Trabalho #6 – COE841

Professor: Ramon Romankevicius Costa

Alunos: André Abido Figueiró

Tiago Bornia de Castro

1. MONTE CARLO LOCALIZATION (MCL)

Algorithm MCL$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ *to* M *do*
        $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
        $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    endfor
    for $m = 1$ *to* M *do*
        draw i *with probability* $\propto w_t^{[i]}$
        add $x_t^{[i]}$ *to* $\mathcal{X}_t$
    endfor
    return $\mathcal{X}_t$

Tabela 2.1 - Algoritmo do MCL.

O algoritmo de *Monte Carlo Localization* se utiliza de partículas para representarem posições hipotéticas (possíveis estados) do robô, de forma semelhante ao implementado no Trabalho 2. Para cada uma das partículas representativas, aplica-se o **Modelo de Movimentação** do robô, que gera as novas posições hipotéticas conforme o modelo probabilístico adotado para este.

Uma vez calculadas as novas posições, o algoritmo se utiliza do **Modelo de Medição** para estimar as probabilidades associadas a cada nova posição hipotética, utilizando para isto o mapa do ambiente e as medidas realizadas pelo robô. Tais probabilidades são utilizadas como os pesos associados a cada posição.

Em um terceiro passo as partículas são reamostradas conforme o peso de cada uma, de forma a representar a função de distribuição de probabilidade associada à posição do robô.

2. AUGMENTED MONTE CARLO LOCALIZATION (AMCL)

Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
  static $w_{\text{slow}}, w_{\text{fast}}$
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ *to* M *do*
    $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M} w_t^{[m]}$
  endfor
  $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
  $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
  for $m = 1$ *to* M *do*
    with probability $\max\{0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}}\}$ *do*
      add random pose to $\mathcal{X}_t$
    else
      draw $i \in \{1, \ldots, N\}$ *with probability* $\propto w_t^{[i]}$
      add $x_t^{[i]}$ *to* $\mathcal{X}_t$
    endwith
  endfor
  return $\mathcal{X}_t$

Tabela 3.1 – Algoritmo do Augmented Monte Carlo Localization

O algoritmo de Augmented *Monte Carlo Localization* apresenta uma inovação em relação ao MCL tradicional de forma a aumentar a robustez do algoritmo, solucionando o "problema do robô sequestrado". O problema em si ocorre quando há uma movimentação do robô não medida pelos seus sensores ou quando se inicia o problema com condições iniciais diferentes das consideradas pelo algoritmo.

Conforme as iterações ocorrem, algoritmo de MCL tende a encontrar uma região de maior probabilidade para a posição do robô, eliminando partículas fora dessa região. Quando ocorre um deslocamento sem que os sensores ou o algoritmo estejam funcionando, é provável que não haja partículas próximas à nova posição do robô, de forma que o algoritmo continua considerando um grande número de partículas em local diverso à nova localização.

De forma a contornar tal problema, o AMCL registra duas médias móveis dos pesos das partículas, sendo uma rápida e outra lenta. Se houver uma mudança significativa nos pesos das partículas em um período de tempo curto, o algoritmo insere partículas com posições randômicas no conjunto de partículas considerado pelo algoritmo. Desta forma, o algoritmo tende a manter partículas em regiões de baixa probabilidade sempre que a probabilidade da posição do robô estar representada adequadamente apresentar redução.

As Figuras 1 a 3 apresentam a evolução do MCL e do AMCL quando ocorre o problema do robô sequestrado.

