamentos individuais faz surgir um complexo padrão de interação entre o grupo e o ambiente em que ele se encontra. O processo de busca é, em verdade, uma interação e dicotomia entre as etapas conhecidas como *exploration* e *exploitation*. Todo e qualquer algoritmo que se baseie em alguma sorte de população, ou enxame (conforme seja a nomenclatura adequada), depende diretamente do fluxo de informação nessa população para seu bom funcionamento. O fluxo de informação é, em diversos casos modulado como uma das características dos algoritmos. Sobre isso, e do mais alto interesse para a aplicabilidade de um algoritmo, diremos que a "localidade", isso é a formação de pequenos grupos que se comunicam mais intimamente entre si do que com o restante do bando, fomenta a exploração simultânea de diversos ótimos locais pelo aumento da diversidade das fontes de informação do bando.

Uma característica fundamental de qualquer problema de otimização é o tipo de função objetivo cujos valores ótimos (sob alguma métrica de "optimalidade"[5]) se deseja conhecer. Em particular, é de grande valia saber *a priori* se a função possui um único valor ótimo ou vários. À primeira classe de funções, denomina-se unimodais e à segunda multimodais. Além disso, há a possibilidade de funções objetivo contendo pontos de sela, descontinuidades e outras anomalias congêneres que possam porventura apresentar obstáculos ao progresso de algumas técnicas de otimização.

Um algoritmo dos algoritmos mais conhecidos e amplamente utilizados dentre esses é o PSO(*Particle Swarm Optimization*) ou otimização por enxame de partículas que mantém um conjunto de entes denominados *partículas*, cada qual representando uma solução codificada em potencial. Sob esse aspecto a noção de enxame possui nos algoritmos de inteligência coletiva (ou inteligência de enxame)[6] um papel análogo ao desempenhado pela *população* em sistemas que se

[5] Empregamos o termo optimalidade como tradução direta do termo *optimality* de origem anglo-saxônica

[6] Campo denotado *Swarm Intelligence* pelos autores de língua inglesa

valem de computação evolucionária. A ideia central sendo a de que a inter-relação entre a experiência individual de cada partícula interaja com a experiência coletiva, direcionando, dessa maneira, a busca pelo espaço de soluções.

O processo de busca ocorre iterativamente e cada partícula, ao longo de sua trajetória pelo espaço de buscas, mantém um registro da melhor posição encontrada até a presente iteração durante sua história. Deveras, a preservação desse tipo de "memória" de cada indivíduo é uma das grandes distinções entre o PSO e algoritmos de computação evolutiva, por exemplo. Na literatura sobre o assunto, encontramos essa interação entre indivíduo e bando quantificada, e descrita, em termos da *variação da velocidade* dos membros do enxame. A velocidade de cada indivíduo é recalculada, a cada iteração, como uma composição de três componentes distintas: uma componente social, uma componente cognitiva[7] e uma componente de inércia. A chamada *componente inercial* (ou componente de inercia) exerce um papel análogo à componente resultando da força homônima na mecânica Newtoniana, *i.e.* opõe-se à alteração da tendência de movimento. Essa componente introduz, portanto um efeito que se opõe a mudanças drásticas na trajetória da partícula, levando a uma preservação do presente curso pelo espaço de buscas. Denomina-se *componente cognitiva* a parcela da velocidade de cada partícula devida à diferença entre a atual posição e a melhor já encontrada. A componente social tem por objetivo causar a busca por novas soluções em potencial nas cercanias das soluções "boas" já encontradas, tornando essas polos de atração para as partículas. Por fim, a chamada *componente social* é responsável por direcionar os indivíduos do enxame para as melhores posições encontradas de que eles tenham conhecimento, quer tenham essas posições sido descobertas por eles próprios ou por um vizinho com o qual haja comunicação desobstruída. Vale ressaltar nesse ponto que a distinção entre os *tamanhos* e *topologias* das vizinhanças empregadas pelo sistema de otimização possuem um efeito especialmente exacerbado sobre a componente social da velocidade. Com efeito, em [7] encontramos referência a modelos simplificados do PSO canônico em que são suprimidas certas componentes da sua velocidade. A esses modelos o autor denomina *Cognition Only Model* (modelo puramente cognitivo), *Social Only Model* (modelo puramente social) e *Selfless Model* (modelo altruísta). Esses modelos simplificados são instrutivos no sentido de que ilustram de modo inequívoco até que ponto a alteração dos três parâmetros até aqui mencionados, qual sejam: o coeficiente de inércia, o coeficiente cognitivo e o coeficiente social; afeta o comportamento do enxame e o desempenho do algoritmo de modo geral. De acordo com [8], o modelo puramente cognitivo é mais suscetível a falhas que o modelo completo. Ainda de acordo com os autores de [8], esse tipo de modelo de velocidade leva a um comportamento de intensa busca próxima das regiões inicialmente populadas, degenerando-se o PSO para um caso de busca aleatória em paralelo. Segundo [9, 7] o modelo puramente cognitivo apresenta ainda uma menor velocidade de convergência, como confirmado por [10]. O chamado mod-

