

# IA368-W

## Métodos Estocásticos em Robótica Móvel

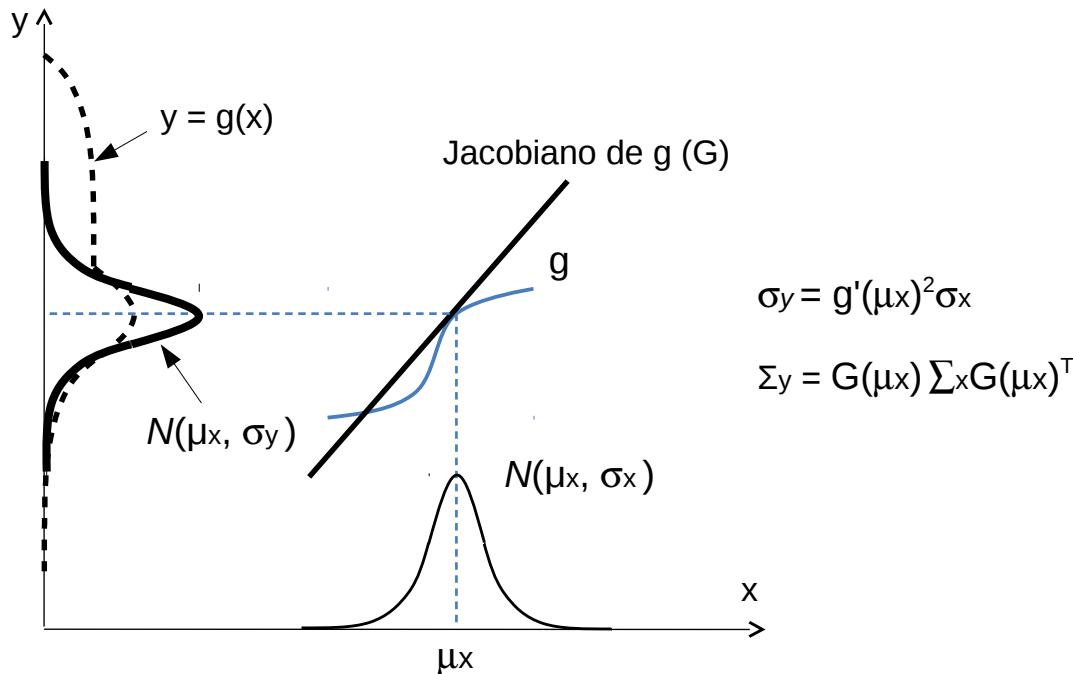
Prof. Eleri Cardozo  
Prof. Eric Rohmer

Colaboradores:  
Leonardo R. Olivi  
Paulo G. Pinheiro  
Ricardo S. Souza  
Fernando C.A. Pinho

## LOCALIZAÇÃO ROBÓTICA: FILTRO DE KALMAN UNSCENTED E FILTRO DE INFORMAÇÃO

# Filtro de Kalman Estendido

Vimos que o Filtro de Kalman Estendido (EKF) utiliza Jacobianos para linearizar o modelo do sistema e a relação estado-sensores.

