Influence of Location and Number of Landmarks on the Monte Carlo Localization Problem

Henrique José dos S. Ferreira Júnior[1], **Daniel Ratton Figueiredo** [2]

[1]Universidade Federal do Rio de Janeiro (UFRJ)
Escola Politécnica (POLI), Departamento de Engenharia Eletrônica (DEL)
Rio de Janeiro - RJ

[2]Universidade Federal do Rio de Janeiro (UFRJ)
Programa de Engenharia de Sistemas e Computação (PESC)
Rio de Janeiro - RJ

henriquejsfj@poli.ufrj.br, daniel@cos.ufrj.br

Abstract. *An important problem in robotics is to determine and maintain the position of a robot that moves through a previously known environment with reference points that are indistinguishable, which is made difficult due to the inherent noise in robot movement and identification of reference pints. Monte Carlo Localization (MCL) is a frequently used technique to solve this problem and its performance intuitively depends on reference points. In this paper we evaluate the performance of MCL as a function of the number of reference points and their positioning in the environment. In particular, we show that performance is not monotonic in the number of reference points and that a random positioning of the reference points is close to optimal.*

Resumo. *Um importante problema em robótica é determinar e manter a posição de um robô em movimento por um ambiente previamente conhecido que possui pontos de referência indistinguíveis, que é dificultado pelo ruído inerente na movimentação do robô e na identificação dos pontos de referência. Localização via Monte Carlo (MCL) é uma técnica comumente utilizada para resolver este problema, cujo desempenho depende intuitivamente dos pontos de referência. Neste artigo avaliamos o desempenho do MCL em função do número de pontos de referência e do posicionamento dos pontos pelo ambiente. Em particular, mostramos que o desempenho não é monotônico no número de pontos de referência e que o posicionamento aleatório dos pontos está próximo do ótimo.*

1. Introdução

Um importante problema em robótica é determinar e manter a posição de um robô em movimento por um ambiente que possui pontos de referência. Neste problema, o robô possui um mapa completo do ambiente assim como a posição exata de pontos de referência que estão espalhados pelo ambiente. Entretanto, tais pontos de referência que são indistinguíveis, ou seja, o robô não sabe distinguir entre um ponto e outro (ex. marcas idênticas espalhadas por uma casa). Além disso, outro desafio neste problema é que a movimentação do robô assim como a identificação de um ponto de referência possuem ruídos, no sentido de que o robô não se movimenta nem afere um ponto de referência com

exatidão (ex. ao tentar se movimentar para o norte, o robô pode permanece no mesmo lugar). O objetivo do robô é navegar pelo ambiente e determinar (e manter) sua posição no mapa, em função dos encontros com os pontos de referência. Este é um problema de localização global, que difere do problema de rastreamento ou navegação, no qual o robô parte de um ponto conhecido e se movimenta até um destino final utilizando um mapa do ambiente.

Uma técnica para resolver este problema de localização global é a *Monte Carlo Localization* (MCL), que apesar de proposta em 1999 [Fox et al. 1999, Thruna et al. 2001], vem sendo aplicada em cada vez mais cenários [Muzio et al. 2016]. De forma sucinta, o MCL utiliza inferência bayesiana recursiva, gerando amostras aleatórias de possíveis locais para o robô (chamadas de partículas), dando mais peso para locais que condizem com suas observações (pontos de referência) e ações do robô. Ao se movimentar, o robô deve aplicar as leis de movimentação deslocando as partículas e ao identificar um ponto de referência deve atribuir pesos maiores as partículas que correspondam a pontos de referência. No próximo passo, as partículas são amostradas novamente seguindo os pesos de forma que as partículas fiquem mais concentradas no local real do robô. Ao longo desse processo as configurações de crenças sobre o ambiente vão ficando com mais pesos até que haja uma região única de alta probabilidade que é interpretada como a localização global do robô. A figura 1 ilustra este processo.

Figura 1. Evolução do posicionamento das partículas ao longo do tempo. No início do algoritmo o robô não sabe onde está, então as partículas estão espalhadas uniformemente. Em seguida há duas regiões possíveis, por conta da ambiguidade nos pontos de referência, até que o robô se localize globalmente.

Intuitivamente, o número de pontos de referência e o posicionamento destes pontos no ambiente influenciam diretamente o desempenho do MCL. Por exemplo, um maior número de pontos de referência permite o robô mais rapidamente encontrar um destes pontos ao se movimentar pelo ambiente. Desta forma, surgem duas perguntas fundamentais: Quantos pontos de referência devem ser colocados em um determinado ambiente? Dado o número de pontos de referência, onde posicionar estes pontos no ambiente? O objetivo deste trabalho é justamente atacar estas duas perguntas, e até onde sabemos este problema ainda não foi considerado na literatura relacionada.