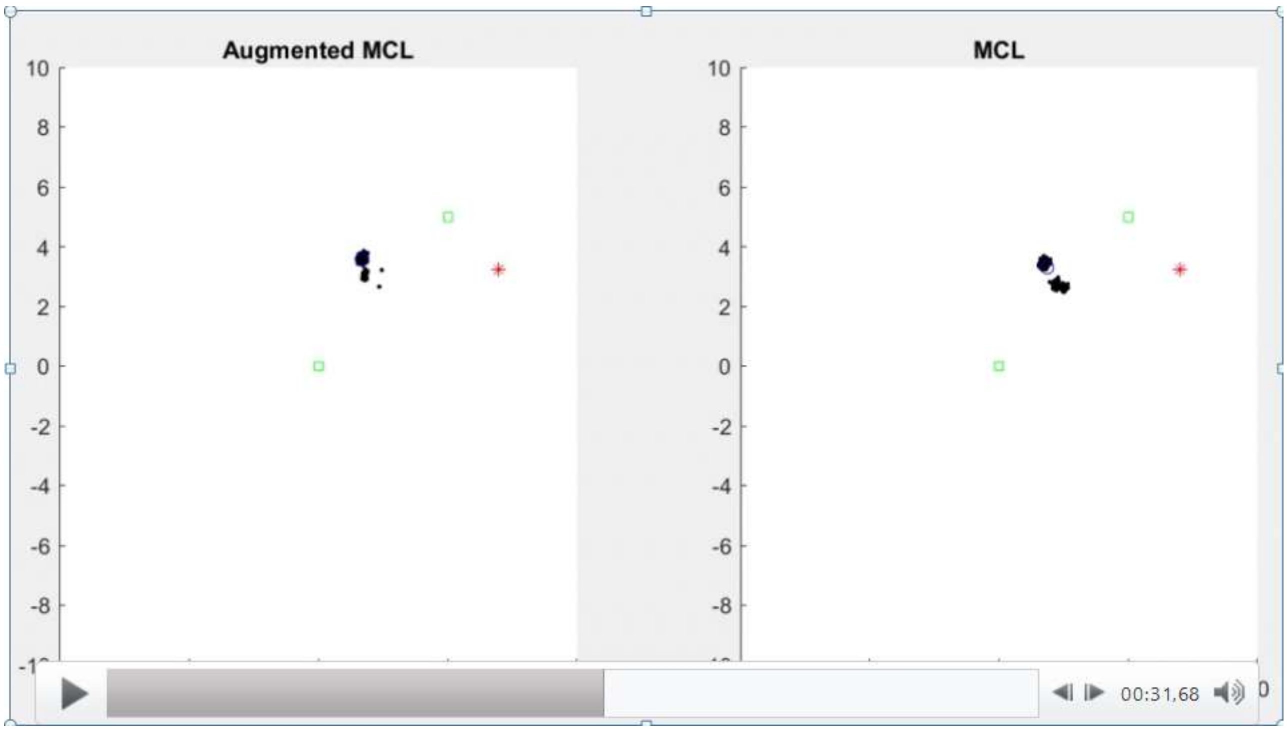

Figura 1: Comparação entre MCL e Augmented MCL. Em verde os landmarks, em vermelho o robô e em preto as partículas.