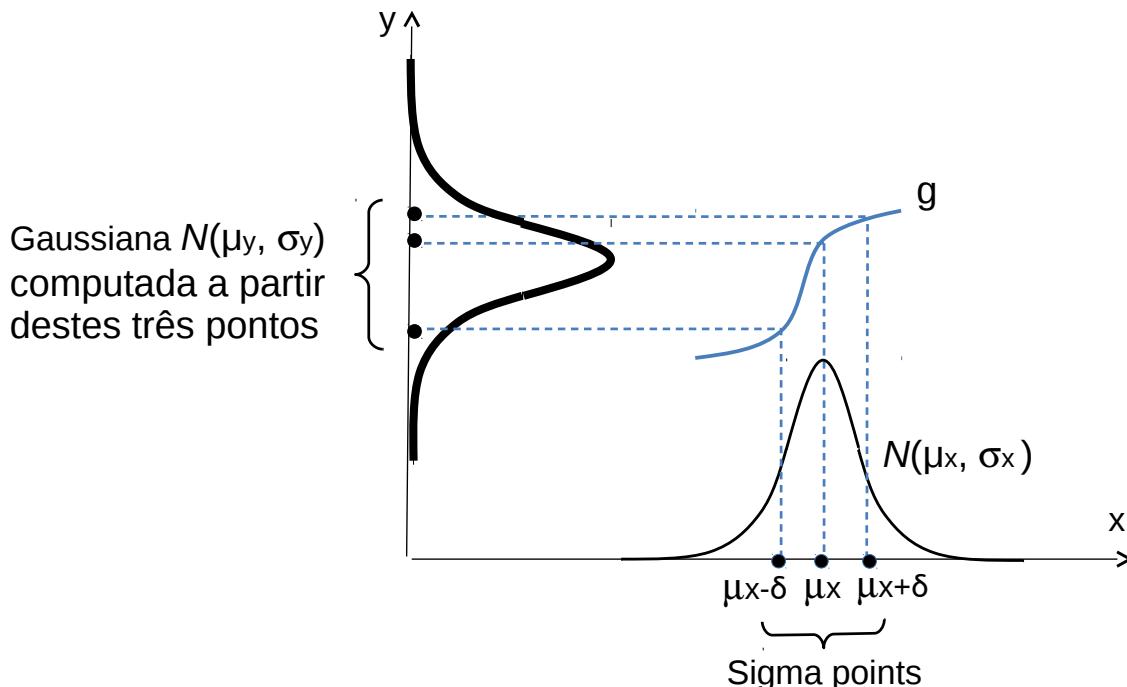


IA368W - Prof. Eleri Cardozo

3

# Filtro de Kalman Unscented

O Filtro de Kalman Unscented (UKF) utiliza linearização via sigma points:



IA368W - Prof. Eleri Cardozo

4

# Transformações Unscented

Seja uma variável aleatória com distribuição normal e dimensão  $n$ , por exemplo a pose do robô ( $n = 3$ ), com média  $\mu$  e covariância  $\Sigma$ . Para cada dimensão são computados 2 sigma points mais um referente à média  $\mu$ , totalizando  $2n + 1$  sigma points.

Os sigma points são computados da seguinte forma:

$$\chi[0] = \mu$$

$$\chi[i] = \mu + \gamma\sqrt{\Sigma}, \quad i = 1, \dots, n$$

$$\chi[i] = \mu - \gamma\sqrt{\Sigma}, \quad i = n + 1, \dots, 2n$$

# Transformações Unscented

Como computar a gaussiana que lineariza  $g$ ?

Primeiramente computa-se dois vetores de pesos,  $w_m$  e  $w_c$ :

$$w_m[0] = \frac{\lambda}{n + \lambda}$$

$$w_c[0] = \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta)$$

$$w_m[i] = w_c[i] = \frac{1}{2(n + \lambda)}, \quad n = 1, \dots, 2n$$

# Transformações Unscented

A partir dos pesos,  $w_m$  e  $w_c$  computa-se a média e variância da gaussiana que lineariza  $g$  ( $\mu'$  e  $\Sigma'$ ):

$$\Upsilon[i] = g(\chi[i])$$

$$\mu' = \sum_{i=0}^{2n} w_m[i] * \Upsilon[i]$$

$$\Sigma' = \sum_{i=0}^{2n} w_c[i] * (\Upsilon[i] - \mu')(\Upsilon[i] - \mu')^T$$

# Transformações Unscented

Parâmetros das transformações unscented:

$$\lambda = \alpha^2(n + \kappa) - n$$

$$\gamma = \sqrt{n + \lambda}$$

$\alpha, \kappa$  : determinam a distância dos sigma points em relação à media  $\mu$ .

$\beta$  : ajuste adicional da gaussiana (usualmente 2).

$\sqrt{\Sigma}$  : Fator de Cholesky,  $R | R^T R = \Sigma$ .

# UKF: Algoritmo

$$\chi_{t-1} = [\mu_{t-1} \ \mu_{t-1} \pm \gamma \sqrt{\Sigma}]^T$$

$$\bar{\chi}_t^* = g(u_t, \chi_{t-1})$$

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m[i] \bar{\chi}_t^*[i]$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c[i] (\bar{\chi}_t^*[i] - \bar{\mu}_t) (\bar{\chi}_t^*[i] - \bar{\mu}_t)^T + R_t$$

$$\bar{\chi}_t = [\bar{\mu}_t \ \bar{\mu}_t \pm \gamma \sqrt{\bar{\Sigma}_t}]^T$$

$$\bar{Z}_t = h(\bar{\chi}_t)$$

# UKF: Algoritmo

$$\hat{z}_t = \sum_{i=0}^{2n} w_m[i] \bar{Z}_t[i]$$

$$S_t = \sum_{i=0}^{2n} w_c[i] (\bar{Z}_t[i] - \hat{z}_t) (\bar{Z}_t[i] - \hat{z}_t)^T + Q_t$$

$$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c[i] (\bar{\chi}_t[i] - \bar{\mu}_t) (\bar{\chi}_t[i] - \bar{\mu}_t)^T$$