Nesse trabalho iremos considerar ambientes na forma de reticulados (grids) bidimensionais onde cada local do reticulado pode possuir um simples ponto de referência (todos idênticos). O desempenho do MCL será medido pela fração de vezes que o algoritmo acerta a posição real do robô. Mostramos que o desempenho em função do número de pontos de referência, para um ambiente fixo, não é monotônico, exibindo um valor ótimo de pontos. Em seguida mostramos que o posicionamento aleatório dos pontos de

referência possui desempenho próximo do melhor posicionamento possível. Para encontrar o melhor posicionamento, utilizamos a técnica de *simulated annealing*, tendo em vista a complexidade combinatorial da quantidade de posicionamentos possíveis. Esta abordagem para solução para deste problema de otimização é uma outra contribuição deste trabalho. Nossos resultados indicam que o posicionamento aleatório dos pontos é muito eficiente, principalmente tendo em vista sua baixa complexidade computacional.

O restante deste artigo está organizado da seguinte forma. Na seção 2 apresentamos sucintamente alguns trabalhos relacionados. Na seção 3 apresentamos o ambiente e o modelo para o robô assim como as métricas para avaliar o desempenho do MCL. Nas seção 4 e 5 apresentamos a avaliação numérica do MCL quanto a quantidade e posicionamento dos pontos de referência, respectivamente. Por fim, apresentamos uma breve conclusão e trabalhos futuros na seção 6.

2. Trabalhos Relacionados

A técnica de *Monte Carlo Localization* (MCL) [Fox et al. 1999, Thruna et al. 2001] para o problema de localização global de robôs se baseia em técnicas de Monte Carlo e inferência bayesiana. Desde que foi proposta, MCL vêm sendo aplicada e generalizada para diversos contextos onde surgem problemas de localização [Torma et al. 2010, Payá et al. 2010, Elinas and Little 2005, Milstein 2008, Muzio et al. 2016, Metropolis et al. 1953, Howard 2006]. Entretanto, nenhum destes trabalhos abordou a questão da quantidade e posicionamento dos pontos de referência no desempenho do MCL, que é o foco deste trabalho.

3. Modelo e Métricas

Iremos considerar um ambiente na forma de um reticulado (grid) bi-dimensional e sem borda (torus) de tamanho $n*n$. Ou seja, cada ponto do ambiente possui exatamente quatro vizinhos e $n + 1$ movimentos na mesma direção levam de volta ao ponto inicial. Cada ponto do ambiente representa um possível local para o robô, que neste caso possui n^2 locais diferentes. Iremos adicionar ao ambiente p pontos de referência, todos idênticos. A figura 2 ilustra um ambiente onde $n = 10$ e $p = 50$, de forma que os pontos de referência são os pontos pretos.

Inicialmente, no tempo $t = 0$, o robô é posicionado de forma aleatória e uniforme em algum estado do ambiente e o mesmo não possui qualquer informação sobre sua localização. Iremos considerar um sistema de tempo discreto, onde a cada passo o robô realiza um movimento e uma leitura para identificar a presença de um ponto de referência. O movimento desloca o robô para um estado adjacente ao atual: para cima, baixo, esquerda ou direita. O direção do movimento é escolhida de forma aleatória, com igual probabilidade. Repare que o robô não tem como objetivo chegar a um alvo predeterminado, mas sim se localizar (e se manter localizado) no ambiente. A figura 2 ilustra a posição do robô em um ambiente com pontos de referência (quadrados pretos) e os possíveis estados para o qual pode ser movimentar (quadrados azuis).

Para modelar o ruído na movimentação do robô, iremos considerar que o robô tem uma chance de 10% de dar um sobrepasso, ou seja andar um estado a mais na mesma direção escolhida, e 10% de dar um subpasso, ou seja, permanecer no mesmo local (não

se mover). Desta forma, a tentativa de movimentação na direção escolhida ocorre satisfatoriamente com chance de 80%. É importante notar que o robô não é capaz de determinar se houve um sobrepasso ou subpasso ao tentar se locomover.

Para modelar ruído na identificação dos pontos de referência, uma leitura informa ao robô se o estado em que ele se encontra é preto (possui ponto de referência) ou branco (não possui ponto de referência) com erro de 10%, em ambos os casos. Ou seja, o robô detecta corretamente a presença ou ausência de um ponto de referência em seu atual estado com chance de 90%.

Uma das variações do MCL, que será utilizada neste trabalho, mantém a probabilidade do robô se encontrar em cada possível estado (local) do ambiente. Esta probabilidade vai sendo modificada a cada passo do robô, tanto em função da sua movimentação quanto em função da observação de um ponto de referência. Seja $P_x(t)$ a probabilidade segundo o MCL do robô se encontrar no estado x do ambiente no instante de tempo t. Repare que $P_x(0) = 1/n^2$, pois neste instante de tempo o robô não tem nenhuma informação sobre sua localização.