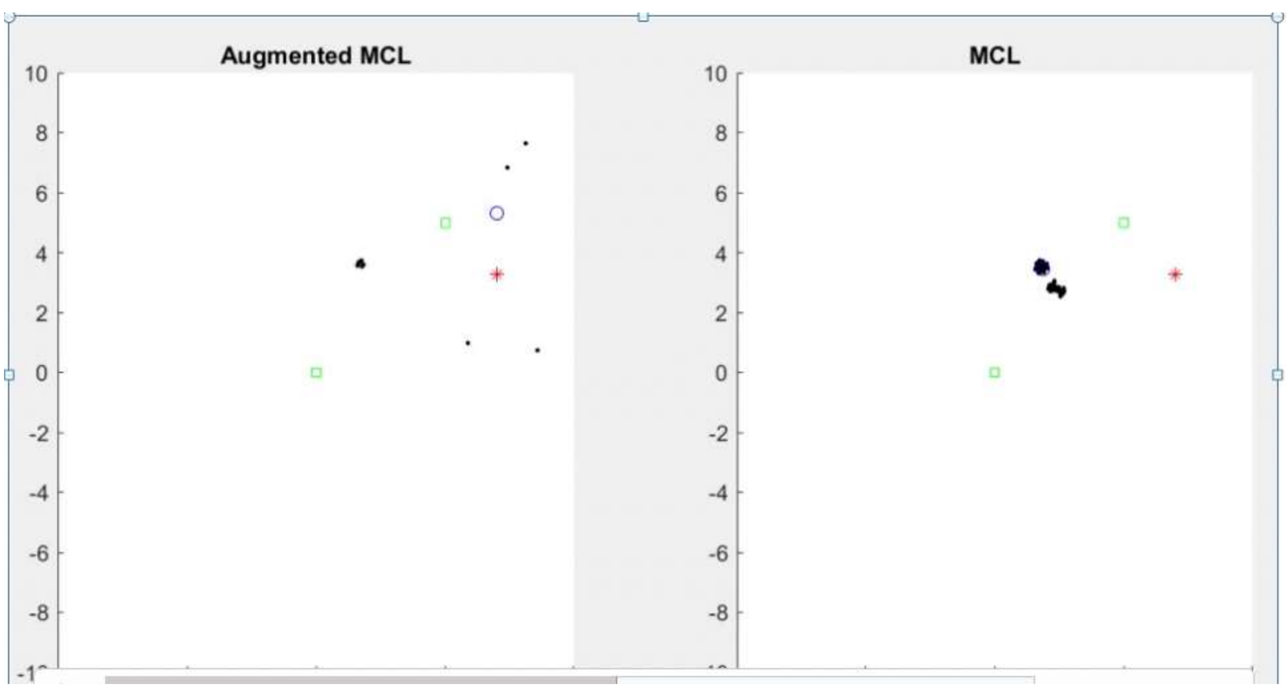

Figura 2: Comparação entre MCL e Augmented MCL. Logo após o robô ser "sequestrado", o AMCL insere partículas de baixa probabilidade de ocorrência no mapa.

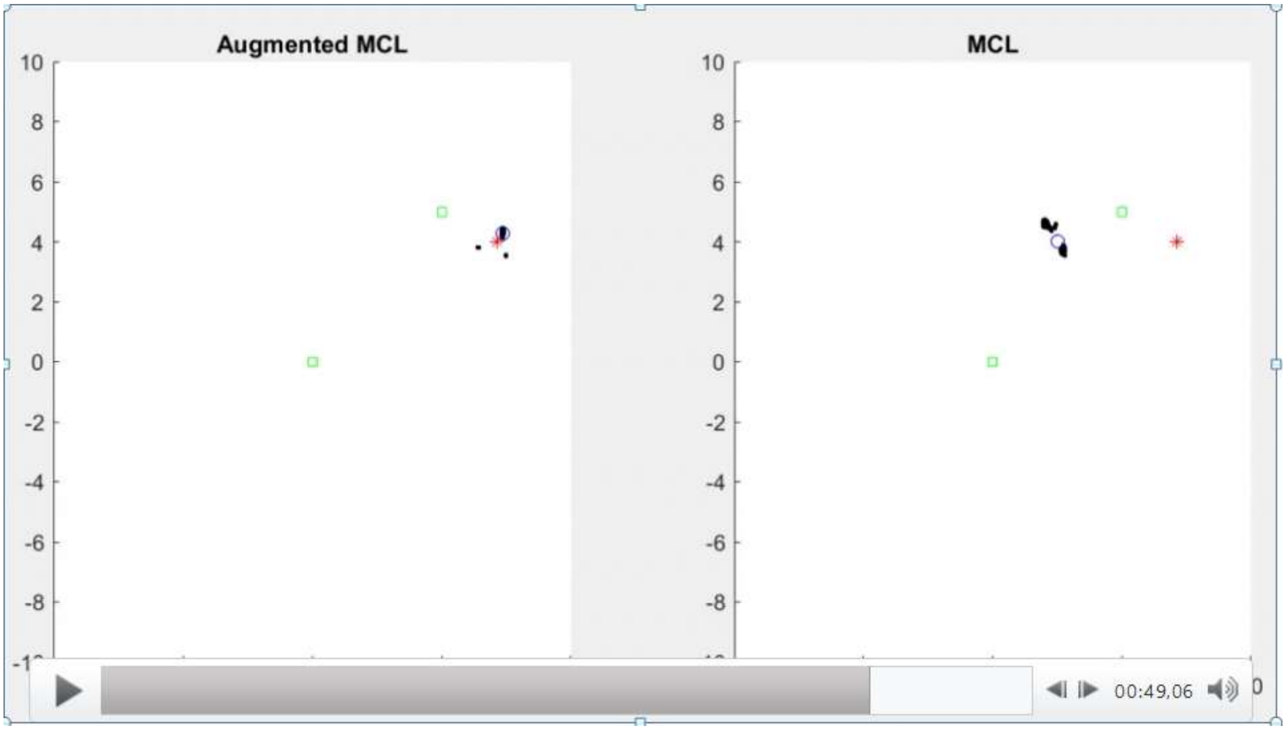

Figura 3: Comparação entre MCL e Augmented MCL. Pouco após o robô ser "sequestrado", o AMCL converge para a região próxima a este.

