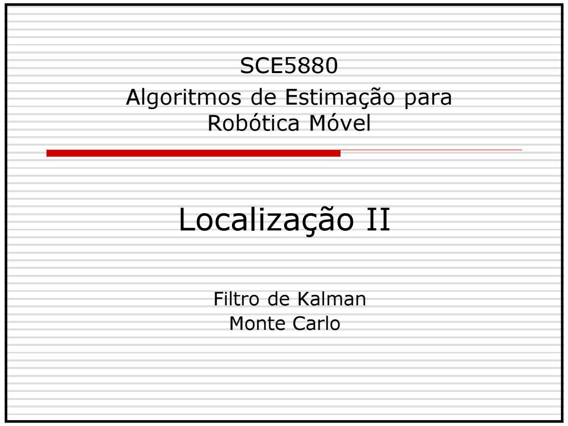
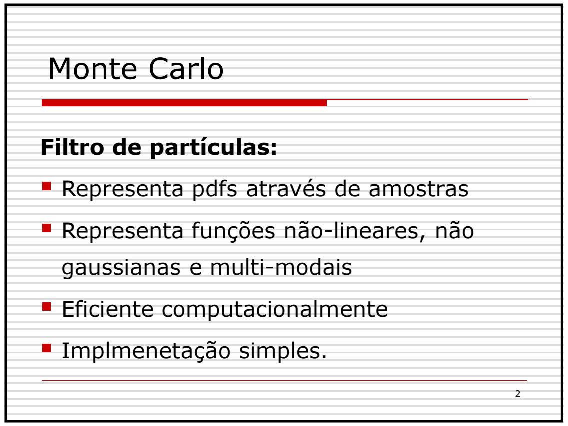


SCE5880
Algoritmos de Estimação para
Robótica Móvel

Localização II

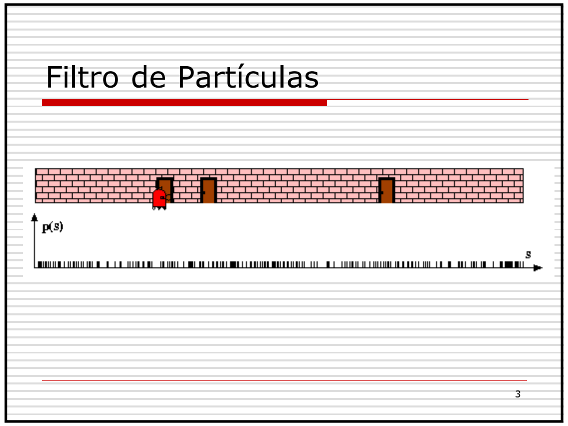
Filtro de Kalman
Monte Carlo



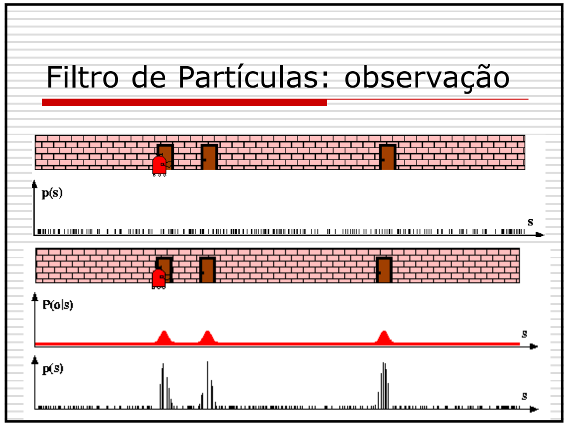
Monte Carlo

Filtro de partículas:

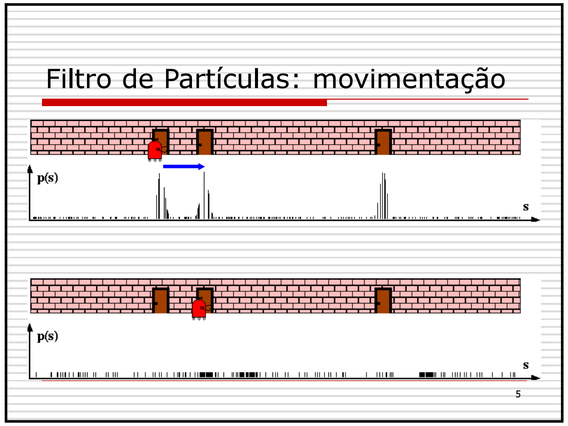
- Representa pdfs através de amostras
- Representa funções não-lineares, não gaussianas e multi-modais
- Eficiente computacionalmente
- Implmenetação simples.