[7] Kennedy et al. também chamaram a componente cognitiva de "nostalgia", fazendo alusão à tendência dos indivíduos de querer retornar a situações psicologicamente confortáveis.

elo puramente social, foi laudado em [8, 10, 7] por ser mais veloz e mais eficiente que os modelos completo e o modelo puramente cognitivo. Esse modelo representa a tendência das partículas de "imitar" seus vizinhos mais bem sucedidos. Por fim, Kennedy menciona o modelo altruísta. O modelo altruísta nada mais é que o modelo social sob a restrição de que o melhor indivíduo de uma vizinhança não possa ser o próprio indivíduo, isto é quando um membro de uma vizinhança procura quem é o melhor indivíduo nas cercanias ele se abstém da competição. Esse modelo foi descrito como sendo mais rápido que o puramente social para alguns problemas em [9], embora tenham sido reportados resultados insatisfatórios para o caso de otimização em ambientes dinâmicos em [10].

5 Alinhamento de Varreduras com PSO

O sistema que propomos utiliza uma versão adaptada da Otimização por Enxame de Partículas para realizar o passo de localização envolvido na tarefa de SLAM. A localização é realizada por meio de um mecanismo de alinhamento de varreduras que busca uma transformação afim da *pose* da entidade que se move para realizar o mapeamento. A transformação deve ser tal que leve à máxima superposição entre os obstáculos observados pelo sensor e aqueles descritos em alguma região do mapa. No Algoritmo 2, se encontra uma de descrição mais detalhada do método proposto. No algoritmo 2, cada partícula p_i está associada a uma posição no espaço de buscas de poses que compreende as duas componentes translacionais do plano e uma componente rotacional, associa-se também a cada partícula uma velocidade v_i. Já que cada partícula deve codificar em sua posição uma roto-translação em um espaço bidimensional, suas posições e velocidades devem ser objetos tridimensionais, assim como suas velocidades. Para denotar as componentes individuais de cada partícula, utilizamos um segundo subíndice k pertencente ao conjunto fechado $[1, 3]$. Seja $rnd()$ uma função tal que retorne um número pseudoaleatório uniformemente distribuído no intervalo $[0, 1]$. Seja também, na explicação desse algoritmo, ω o símbolo usado para representar o coeficiente de inércia de cada partícula e ϕ o coeficiente cognitivo. Assumimos *ab initio* que o coeficiente cognitivo seria de mesmo valor que o coeficiente social. Para facilitar a leitura e compreensão, denotamos por $pbest_{ik}$ a componente k da melhor posição alcançada por uma determinada partícula durante toda sua exploração do espaço de buscas, e $gbest_k$ a melhor partícula do enxame na presente iteração. A velocidade referida no Algoritmo 2, segue a Equação 4,

$$x_{ik}(t+1) = \omega * v_{ik}(t) + \phi * (rnd() * (pbest_{ik} - x_{ik}(t)) + rnd() * (gbest_k - x_{ik}(t)). \quad (4)$$