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

# UKF em Localização Robótica

$$M_t = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \quad \begin{aligned} \sigma_u &= K_v V_t + K_\omega \omega_t \\ \sigma_u &= K_d V_d + K_e V_e \end{aligned}$$

$$Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

$$\Sigma_{t-1}^a = \begin{bmatrix} \Sigma_{t-1} & 0 & 0 \\ 0 & M_t & 0 \\ 0 & 0 & Q_t \end{bmatrix} \quad 7 \times 7$$

# UKF em Localização Robótica

$$\mu_{t-1} = [x_{t-1} \ y_{t-1} \ \theta_{t-1}]^T$$

$$\mu_{t-1}^a = [\mu_{t-1}^T \ 0 \ 0 \ 0 \ 0]^T$$

$$\chi_{t-1}^a = [\chi_{t-1}^x \ \chi_{t-1}^u \ \chi_{t-1}^z]^T = [\mu_{t-1}^a \ \pm \gamma \sqrt{\Sigma_{t-1}^a}]^T$$

$$\bar{\chi}_t^x = g(u_t + \chi_t^u, \chi_{t-1}^x)$$

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m[i] \bar{\chi}_t^x[i]$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c[i] (\bar{\chi}_t^x[i] - \bar{\mu}_t) (\bar{\chi}_t^x[i] - \bar{\mu}_t)^T$$

$$\bar{Z}_t = h(\bar{\chi}_t^x) + \bar{\chi}_t^z$$

# UKF em Localização Robótica

$$\hat{z}_t = \sum_{i=0}^{2n} w_m[i] \bar{Z}_t[i]$$

$$S_t = \sum_{i=0}^{2n} w_c[i] (\bar{Z}_t[i] - \hat{z}_t) (\bar{Z}_t[i] - \hat{z}_t)^T + Q_t$$

$$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c[i] (\bar{\chi}_t[i] - \bar{\mu}_t) (\bar{\chi}_t[i] - \bar{\mu}_t)^T$$

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

# UKF em Localização Robótica

UKF é considerado mais preciso para sistemas com não-linearidades pronunciadas.

A implementação é similar à implementação do EKF.

UKF substitui os Jacobianos por medias ponderadas.

Ao implementar UKF deve-se garantir que a matriz  $\bar{\Sigma}_t$  seja sempre positiva definida.

As mesmas desvantagens do EKF persistem no UKF.

# Filtro de Informação

O Filtro de Informação é similar ao Filtro de Kalman no sentido que emprega gaussianas para modelar incertezas. O Filtro de Informação é obtido reescrevendo-se a expressar de uma PDF normal:

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right\}$$

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left\{-\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu - \frac{1}{2}\mu^T \Sigma^{-1} \mu\right\}$$

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left\{-\frac{1}{2}\mu^T \Sigma^{-1} \mu\right\} \exp\left\{-\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu\right\}$$

$$p(x) = \eta \exp\left\{-\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu\right\}$$

# Filtro de Informação

Definindo-se

$$\Sigma^{-1} = \Omega$$

$$\Sigma^{-1}\mu = \Omega\mu = \xi$$

Temos

$$p(x) = \eta \exp\left\{-\frac{1}{2}x^T \Omega x + x^T \xi\right\}$$

# Filtro de Informação

Considerando uma planta linear:

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

$$z_t = Cx_t + \delta_t$$

O Filtro de Informação é dado por:

$$\bar{\Omega}_t = (A \Omega_{t-1}^{-1} A^T + R_t)^{-1}$$

$$\bar{\xi}_t = \bar{\Omega}_t (A \Omega_{t-1}^{-1} \xi_{t-1} + Bu_t)$$

$$\Omega_t = C^T Q_t^{-1} C + \bar{\Omega}_t$$

$$\xi_t = C^T Q_t^{-1} z_t + \bar{\xi}_t$$

$$x_t = \Omega_t^{-1} \xi_t$$

# Filtro de Informação

Considerando uma planta não linear:

$$\begin{cases} x_t = g(u_t, x_{t-1}) + \epsilon_t \\ z_t = h(x_t) + \delta_t \end{cases}$$