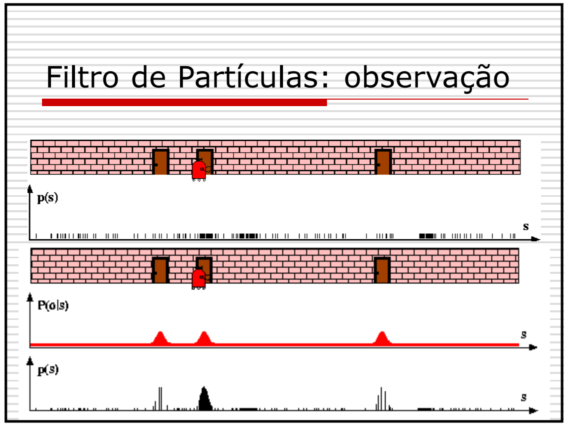
Filtro de Partículas



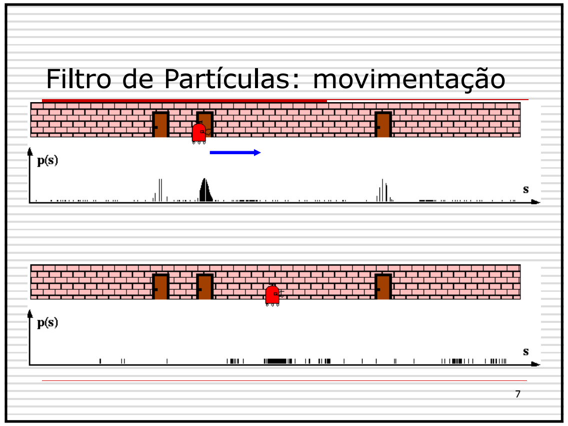
Filtro de Partículas: observação



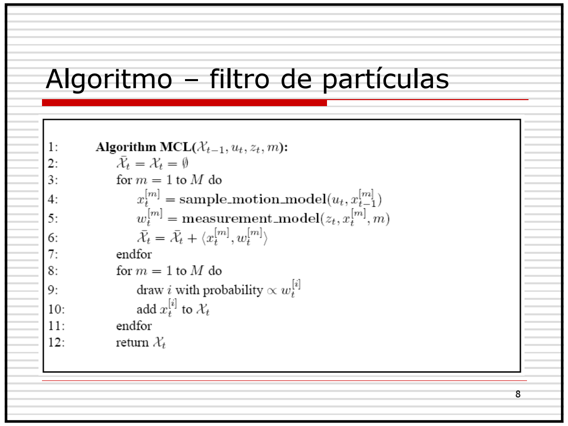
Filtro de Partículas: movimentação



Filtro de Partículas: observação

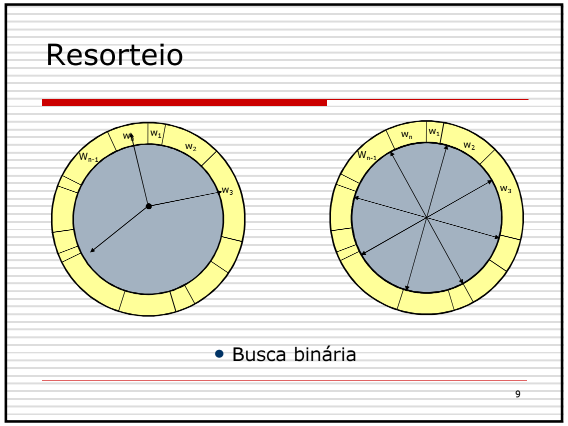


Filtro de Partículas: movimentação



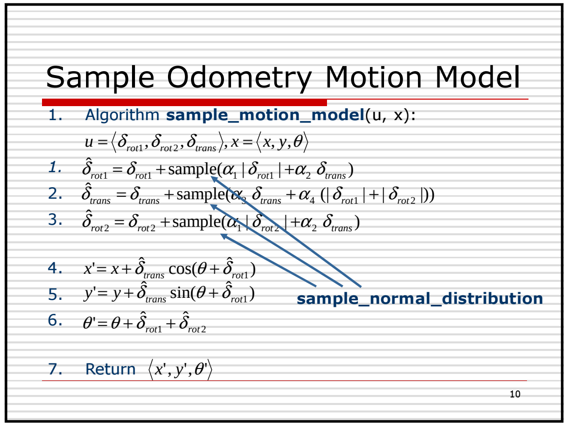
Algoritmo – filtro de partículas

```
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 $\mathcal{X}_t$
```