3. KDL SAMPLING MONTE CARLO LOCALIZATION (KDL SAMPLING MCL)

Algorithm KLD_Sampling_MCL($\mathcal{X}_{t-1}, u_t, z_t, m, \varepsilon, \delta$):
    $\mathcal{X}_t = \emptyset$
    $M = 0,\ M_\chi = 0,\ k = 0$
    for all b *in* H *do*
        $b =$ *empty*
    endfor
    do
        draw i *with probability* $\propto w_{t-1}^{[i]}$
        $x_t^{[M]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[i]})$
        $w_t^{[M]} =$ **measurement_model**$(z_t, x_t^{[M]}, m)$
        $\mathcal{X}_t = \mathcal{X}_t + \langle x_t^{[M]}, w_t^{[M]} \rangle$
        if $x_t^{[M]}$ *falls into empty bin* b *then*
            $k = k + 1$
            $b =$ *non-empty*
            if $k > 1$ *then*
                $M_\chi := \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta}\right\}^3$
        endif
        $M = M + 1$
    while $M < M_\chi$ *or* $M < M_{\chi_{min}}$
    return $\mathcal{X}_t$

Tabela 4.1 – Algoritmo KDL Sampling MCL.

O algoritmo de MCL necessita de um grande número de partículas para representar adequadamente a posição do robô, utilizando recursos computacionais preciosos em um sistema embarcado. Por outro lado, após sua execução por um determinado tempo, a grande maioria das partículas tende a se concentrar em uma pequena região, próxima à posição real do robô.

Com a finalidade de reduzir a necessidade de recursos computacionais, a Amostragem KLD permite obter, por meio da distribuição das partículas em histograma, um número adequado de partículas a serem utilizadas em função de sua distribuição.

As figuras de 4 a 6 apresentam a redução do número de partículas conforme a posição destas converge para a posição do robô.

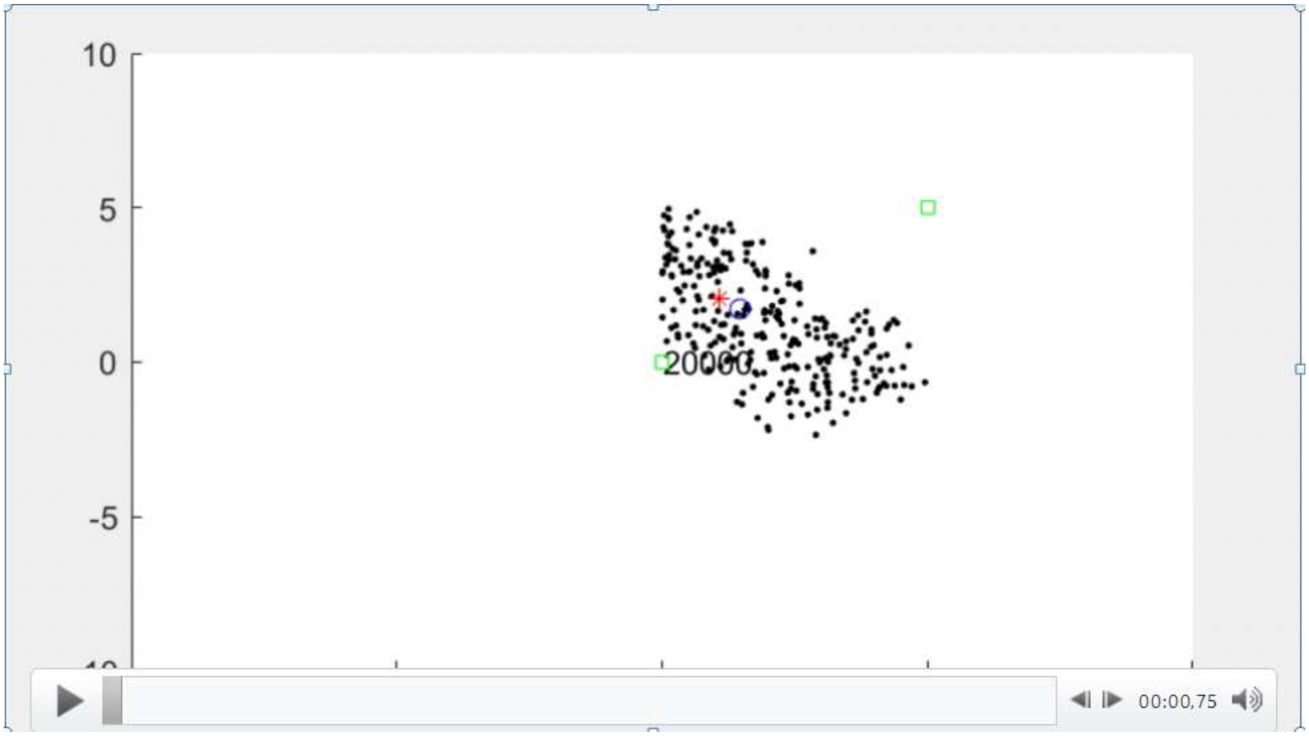

Figura 4: KDL Sampling MCL com número inical de 20.000 partículas.