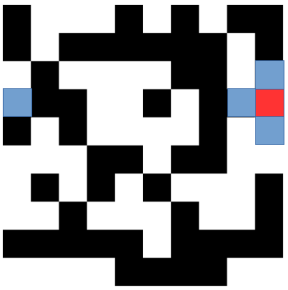

Figura 2. Exemplo de um grid com p=50 com robô no quadrado vermelho

3.1. Métricas de desempenho

Precisamos definir métricas para aferir o desempenho do algoritmo MCL de forma a quantificar seu desempenho em função da quantidade e posicionamento dos pontos de referência, propusemos então duas. A primeira métrica captura a fração de vezes (movimentos) que o MCL acerta a posição correta do robô. A segunda métrica captura quanto tempo (movimentos) é necessário para o robô se encontrar corretamente pela primeira vez (a partir de um estado inicial desconhecido). No que segue formalizamos estas duas métricas, que foram propostas neste trabalho apesar de intuitivas.

Seja $R(t)$ o estado (posição no ambiente) do robô no instante de tempo t. Seja $L(t) = \{y \mid y = \arg\max_x P_x(t)\}$, onde $P_x(t)$ é a probabilidade segundo o MCL do robô estar no estado x no tempo t. Ou seja, $L(t)$ é o conjunto de estados de maior probabilidade no tempo t. Vamos definir $A(t) = 1$ se $|L(t)| = 1$ e $L(t) = R(t)$, e 0 caso contrário. Ou seja, $A(t)$ indica que o MCL acertou a posição do robô no instante de tempo t. Seja r o número de passos executados no MCL em um determinado ambiente G. Vamos definir

$$E_G(r) = \frac{\sum_{t=1}^{r} A(t)}{r} .$$

A métrica $E_G(r)$ busca capturar a intuição de que o desempenho do MCL é diretamente proporcional a fração de tempo (passos) em que o robô se localiza corretamente

no ambiente. De fato, $E_G(r)$ é a taxa de acertos do MCL depois de r passos, que converge quando $r \to \infty$, tendo em vista que o ambiente é finito. Entretanto, repare que $E_G(r)$ (para r fixo) é uma variáel aleatória tendo em vista que o MCL é aleatório. Para um r fixo (ex. $r = 100$), podemos estimar o valor esperado de $E_G(r)$ fazendo a média amostral de várias rodadas.

Uma outra métrica importante é o tempo (medido em número de passos) até o primeiro acerto. Seja F_G o primeiro instante de tempo t onde $A(t) = 1$ para um determinado ambiente G. Ou seja:

$$F_G = \min t \mid A(t) = 1 \, .$$

Repare que F_G é uma variável aleatória, pois depende da execução do algoritmo (pois movimentação e leitura são aleatórios). Iremos reportar a média amostral de F_G ao executar o algoritmo por muitas rodadas independentes. Repare que um baixo valor nesta métrica é importante, pois indica que o robô é capaz de rapidamente se localizar no ambiente sem possuir qualquer informação prévia sobre sua posição.

3.2. Simulador

Um simulador do modelo apresentado para movimentar o robô e detectar a presença de pontos de referência foi projetado e implementado para este trabalho. Além disso, o algoritmo de MCL também foi implementado para determinar a localização do robô em função das observações. O mesmo possui diversos parâmetros, tais como o tamanho do ambiente (grid bi-dimensional), e a quantidade e posicionamento dos pontos de referência. As métricas de desempenho propostas foram coletadas pelo simulador, que executa diversas rodadas independentes. O simulador está disponível publicamente em: `https://gitlab.com/henriquejsfj/Monte_Carlo_Localization.git`

4. Avaliação da Quantidade de Pontos

Iremos avaliar o desempenho do MCL em função do número de pontos de referência, p, presentes no ambiente. Os pontos de referência serão posicionados de forma aleatória e uniforme no ambiente. Para cada posicionamento dos pontos, executamos 100 rodadas independentes usando o simulador, cada uma com $r = 2000$, e apresentamos a média amostral das medidas de desempenho destas rodadas.

As figuras 3 e 4 mostram as duas métricas de desempenho do MCL para diferentes quantidades de pontos de referência nos dois tamanhos de grid, respectivamente. Note que nas duas métricas, o desempenho melhora com o aumento da quantidade de pontos de referência até atingir um valor ótimo e depois piora com o aumento da quantidade de pontos. Ou seja, a fração de acertos tem um valor máximo e o tempo até o primeiro acerto possui um valor mínimo. Nos dois casos, este valor ótimo de pontos de referência está em $n^2/2$ ou seja, quando exatamente a metade dos estados possuem pontos de referência. Além disso, este valor máximo para a fração de acertos é cinco vezes maior do que o valor nos extremos, ou seja, a diferença de desempenho é bastante significativa.