Resorteio

- Busca binária



Sample Odometry Motion Model

1. Algorithm **sample_motion_model**(u, x):

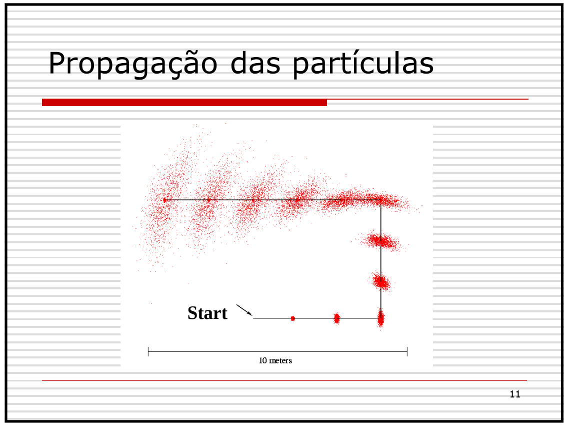
   $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \theta \rangle$

1. $\hat{\delta}_{rot1} = \delta_{rot1} + \text{sample}(\alpha_1 |\delta_{rot1}| + \alpha_2\, \delta_{trans})$
2. $\hat{\delta}_{trans} = \delta_{trans} + \text{sample}(\alpha_3\, \delta_{trans} + \alpha_4\, (|\delta_{rot1}| + |\delta_{rot2}|))$
3. $\hat{\delta}_{rot2} = \delta_{rot2} + \text{sample}(\alpha_1 |\delta_{rot2}| + \alpha_2\, \delta_{trans})$

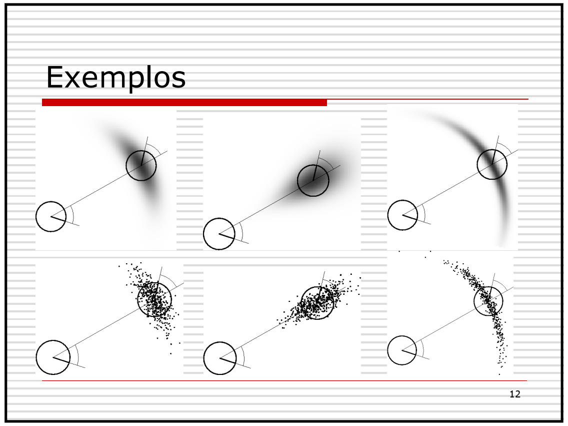
4. $x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
5. $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6. $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