Figura 5: KDL Sampling MCL com número inical de 20.000 partículas

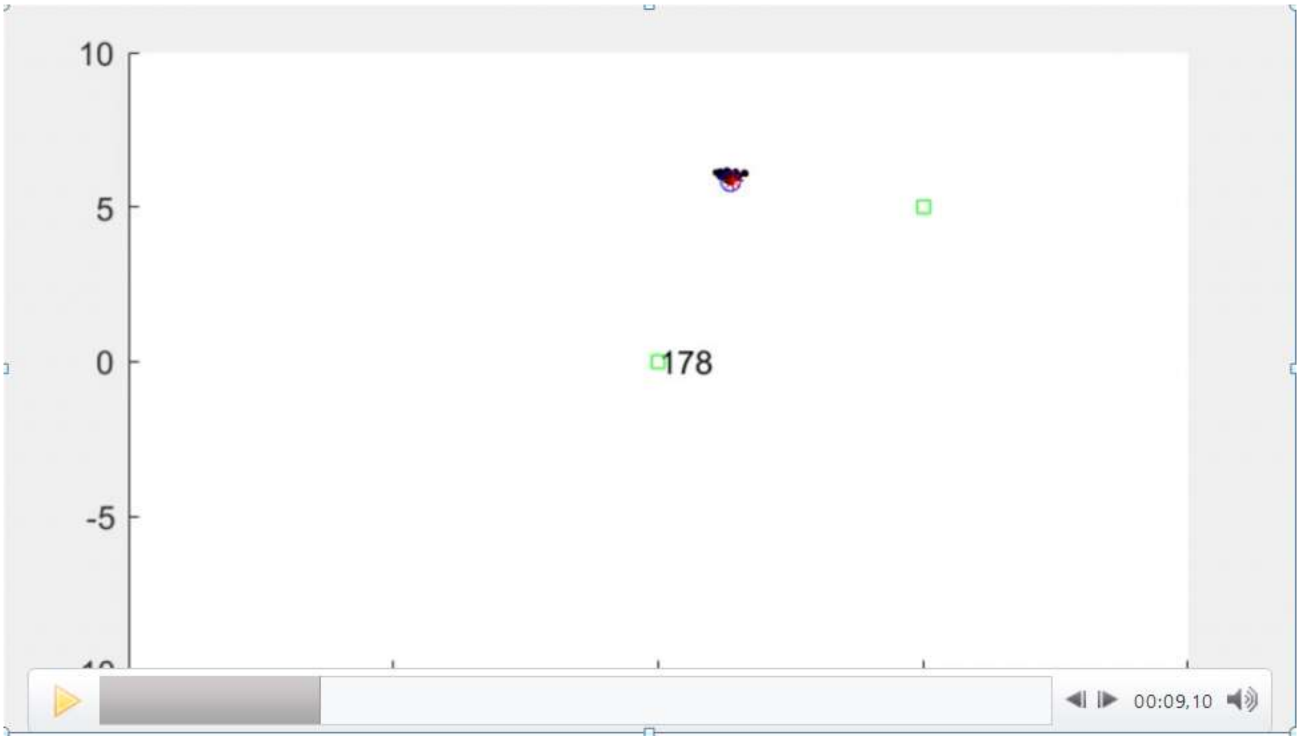

Figura 6: KDL Sampling MCL com número inical de 20.000 partículas. Nota-se a redução significativa do número de partículas conforme o sistema converge para a região onde encontra-se o robô.

4.GRID LOCALIZATION

1: **Algorithm Grid_localization**($\{p_{k,t-1}\}, u_t, z_t, m$):
2: *for all* k *do*
3: $\bar{p}_{k,t} = \sum_i p_{i,t-1}$ **motion_model**$(\text{mean}(\mathbf{x}_k), u_t, \text{mean}(\mathbf{x}_i))$
4: $p_{k,t} = \eta\, \bar{p}_{k,t}$ **measurement_model**$(z_t, \text{mean}(\mathbf{x}_k), m)$
5: *endfor*
6: *return* $\{p_{k,t}\}$

Tabela 1.1 - Algoritmo do Grid Localization.