A atualização de posição que rege o movimento de cada partícula do enxame segue a Equação 5,

$$x_{ik}(t+1) = x_{ik}(t) + v_{ik}(t+1). \quad (5)$$

Empregamos, para a avaliação da distância entre a varredura esperada e a obtida, a Equação 6,

$$f(x,y) = \sum_{i=1}^{n} \frac{255 - map(x,y)}{255} + \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} map(x_r + k, y_r + l). \quad (6)$$

Algorithm 2 PSO para Alinhamento de Varredura

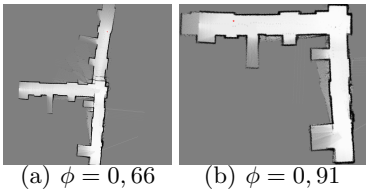
entrada z_t,z_{t-1},ϵ,ω,ϕ,N
  Inicialize o Enxame
  repita
    Ponha uma cópia da melhor transformação do último passo de localização no lugar daquela com o maior d_i.
    para $i = 1, 2, ..., N$ **faça**
      Transforme a pose atual do robô de acordo com p_i.
      Avalie a diferença (d_i) entre as posições esperadas e as observadas para cada obstáculo(dado o mapa) e a varredura adquirida
      por meio da Equação 6.
      Atualize o **Gbest** para que se refira à partícula com o menor d.
      para $k = 1, 2, .., 3$ **faça**
        Atualize v_{ik} segundo a Equação 4.
      fim para
    fim para
  até $d \leq \epsilon$

Os argumentos para a equação 6 são a varredura mais recente, o mapa gerado até a presente iteração e a pose do robô (ou ao menos a melhor estimativa disponível) e sua imagem é um número real quantificando a diferença entre a ocupação observada e a estimativa *a priori* (expressa sob a forma de intensidade do pixel no mapa em escala de cinza). Na Equação 6, o primeiro somatório é realizado com respeito a todos os feixes de raio laser presentes na varredura mais recente. É importante notar que, na forma apresentada, o algoritmo funciona pela minimização de um termo proporcional à probabilidade de que cada célula do mapa que representa um obstáculo esteja desocupada. Assume-se que x e y representem, na Equação 6, o conjunto das coordenadas cartesianas de cada obstáculo detectado no sistema global de coordenadas cartesianas (sistema de coordenadas do mapa). Além disso, observamos que a inclusão do segundo somatório, em que penalizamos as hipóteses de que o robô se encontre em regiões ocupadas. Com efeito, o segundo termo da Equação 6, penaliza a existência de obstáculos numa região de nove células em torno do robô. Nesse termo, x_r e y_r representam as coordenadas translacionais do robô no sistema de referência do mapa. Com efeito, realizar a avaliação da qualidade de uma estimativa de pose baseada num alinhamento do tipo pose a mapa, toda a história passada é levada em consideração consoante seu efeito no presente estado do mapa. A vantagem de uma abordagem desse tipo é relativa a uma maior insensibilidade à insensibilidade da estimativa obtida em relação à incerteza acerca do movimento do robô (*motion model uncertainty*). Por outro lado, levando em consideração apenas os efeitos visíveis no mapa das observações passadas torna possível que se fique "preso" em um ótimo local se, por exemplo, o ambiente apresentar relativamente poucas características observáveis. Por ambiente com relativamente poucas características observáveis referi-mo-nos a caso em que mudanças (potencialmente grandes) de posição levem a poucas ou nenhuma alteração na percepção das

cercanias do robô, como é o caso, por exemplo, em grandes corredores. Em casos assim, outras hipóteses devem ser feitas para que o sistema seja capaz de realizar a tarefa a que se propõe. Em [5] os autores explicitam a expectativa de que o robô deve exibir inércia e, portanto, ser incapaz de realizar mudanças repentinas ao seu *momentum*. Mingas et al. usam essa hipótese(vide [5]) para justificar o uso que fazem do elitismo e do modo como inicializam a população na implementação deles de um algoritmo genético para a realização de SLAM.