7. Return $\langle x', y', \theta' \rangle$

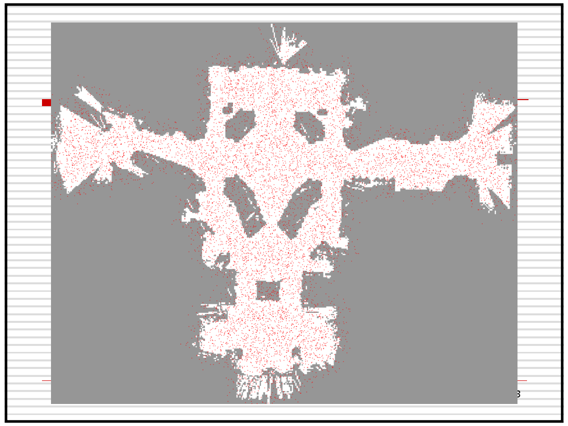
sample_normal_distribution

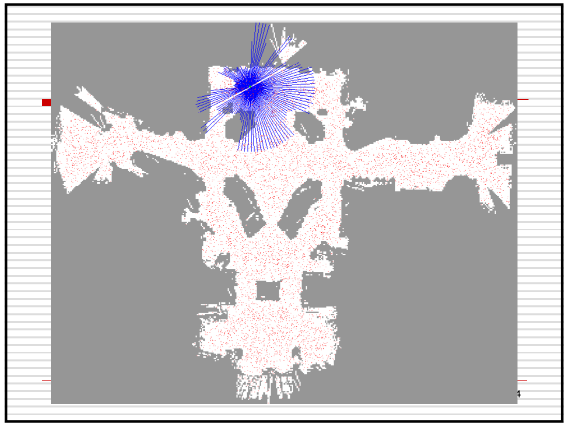


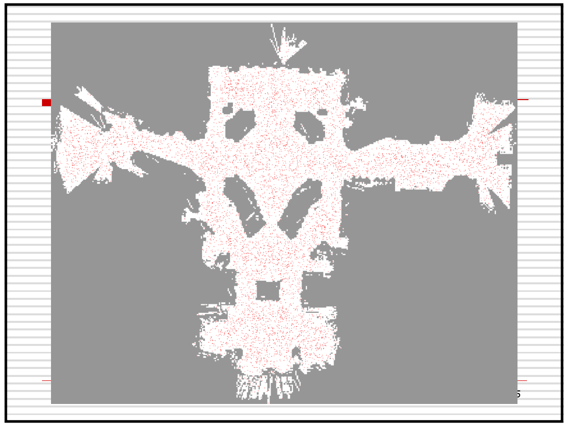
Propagação das partículas

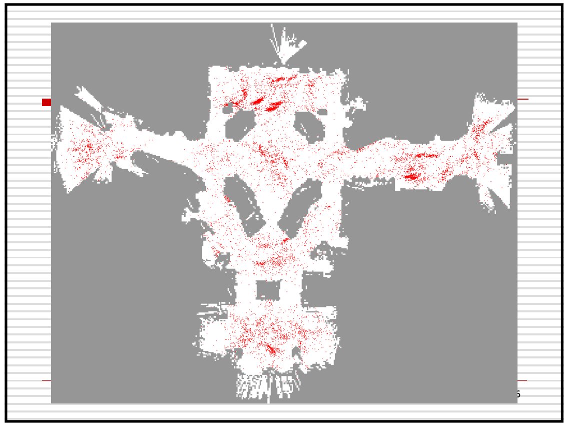


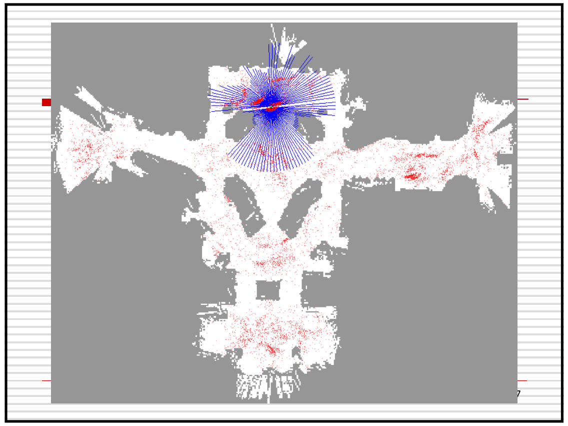
Exemplos

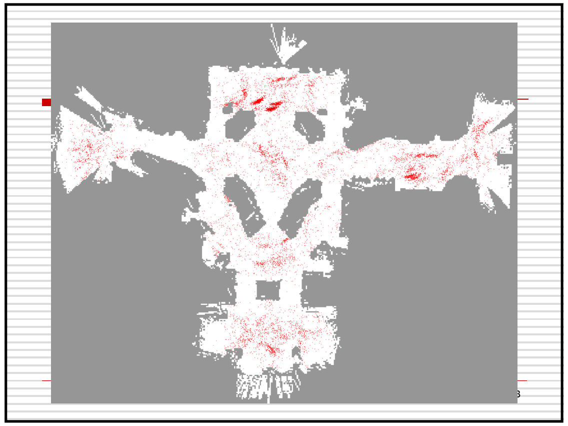

3

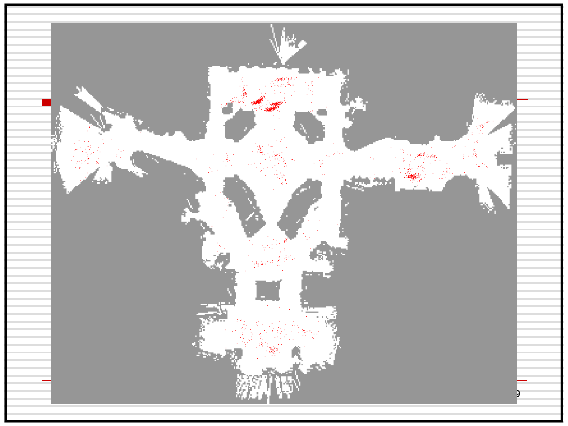

4

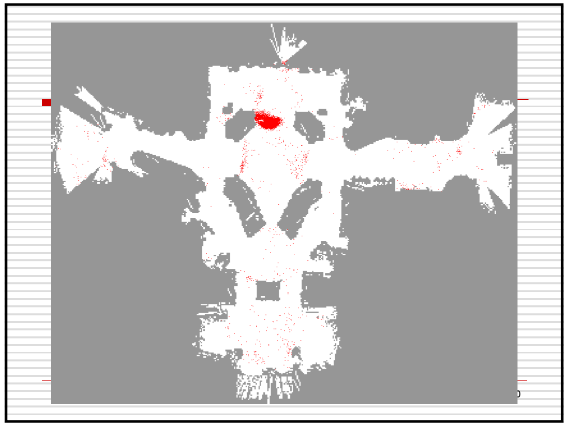

5

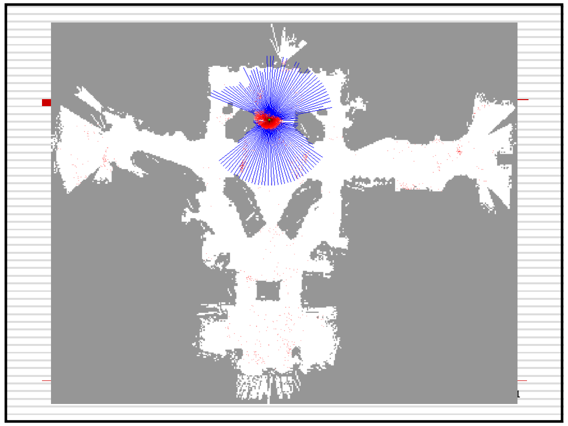

6

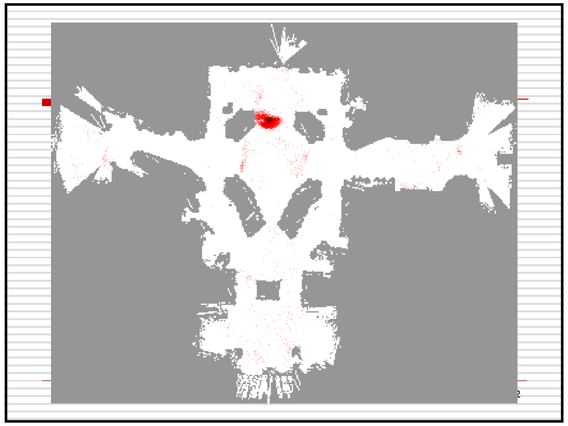

7