O Algoritmo de *Grid Localization* aproxima a probabilidade posterior utilizando-se de um filtro de histograma sobre uma decomposição em grade da pose do robô. O filtro discreto de Bayes mantém uma coleção de valores discretos de probabilidade $bel\ (x_t)\ =\ \{p_{k,t}\}$ onde cada probabilidade $p_{k,t}$ é definida sobre uma célula de grade x_k.

A Tabela 1.1 corresponde ao algoritmo do Grid Localization. Ele recebe como entrada os valores de probabilidade discretos $\{P_{k,t-1}\}$, além da medição, controle e o mapa mais recentes. O loop interno itera por todas as células da grade. A linha 3 implementa a atualização do modelo de movimento e a linha 4 a atualização da medição. As probabilidades finais são normalizadas por meio do normalizador η na linha 4. As funções motion_model e measure_model podem ser implementadas por qualquer um dos modelos de movimento no Capítulo 5 e modelos de medição no Capítulo 6, do livro do Thrun.

A Figura 8.1 ilustra a localização da grade em nosso exemplo de corredor unidimensional. Este diagrama é equivalente ao do filtro Bayes geral, exceto pela natureza discreta da representação. Como antes, o robô começa com a incerteza global, representada por um histograma uniforme. Conforme detecta, as células da grade correspondentes têm seus valores de probabilidade associados incrementados. O exemplo destaca a capacidade de representar distribuições multimodais com o *Grid Localization*.

A implementação foi realizada através de adaptações do código [1]. O modelo de movimento utilizado foi o modelo baseado em odometria. E o modelo de medição foi o Beam Range Finder.

A Figura 1.1 ilustra o mapa da simulação, e a trajetória executada pelo robô.

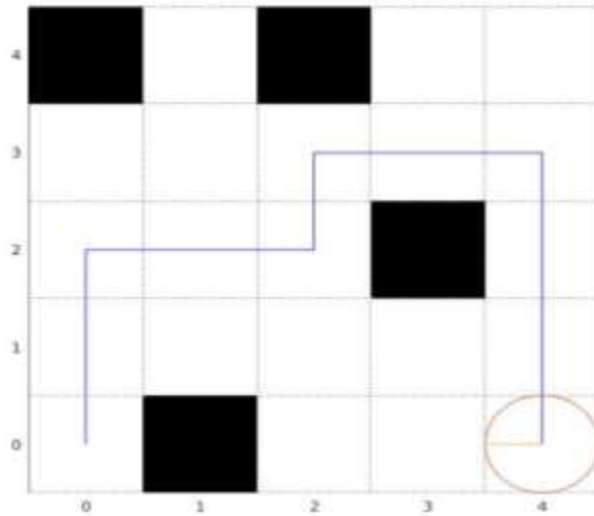

Figura 1.1 – Mapa e Trajetória do Robô

A Figura 1.2 apresenta as probabilidades iniciais de pose para cada ladrilho do mapa. A implementação inicia com uma distribuição uniforme de probabilidade.

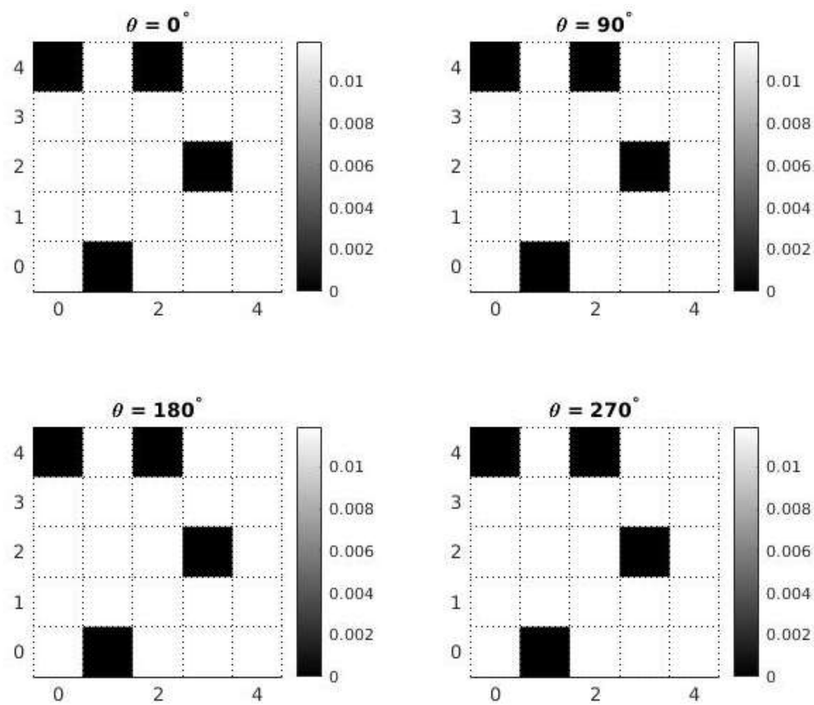

Figura 1.2 – Probabilidades Iniciais do Grid.