Intuitivamente, com $n^2/2$ pontos de referência temos a melhor quantidade de informação no ambiente, já que metade dos estados são pontos de referência. Ao aumentarmos ou diminuirmos este valor, iremos ter mais simetria (ou com pontos de referência

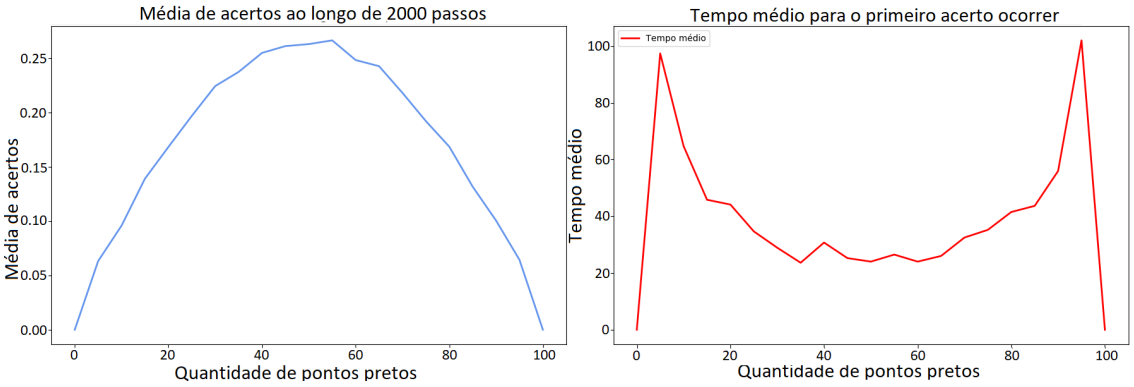

Figura 3. Desempenho do MCL em função da quantidade de pontos de referência (com $n = 10$): taxa de acertos, E_G (à direita), tempo médio do primeiro acerto, F_G (à esquerda).

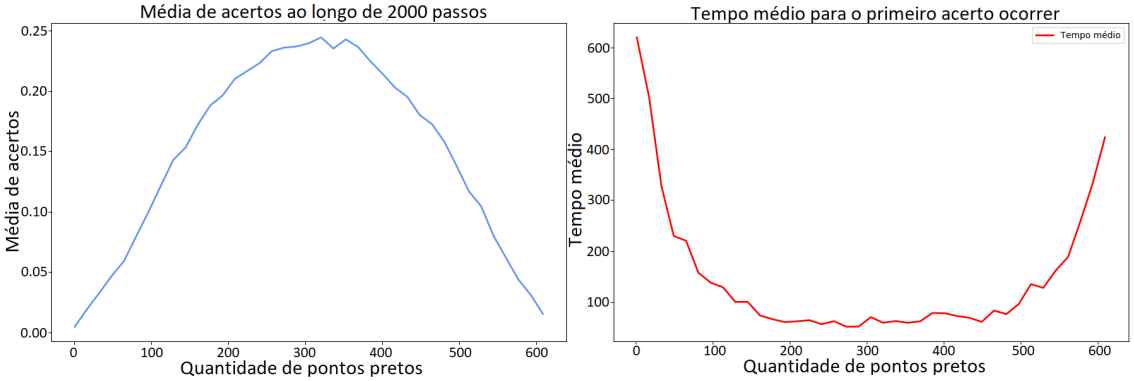

Figura 4. Desempenho do MCL em função da quantidade de pontos de referência (com $n = 25$): taxa de acertos, E_G (à direita), tempo médio do primeiro acerto, F_G (à esquerda).

ou com a ausência destes), levando a um desempenho pior. Este resultado está relacionado a entropia que existe no ambiente, que é máxima quando temos metade dos estados como sendo pontos de referência.

Para analisar esta intuição de informação no ambiente induzida pelos pontos de referência podemos considerar a evolução de $P_{R(t)}(t)$, ou seja, da probabilidade dada pelo MCL para a real posição do robô ao longo do tempo. A figura 5 ilustra uma evolução desta probabilidade para $p = 1$ e $p = 50$. Os picos da curva com $p = 1$ ocorrem exatamente quando o robô encontra algum ponto de referência, entretanto, esta probabilidade rapidamente cai, por conta dos ruídos na movimentação do robô. Por outro lado, a curva com $p = 50$ tem muitos picos e é mais estável (e com valores maiores), já que encontra pontos de referência com maior frequência.

5. Avaliação do Posicionamento dos Pontos

Encontrar o posicionamento dos pontos de referência que maximiza o desempenho do MCL não é uma tarefa fácil pois a quantidade de possíveis posicionamentos é muito grande. Por exemplo, para $n = 10$ e $p = 50$ temos um total de $\frac{100!}{50!50!} = 10^{29}$ combinações possível. Além disso, a estimativa do desempenho do MCL para um dado posicionamento