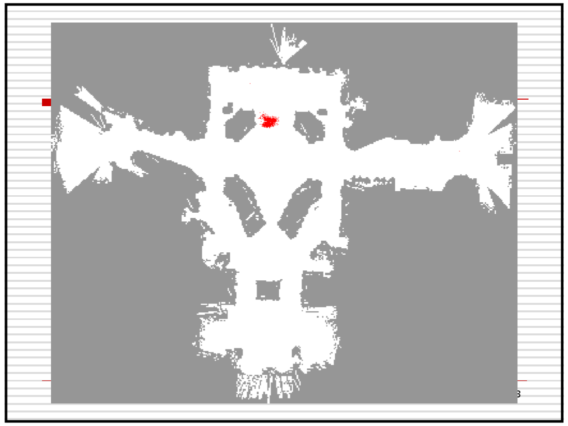

8

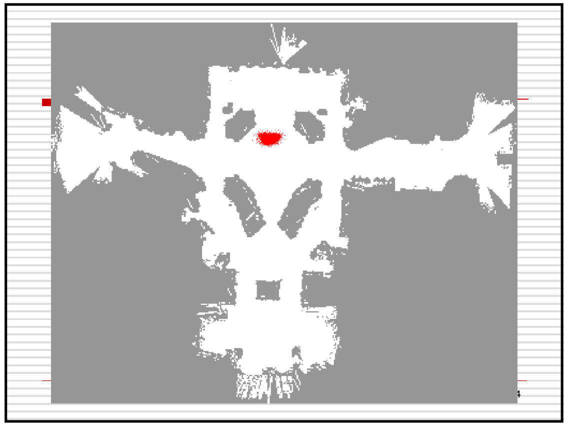

9

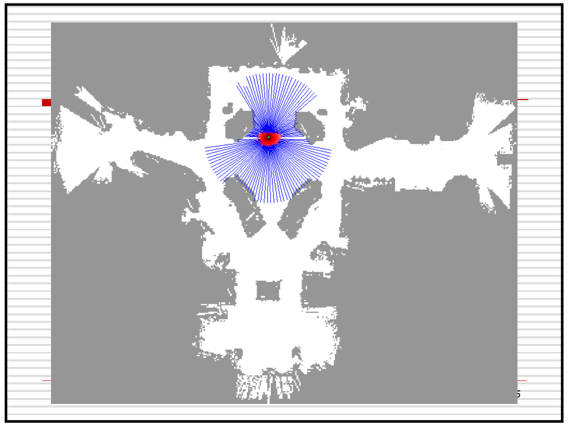

0

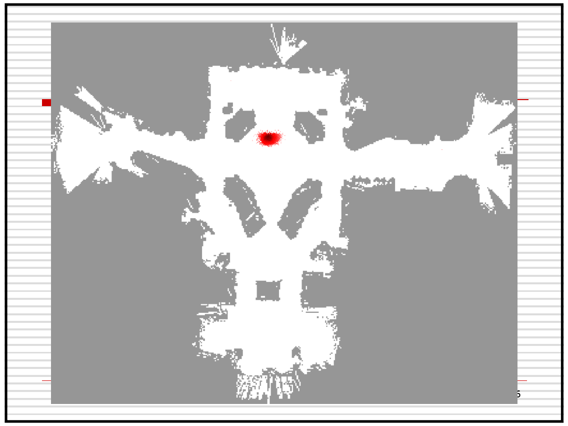

1

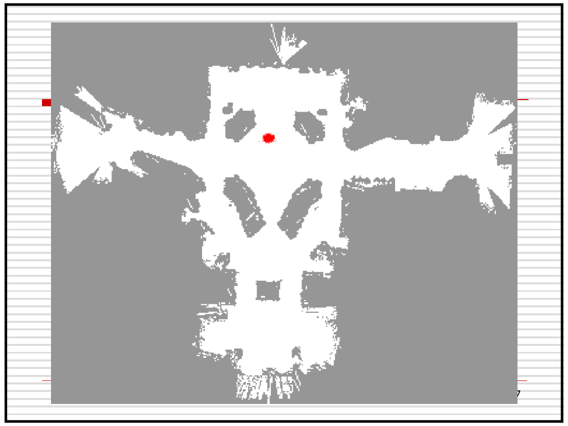

2

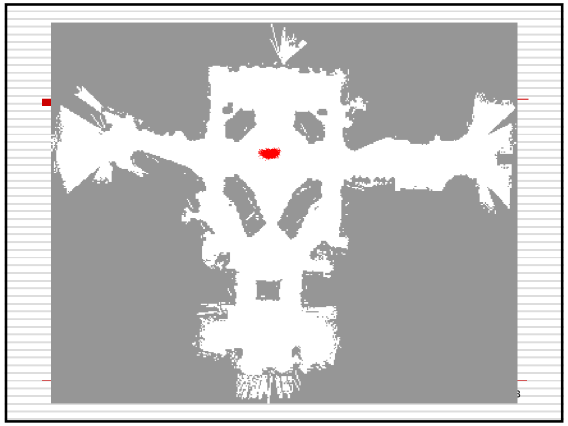

3

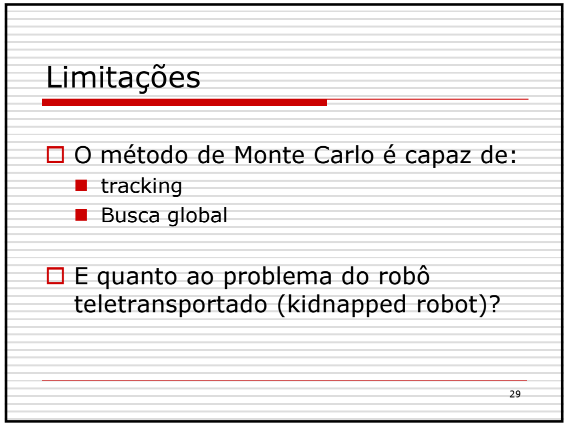

4





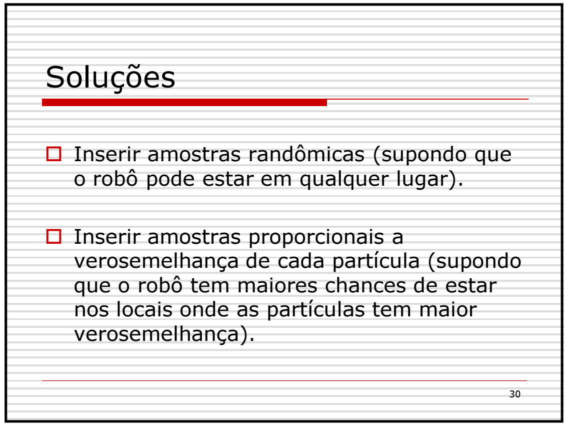






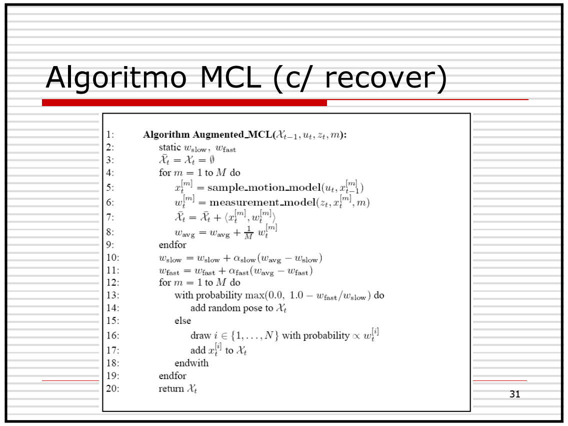
Limitações

- □ O método de Monte Carlo é capaz de:
  - tracking
  - Busca global
- □ E quanto ao problema do robô teletransportado (kidnapped robot)?



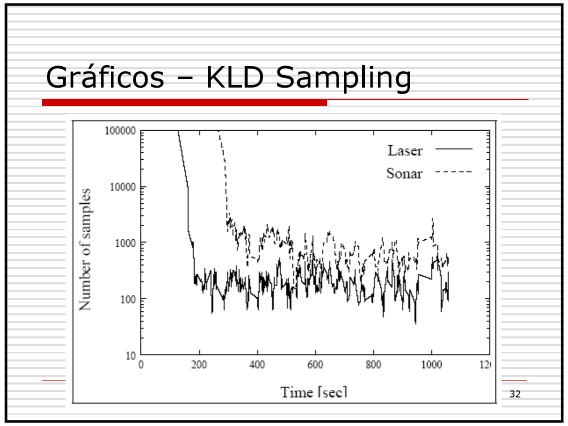