A Figura 1.3 mostra a probabilidade de pose atualizada, para cada ladrilho, após a execução de todos os movimentos do robô. Quanto mais clara a cor do ladrilho, maior a probabilidade do robô estar localizado nele.

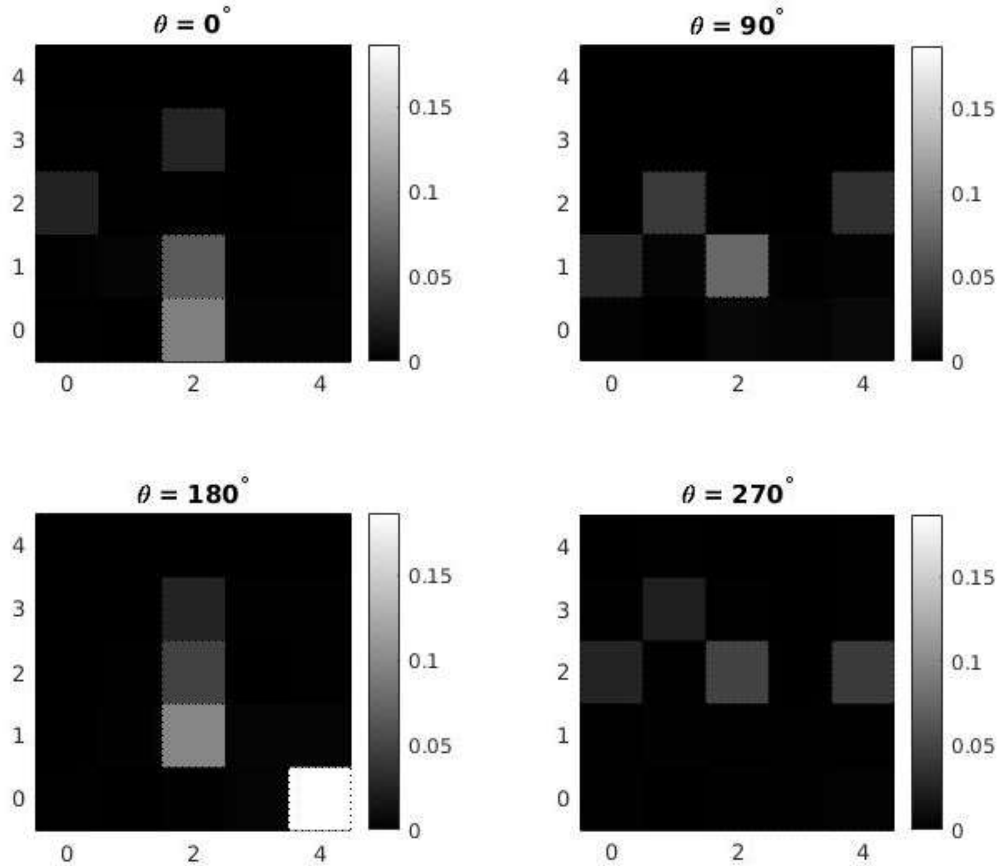

Figura 1.3 – Probabilidades Finais do Grid.

1. TEST RANGE MEASUREMENT

Algorithm test_range_measurement($z_t^k, \bar{\mathcal{X}}_t, m$):

$p = q = 0$

for $m = 1$ *to* M *do*

$p = p + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t^{[m]}, m)$

$q = q + z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t^{[m]}, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t^{[m]}, m)$
$+ z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t^{[m]}, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t^{[m]}, m)$

endfor

if $p/q \leq \chi$ *then*

return accept

else

return reject

endif

Tabela 5.1 – Algoritmo Test Range Measurement.

Este algoritmo lida com ambientes dinâmicos, que incluem situações em que a presença de pessoas pode afetar as medições. A ideia é investigar a causa de uma medição de sensor e rejeitar aqueles que podem ser afetados por dinâmicas ambientais não modeladas. Conforme observado no modelo Beam Range Finder, o termo que envolve z_{short} e p_{short} corresponde a objetos inesperados. As medições que caracterizam objetos inesperados são então rejeitadas.