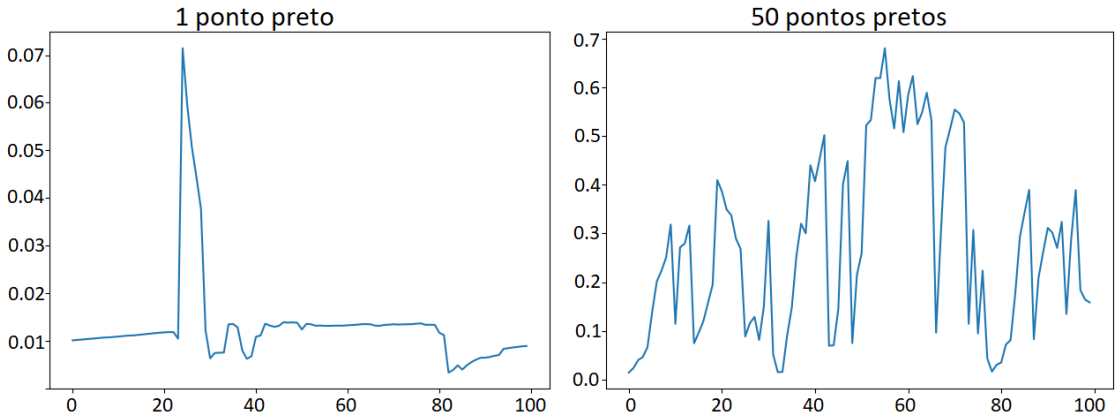

Figura 5. Evolução da probabilidade dada pelo MCL para a real posição do robô ao longo do tempo, para $n = 10$, e duas quantidades de pontos de referência, $p = 1$ e $p = 50$.

de pontos de referência leva em torno de 50s (assumindo $r = 2000$, como na avaliação anterior), de forma que avaliar todas as possíveis combinações levaria 10^{22} anos, que é bilhões de vezes maior que a idade do universo!

5.1. *Simulated Annealing*

Para atacar o problema de encontrar o posicionamento ótimo dos pontos de referência iremos utilizar *simulated annealing*, que é uma técnica de otimização que utiliza cadeias de Markov e métodos de Monte Carlo [Kirkpatrick et al. 1983]. Para aplicar a técnica é necessário definir o espaço de estados, as possíveis transições entre os estados e suas probabilidades (ou seja, a cadeia de Markov), o valor (recompensa) de um estado, e uma estratégia de resfriamento. Algumas vantagens desta técnica é ser iterativa (não precisar gerar o espaço de estados), e não ficar restrita a máximos locais sendo possível encontrar valores ótimos globais em alguns casos (dependendo da estratégia de resfriamento).

No problema em questão, iremos considerar um espaço de estados que consiste de todas as possíveis configurações de pontos de referência no ambiente, para um determinado número fixo (p) de pontos de referência. Seja G um estado, ou seja, um determinado posicionamento dos pontos de referência no ambiente. Iremos avaliar a qualidade de G usando a métrica E_G, ou seja, a fração de acertos de localização do robô nesta configuração de pontos de referência.

A técnica de *simulated annealing* utiliza a distribuição de Boltzmann sobre o espaço de estados que possui ainda uma temperatura, que vai sendo resfriada (diminuída) ao longo do processo iterativo de otimização. Em particular, a distribuição de probabilidade sobre os estados para um determinada temperatura T é dada por

$$\pi_G = \frac{e^{\frac{E_G}{T}}}{Z},$$

onde $Z = \sum_{G \in \mathcal{G}} e^{\frac{(E_G}{T}}$ é a constante de normalização e $\mathcal{G}$ é o conjunto com todas as configurações possíveis de posicionamento dos pontos de referência.

5.2. Regra de transição entre configurações

A regra de transição entre os estados é um aspecto central à técnica de *simulated annealing*, que determina sua eficiência em encontrar estados ótimos. Consideramos a seguinte regra simples de transição: dada uma configuração G de pontos de referência, escolhemos aleatoriamente um dos pontos de referência e escolhemos aleatoriamente um dos locais onde não há pontos de referência (ambas escolhas com probabilidade uniforme), movemos o ponto de referência para o local escolhido, dando origem a uma nova configuração G'. Repare que G e G' diferem em apenas um ponto de referência. Além disso, qualquer configuração G terá como vizinho um total de $p(n^2 - p)$ estados vizinhos, onde p é o número de pontos de referência. Além disso, todas estas transições são equiprováveis, fazendo com que a cadeia de Markov seja simétrica, ou seja $p_{G,G'} = p_{G',G}$ para todo par G, G' de configurações.

Para evitar a enumeração explícita de todos as configurações vizinhas ao executar o *simulated annealing*, podemos escolher a configuração vizinha utilizando Metropolis-Hasting, um método de transição para cadeias de Markov que garante a distribuição estacionária desejada.

O Metropolis-Hasting funciona atribuindo uma probabilidade de aceitação $a(G, G')$ da transição da configuração G para a configuração G' que depende da distribuição estacionária desejada π e na probabilidade de transição da cadeia de Markov original. Em nosso caso, temos que π_G é dado pela distribuição de Boltzman (apresentada acima) e que a cadeia de Markov original é simétrica. Neste caso, temos a seguinte definição para as probabilidades de aceitação (para um determinado valor de T fixo):

$$a(G, G') = min(1, e^{(E_G - E_{G'})/T)}.$$

Repare que se $E_{G'} > E(G)$ então a probabilidade de aceitação é 1, ou seja, sempre aceitamos uma configuração melhor que a atual. Caso contrário, o novo estado apesar de inferior é aceito com probabilidade não nula. Em todo o caso, é desejado que o número de estados não aceitos (rejeitados), seja baixo, de forma que o espaço de estados possa ser mais explorado de forma mais eficaz.