Soluções

- □ Inserir amostras randômicas (supondo que o robô pode estar em qualquer lugar).
- □ Inserir amostras proporcionais a verosemelhança de cada partícula (supondo que o robô tem maiores chances de estar nos locais onde as partículas tem maior verosemelhança).

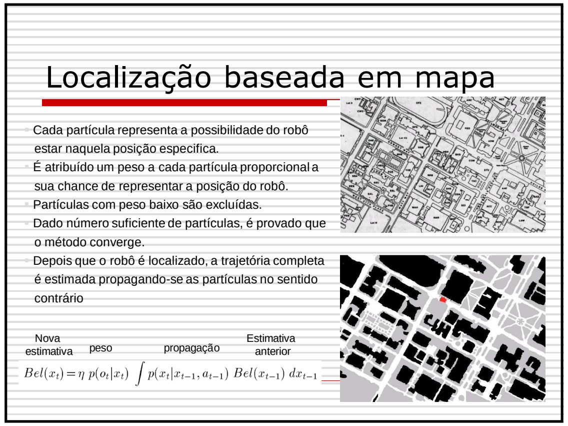


Algoritmo MCL (c/ recover)

```
Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
    static $w_{\rm slow}$, $w_{\rm fast}$
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to $M$ do
        $x_t^{[m]}$ = sample_motion_model($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]}$ = measurement_model($z_t, x_t^{[m]}, m$)
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
        $w_{\rm avg} = w_{\rm avg} + \frac{1}{M}\, w_t^{[m]}$
    endfor
    $w_{\rm slow} = w_{\rm slow} + \alpha_{\rm slow}(w_{\rm avg} - w_{\rm slow})$
    $w_{\rm fast} = w_{\rm fast} + \alpha_{\rm fast}(w_{\rm avg} - w_{\rm fast})$
    for $m = 1$ to $M$ do
        with probability $\max(0.0,\ 1.0 - w_{\rm fast}/w_{\rm slow})$ 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 $\mathcal{X}_t$
```