6 Calibração do PSO

O algoritmo de otimização por enxame de partículas possui numerosos parâmetros em sua formulação, de cujo ajuste depende o desempenho apresentado pela técnica em determinada tarefa. Além dos parâmetros previamente citados, há também que se fixar um número máximo de iterações, cuja especificação garante o término do processo de otimização mesmo que a função objetivo seja tão irregular que nem a imposição de uma tolerância de precisão nem condições de estagnação sejam atendidas num tempo razoável. Antes de realizar qualquer tentativa de analisar a influência dos parâmetros que afetam a velocidade das partículas possuem na acurácia do sistema como um todo, assumimos, como já foi dito anteriormente, que os coeficientes cognitivo e social deveriam ter o mesmo valor. A razão para tal decisão foi a crença de que igual importância deveria ser atribuída à sabedoria coletiva do enxame (busca global) e à sabedoria individual (busca local). O processo de calibração começou pela realização de uma amostragem dos efeitos advindos da alteração dos supramencionados parâmetros que influenciam a velocidade causavam na trajetória inferida. Durante essa fase, aplicamos o sistema a um simples ambiente, na Figura 1 mostramos duas execuções diferentes do algoritmo, a representação interna usou uma escala de 10 cm por célula. A massa de dados que utilizamos para a

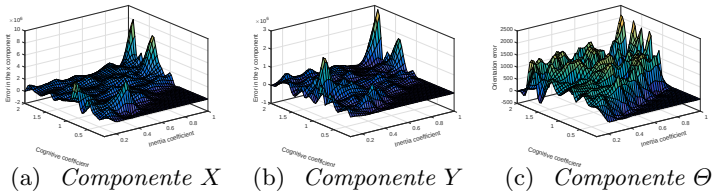


(a) $\phi = 0,66$ (b) $\phi = 0,91$

Fig. 1. As Figuras mostram duas execuções com 300 partículas e $\omega = 0,51$ executadas com dois coeficientes cognitivos diferentes no mesmo corpo de dados.

calibração desses parâmetros foi obtida realizando 30 repetições em cada configuração. Os parâmetros foram espaçados em intervalos de $0,01$, sendo o coeficiente de inercia pertencente ao intervalo fechado $[0,1]$, e o coeficiente cognitivo retirado do intervalo fechado $[0,2]$. Os dados assim obtidos foram, em seguida usados para construir uma interpolação da soma dos erros quadráticos em relação

à trajetória estimada por meio da média de diversas rodadas com o sistema de [5] e assumida como *ground truth*. A Figura 2(a) e Figure 2(b) mostram o erro translacional como função dos coeficientes de inércia e cognitivo, a Figura 2(c) mostra o erro na componente rotacional da pose em função desses mesmos parâmetros. É evidente, por inspeção dos gráficos, que se a componente angular possui carácter multi-modal e, portanto, é extremamente sensível a variações dos parâmetros. A determinação de um conjunto adequado de parâmetros, de



(a) *Componente X* (b) *Componente Y* (c) *Componente Θ*

Fig. 2. A Figura ilustra as componente do erro estimadas em função dos coeficientes cognitivo e social.

posse de uma amostra representativa do espaço de buscas em questão, é uma instância de um problema de otimização. Reduzimos, inicialmente, o problema multi objetivo de reduzir o erro sobre a trajetória, como mensurado pela função objetivo definida na Equação 7,

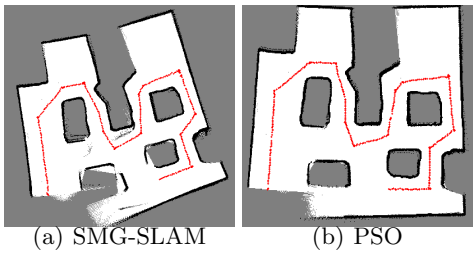
$$h(cog, inercia) = \sum_{i=1}^{n} u^2 + v^2 + w^2 \tag{7}$$

,em que denotamos por u o erro na trajetória na direção x, por v o erro na trajetória na direção y e por w o erro na orientação. Todos esses termos estão ilustrados nas Figuras 2(a),2(b) e 2(c), respectivamente. A função objetivo para a otimização dos dois parâmetros com que lidamos até agora é definida pela Equação 7. Aplicamos à Equação 7 o fminsearch do MATLAB, com o coeficiente de inercia restrito a pertencer ao intervalo fechado $[0, 1]$ e o coeficiente cognitivo restrito ao intervalo $[0, 2]$. O procedimento convergiu consistentemente para os valores de $0,4953$ para o coeficiente de inercia e $1,3927$, ambos bastante próximos dos valores usuais de $0,7$ e $1,49$. Uma vez fixados esses parâmetros mudamos nossa atenção para o número de partículas, e com alguma experimentação descobrimos que a precisão do sistema não se alterava significativamente com enxames muito maiores que 45 a 50 partículas.