A estratégia de resfriamento consiste de uma função de decaimento para T em função do número de transições realizadas na cadeia de Markov. Entretanto, neste trabalho iremos assumir que $T = 1$ e constante, ou seja, não teremos uma estratégia de resfriamento. Uma vantagem desta abordagem é não precisar determinar a estratégia de resfriamento mais adequada para o problema. Além disso, iremos guardar o melhor estado (configuração de pontos) visitado pelo processo de otimização, que será usado como solução ao final. Intuitivamente, isto permite explorar melhor o espaço de estados sem ficar preso em máximos locais, tendendo a seguir por estados que possuem melhor desempenho.

5.3. Resultados

Um aspecto importante é a exploração do espaço de estados (posicionamento dos pontos de referência) pelo método de *simulated annealing*. A figura 6 ilustra uma configuração de pontos de referência simétrica, com os pontos alinhados em listras, que foi utilizada

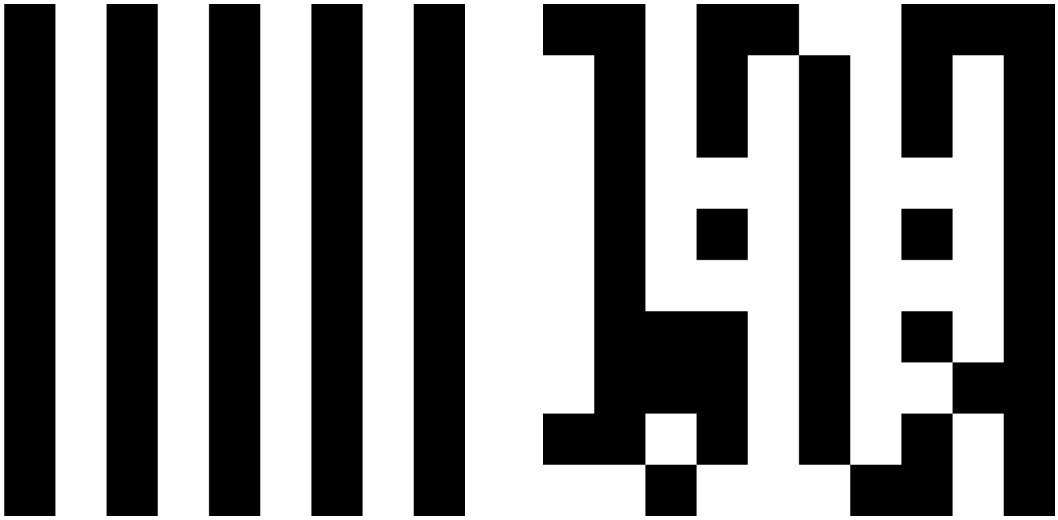

Figura 6. A esquerda, configuração de pontos de referência em padrão listrado, usado como estado inicial no processo de otimização. A direita, configuração dos pontos de referência depois de 10 passos do processo de otimização.

como estado inicial da técnica de otimização. A figura também ilustra o estado do depois de apenas 10 passos do processo de otimização, que já visivelmente diferente e bem mais eficiente do que a configuração inicial.

É importante também considerar a distribuição do valor de E_G tendo em vista que para uma configuração de pontos de referência e um valor fixo de r, E_G é uma variável aleatória. Em particular, para avaliar a qualidade de um estado (configuração), iremos usar $r = 2000$ para não comprometer a eficiência do método e podermos explorar mais estados. Repare que a qualidade de uma configuração será uma amostra de E_G, pois o mesmo é avaliado apenas uma vez (com $r = 2000$).

A figura 7 apresenta um histograma dos valores de E_G obtidos em 1000 avaliações independentes de uma configuração de pontos de referência escolhida de forma aleatória (mas usada para as 1000 avaliações). Podemos observar que a distribuição se assemelha de uma distribuição gaussiana, com média 0.263 e desvio padrão amostral de $\sigma = 0.00336$. Além disso, os valores obtidos na cauda da distribuição estão com cerca de apenas $3{*}\sigma$, sendo este decaimento parecido com a distribuição gaussiana. A figura indica que o erro na estimativa do valor de E_G para uma determinada configuração G usando apenas uma amostra não será muito grande.

As figuras 8 e 9 mostram o valor da melhor configuração de pontos de referência visitada (maior valor de E_G) em função do número de transições do *simulated annealing*, para configurações iniciais diferentes (com $n = 10$ e $p = 50$). Além disso, as figuras também indicam as transições que foram rejeitadas pelo Metropolis-Hasting.