Gráficos – KLD Sampling

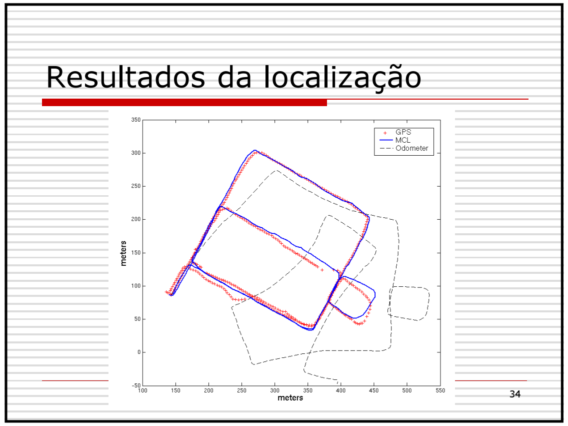


Localização baseada em mapa

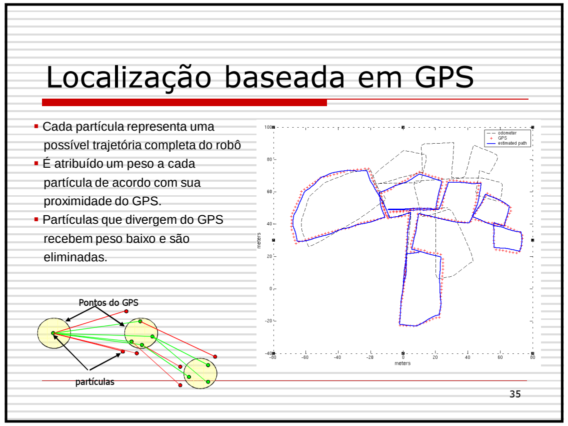
- Cada partícula representa a possibilidade do robô estar naquela posição especifica.
- É atribuído um peso a cada partícula proporcional a sua chance de representar a posição do robô.
- Partículas com peso baixo são excluídas.
- Dado número suficiente de partículas, é provado que o método converge.
- Depois que o robô é localizado, a trajetória completa é estimada propagando-se as partículas no sentido contrário

Nova estimativa — peso — propagação — Estimativa anterior

$$\underbrace{Bel(x_t)}_{\text{Nova estimativa}} = \eta\, \underbrace{p(o_t|x_t)}_{\text{peso}} \int \underbrace{p(x_t|x_{t-1}, a_{t-1})}_{\text{propagação}}\ \underbrace{Bel(x_{t-1})}_{\text{Estimativa anterior}}\ dx_{t-1}$$



Resultados da localização

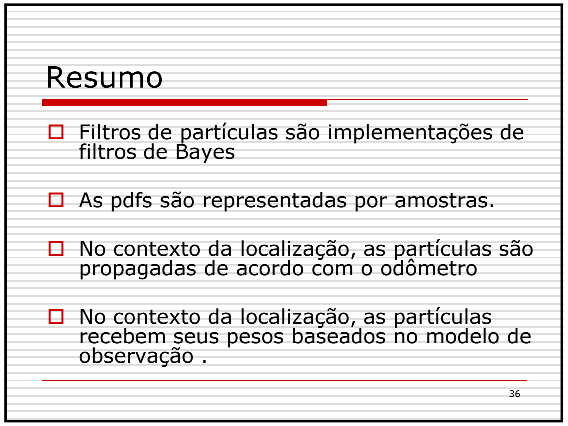


Localização baseada em GPS

- Cada partícula representa uma possível trajetória completa do robô
- É atribuído um peso a cada partícula de acordo com sua proximidade do GPS.
- Partículas que divergem do GPS recebem peso baixo e são eliminadas.

Pontos do GPS

partículas



Resumo

- □ Filtros de partículas são implementações de filtros de Bayes
- □ As pdfs são representadas por amostras.
- □ No contexto da localização, as partículas são propagadas de acordo com o odômetro
- □ No contexto da localização, as partículas recebem seus pesos baseados no modelo de observação .