A Tabela 5.1 descreve o algoritmo Test Range Measurement no contexto de filtros de partículas. Ele recebe como entrada um conjunto de partículas $\bar{X}_t$ representativo da crença $\overline{bel}\,(x_t)$, junto com uma medição de alcance z_t^k e um mapa. Retorna “rejeitar” se com probabilidade maior que χ a medida corresponder a um objeto inesperado; caso contrário, retorna "aceitar".

Como regra geral, a rejeição de medidas atípicas geralmente é uma boa ideia. Quase não existem ambientes estáticos; mesmo em ambientes de escritório, os móveis são movidos, as portas são abertas / fechadas.

Na implementação fizemos uma adaptação do modelo Beam Range Finder, utilizando a lógica do Test Measurement aplicada à partículas.

A Figura 5.1 ilustra os resultados obtidos. A figura 5.1a apresenta o mapa e a posição real do robô. A Figura 5.1b mostra o conjunto de partículas representando a crença da posição do robô. A Figura 5.1c ilustra as partículas após o uso do filtro, onde as partículas azuis foram rejeitadas e as partículas vermelhas foram aceitas. A Figura 5.1d representa apenas o conjunto com as partículas aceitas.

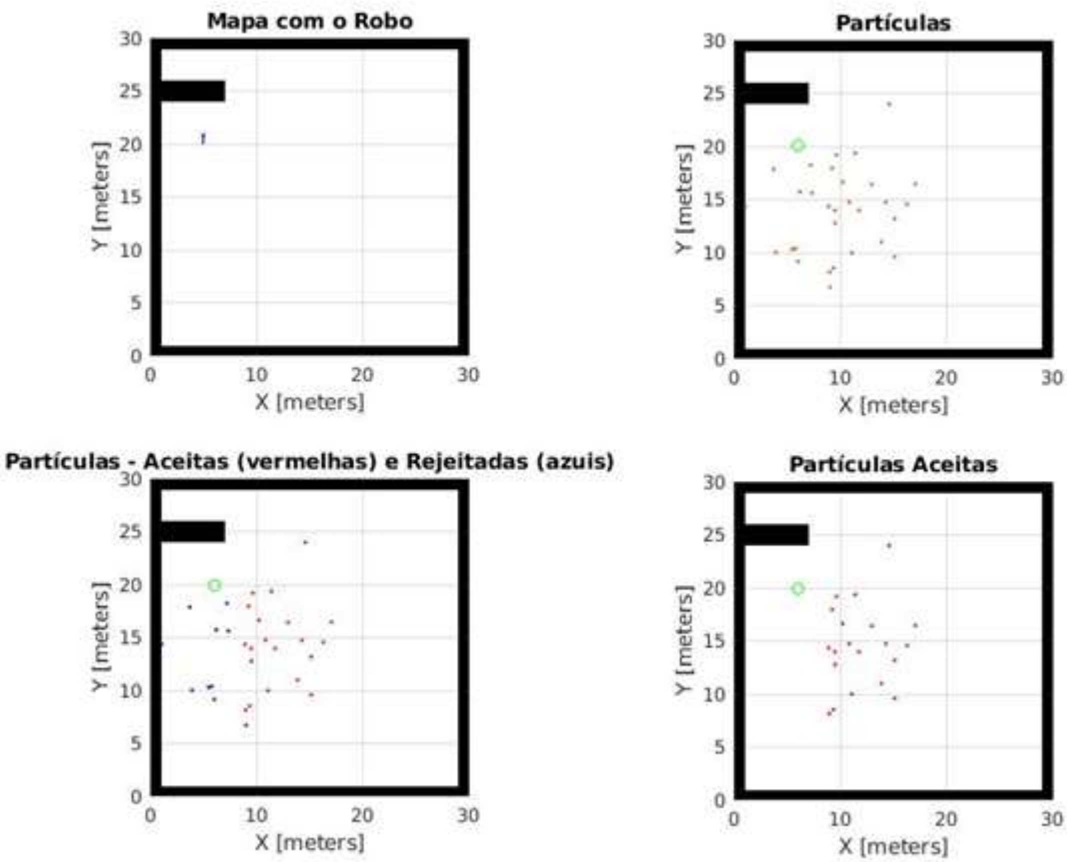

Figura 5.1 – Probabilidades Finais do Grid.

BIBLIOGRAFIA

[1] https://github.com/plusk01/grid-localization.

[2] Dieter Fox , Adapting the Sample Size in Particle Filters Through KLD-Sampling

[3] Thrun et al - Probabilistic Roboticcs