Repare que na configuração listrada de pontos de referência (estado inicial na figura 8), o valor de E_G é zero, pois o robô não consegue se localizar devido a simetria do posicionamento dos pontos de referência. Apesar deste estado inicial ruim, o método *simulated annealing* consegue rapidamente visitar estados bem melhores (maior valor para E_G), até convergir em configurações que não possuem valores muito similares para E_G, de forma que o melhor valor não é mais alterado, depois de 800 transições (ver

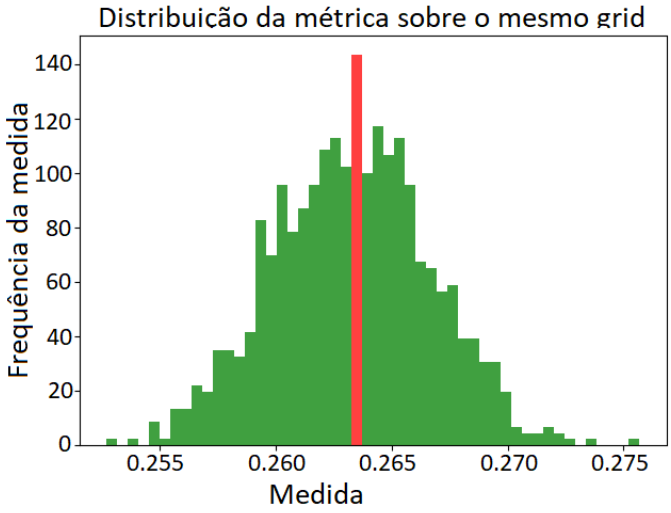

Figura 7. Histograma de 1000 amostras de E_G com $r = 2000$ para uma configuração fixa de pontos de referência, com média destacada em vermelho, e desvio padrão amostral $\sigma = 0.00336$ ($n = 10, p = 50$).

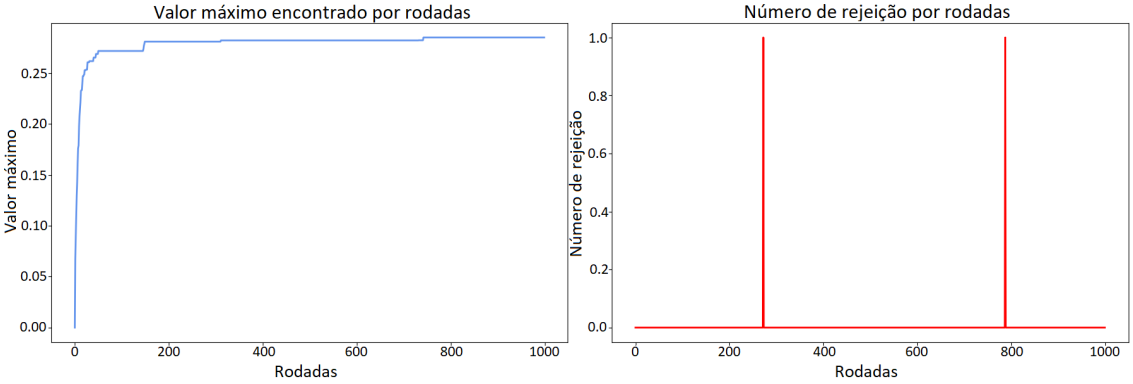

Figura 8. Melhor valor para E_G visitado em função do número de transições a partir da configuração inicial listrada (esquerda); rejeição de transição como determinado pelo Metropolis-Hasting em função das transições do *simulated annealing* (direita).

figura 8).

É interessante notar que nessa subida rápida do valor de E_G das primeiras 200 transições não houve nenhuma rejeição, de forma que o *simulated annealing* apenas realizou trocas de pontos de referência aleatoriamente, sem encontrar uma configuração que fosse inferior a atual. Note ainda que o valor para onde o melhor E_G convergiu foi similar ao valor médio de E_G para uma configuração aleatória (ilustrado na figura 7).

Estamos interessados agora em entender se o método de *simulated annealing* consegue obter configurações melhores que a aleatória, e neste caso qual é o valor de E_G destas configurações. Para avaliar este cenário, iremos iniciar o *simulated annealing* com uma configuração aleatória, e realizar transições pelo espaço de estado.