7 Resultados

Todos os testes cujos resultados aqui apresentamos foram realizados em PCs com processador Intel Core i7 950 3GHz, memória RAM de 8 Gb e sistema

operacional Microsoft Windows 7 Home Premium. tempo médio de processamento para 50 particulas foi de $0,0530$s por varredura[8], com um desvio padrão de $0,0018$s. Levando em consideração que nosso trabalho foi fortemente inspirado por [5], escolhemos começar por comparar nossos resultados aos deles. Na Figura 3(a), nós exibimos um mapa contido em [5] (disponível no Repositório [11]) e na Figura 3(b) exibimos o resultado de nosso método aplicado ao mesmo corpo de dados de varredura (sem informação odométrica). O GA foi rodado com uma população de 2000 indivíduos e usando as configurações padrão. O PSO foi executado com um enxame de 50 partículas, $\omega = 0,4953$ e $\phi = 1,3927$. Apli-



(a) SMG-SLAM (b) PSO

Fig. 3. As trajetória estimadas pelos métodos comparados encontram-se marcadas em vermelho.

camos nosso método também ao Dataset disponibilizado em [12] por Andrew Howard, denominado "usc-sal200-021120", e o resultado pode ser observado na Figura 4. Nesse experimento não fizemos uso dos dados odométricos fornecidos, com o intuito de atestar que o método proposto é capaz de realizar o passo de localização adequadamente, possibilitando a construção de mapas coesos em ambientes com laços (*loops*). O mapa exibido na Figura 4 apresenta uma resolução

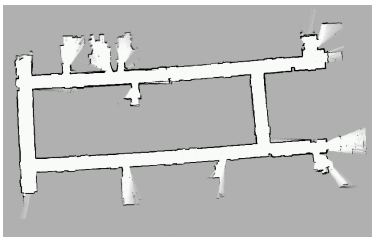


Fig. 4. Mapa obtido com PSO do dataset USC-SAL.

de e foi obtido mediante o uso de um enxame composto por 50 partículas com $\omega = 0,4953$ e $\phi = 1,3927$. A Figura 4 exibe um mapa que o sistema, após a

[8] Os testes foram repetidos 30 vezes.

devida calibração, se mostrou capaz de distinguir entre os diversos corredores presentes no ambiente mapeado de modo suficientemente preciso para formular um mapa razoável.

8 Conclusões

O sistema apresentado, embora possua resultados promissores, exibe grande sensibilidade aos parâmetros do método de buscas, como mostramos. A capacidade geral de detectar *Loop Closures* é particularmente sensível à alteração desses parâmetros. De fato, durante a etapa de calibração, numerosas vezes observou-se o fenômeno de redesenho do mesmo obstáculo em regiões distintas do mapa, indicando a falha do sistema de inferir que retornou ao mesmo local.

Referências

1. M. Kaess, A. Ranganathan, and F. Dellaert, "isam : Incremental smoothing and mapping," 2008.
2. R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings - IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, 2011, pp. 3607–3613.
3. F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997.
4. P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," pp. 239–256, 1992.
5. G. Mingas, E. Tsardoulias, and L. Petrou, "An fpga implementation of the smg-slam algorithm," *Microprocess. Microsyst.*, vol. 36, no. 3, pp. 190–204, May 2012. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2011.12.002
6. S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, 2005.
7. A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.
8. J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.
9. J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Evolutionary Computation, 1997., IEEE International Conference on*, Apr 1997, pp. 303–308.
10. A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *International conference on artificial intelligence*, vol. 1, 2000, pp. 429–434.
11. G. Mingas, E. Tsardoulias, and P. Loukas. (2015) Smg-slam. [Online]. Available: https://github.com/gmingas/slam-1.0
12. A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/