A figura 9 ilustra o melhor valor para E_G em função do número de transições do *simulated annealing*, ao iniciarmos de uma configuração aleatória. Podemos observar que

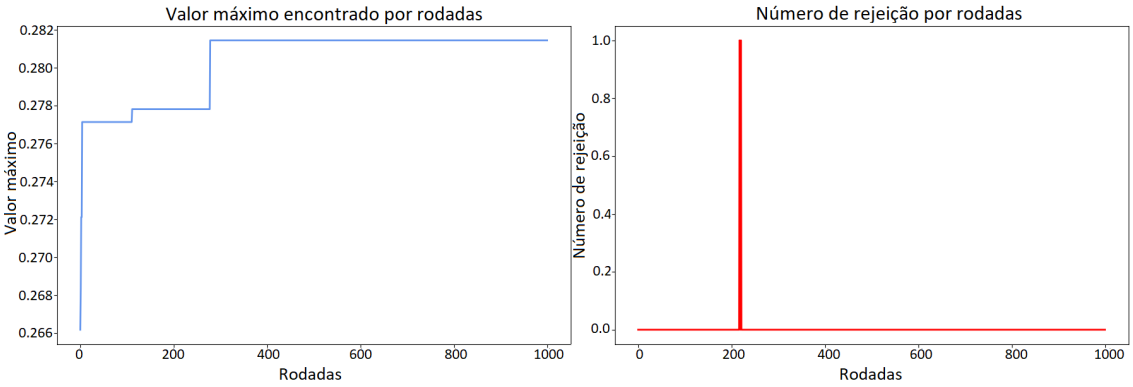

Figura 9. Melhor valor para E_G visitado em função do número de transições a partir da configuração inicial aleatória (esquerda); rejeição de transição como determinado pelo Metropolis-Hasting em função das transições do *simulated annealing* (direita).

a configuração inicial possui um bom desempenho e que a a melhor configuração encontrada possui desempenho um pouco superior, com diferença de 0.015. Esse valor pode ser considerado baixo pois não ultrapassa nem 5σ, tendo em vista a análise do desvio padrão de E_G para uma configuração aleatória (vide figura 7). De fato, o método encontrou uma configuração com desempenho superior que a configuração inicial, mas este benefício é pequeno e depende também da configuração inicial, que foi escolhida aleatoriamente.

A análise das rejeições das transições (figura 9, direita) mostra que praticamente não houveram rejeições, indicando que o *simulated annealing* continuou gerando configurações aleatórias cujo desempenho, E_G, são bem parecidos. Desta forma, podemos concluir que a configuração aleatória para os pontos de referência, em geral tem um desempenho próximo ao ótimo.

6. Conclusão

Este trabalho avaliou o desempenho do algoritmo MCL em função da quantidade e posicionamento dos pontos de referência indistinguíveis em reticulados bi-dimensionais. Através de um estudo empírico, utilizando simulação de um modelo de movimentação do robô com ruídos, mostramos que a quantidade de pontos influencia significativamente o desempenho do MCL, tanto na taxa de localização correta quanto no tempo para se localizar pela primeira vez.

Um aspecto interessante foi a observação de que este comportamento não é monotônico, e que um aumento demasiado do número de pontos de referência leva a um decaimento do desempenho do MCL. Isto ocorre pois um valor muito alto de pontos de referência leva a uma maior simetria, dificultando o posicionamento do robô. Os resultados indicam que o número ótimo de pontos de referência é justamente a metade dos possíveis estados (locais) do ambiente. Intuitivamente, com esta quantidade de pontos de referência maximizamos a quantidade de informação disponível para o robô realizar a localização.

A avaliação da influência do posicionamento dos pontos de referência foi realizada utilizando o método de *simulated annealing*, que demonstrou ser satisfatório em encontrar configurações que tenham bom desempenho (na métrica E_G). Isto foi verificado

quando a configuração inicial era um padrão listrado (sem a possibilidade de localização), e neste caso o método encontro configurações com desempenho bem superior, e também quando a configuração inicial era aleatória, apesar de neste caso o método ter encontrado configurações com desempenho superior desprezível. Isto indica que configuração aleatória dos pontos de referência em geram possuem um desempenho próximo ao ótimo.

Por fim, este estudo delineia um plano de ação para um melhor desempenho do MCL. Dado um ambiente qualquer, deve-se inserir no mesmo uma quantidade de pontos de referência que seja a metade da quantidade de estados (locais) para o robô. Além disso, estes pontos devem ser distribuídos de forma aleatória (e uniforme) pelo ambiente. Desta forma, o MCL terá desempenho próximo ao ótimo.

Referências

Elinas, P. and Little, J. J. (2005). σmcl: Monte-carlo localization for mobile robots with stereo vision. In *Proc. of Robotics: Science and Systems (RSS)*, volume 53.

Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 343–349.

Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Milstein, A. (2008). Occupancy grid maps for localization and mapping. In *Motion Planning*. InTech.

Muzio, A., Aguiar, L., Máximo, M. R., and Pinto, S. C. (2016). Monte carlo localization with field lines observations for simulated humanoid robotic soccer. In *XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pages 334–339.

Payá, L., Fernández, L., Gil, A., and Reinoso, O. (2010). Map building and monte carlo localization using global appearance of omnidirectional images. *Sensors*, 10(12):11468–11497.

Thruna, S., Foxb, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141.

Torma, P., György, A., and Szepesvári, C. (2010). A markov-chain monte carlo approach to simultaneous localization and mapping. In *Proc. 13th International Conference on Artificial Intelligence and Statistics*, pages 852–859.