$$\mu_{t-1} = \Omega_{t-1}^{-1} \xi_{t-1}$$

$$\bar{\Omega}_t = (G_t \Omega_{t-1}^{-1} G_t^T + R_t)^{-1}$$

$$\bar{\mu}_t = g(x_{t-1}, u_t)$$

$$\bar{\xi}_t = \bar{\Omega}_t \bar{\mu}_t$$

$$\Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t$$

$$\xi_t = \bar{\xi}_t + H_t^T Q_t^{-1} [z_t - h(\bar{\mu}_t) + H_t \bar{\mu}_t]$$

$$x_t = \Omega_t^{-1} \xi_t$$

# Filtros de Informação em Localização Robótica

Jacobianos G e H e cinemática g conforme definidos anteriormente.

Matriz  $\Omega$  ( $3 \times 3$ ) inicializada com diagonal "infinita" e vetor  $\xi$  ( $3 \times 1$ ) inicializado com zeros.

As matrizes  $\Omega$  e  $\xi$  são computadas aditivamente para cada feature.

Implementação mais simples que Filtros de Kalman.

Requer inversão de matrizes de dimensões  $2 \times 2$  e  $3 \times 3$  apenas.

Base para o algoritmo GraphSLAM.

## Filtros de Informação: Implementação

### Inicialização

Defina as matrizes R e Q:

$$R_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \quad Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

Inicialize a matriz  $\Omega$ , o vetor  $\xi$  e a pose  $\mu$  (GET /motion/pose):

$$\Omega_0 = \begin{bmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{bmatrix} \quad \xi_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \mu_0 = \begin{bmatrix} \mu_{x,0} \\ \mu_{y,0} \\ \mu_{\theta,0} \end{bmatrix}$$

$$\infty = 10^8$$

# Filtro de Informação: Implementação

## Interação t

### 1. Fase de Predição

Leia a pose do robô (GET /motion/pose) e as velocidades das rodas  $V_{dir}$  e  $V_{esq}$  (GET /motion/vel2) :

$$\bar{\mu}_t = \begin{bmatrix} \mu_{x,t} \\ \mu_{y,t} \\ \mu_{\theta,t} \end{bmatrix}$$

Converta  $\mu_{\theta,t}$  para radianos.  
 OBS: Poderíamos estimar  $\mu_t$  a partir do modelo cinemático.

$$\Delta_s = \frac{V_{dir} + V_{esq}}{2} \Delta t \quad \Delta_\theta = \frac{V_{dir} - V_{esq}}{2 * b} \Delta t$$

$b = 165$  mm (robô Pioneer)

# Filtro de Informação: Implementação

Compute o Jacobiano de  $g$  em função de  $\Delta_s$  e  $\Delta_\theta$  :

$$G_t = \begin{bmatrix} 1 & 0 & -\Delta_s * \sin(\mu_{\theta,t-1} + \frac{\Delta_\theta}{2}) \\ 0 & 1 & \Delta_s * \cos(\mu_{\theta,t-1} + \frac{\Delta_\theta}{2}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Omega_t = (G_t(\Omega_{t-1})^{-1} G^T + R_t)^{-1}$$

$$\xi_t = \Omega_t \bar{\mu}_t$$

# Filtro de Informação: Implementação

## 2. Fase de atualização:

Primeiramente detecte as features a partir da pose e da leitura do laser:

$$f = \text{FeatureDetection}(dists, angles, L, Pose)$$

dist --> o que retorna de /perception/laser/1/distances?range=-90:90:1

angles --> [-90 90 1]

L --> Landmarks reais:

L.x = [0, 0, 920, 920, 4190, 4190, 4680, 4680, 4030]; % [mm]

L.y = [0, 2280, 2280, 3200, 3200, 2365, 2365, 650, 0]; % [mm]

Pose --> estrutura p.x, p.y e p.th (th em radianos)

Cada linha da matriz de features f contém:

[xf yf xl yl δb] :

xf, yf: feature detectada (posição x,y)

xl, yl: landmark correspondente (feature real)

δb: angulo entre a feature detectada e o landmark (radianos)

# Filtro de Informação: Implementação

Para cada feature k detectada compute:

$$\delta = [(f(k, 3) - \mu_{x,t}) \quad (f(k, 4) - \mu_{y,t})]$$

$$q = \delta \delta^T$$

$$dx = \delta(1)$$

$$dy = \delta(2)$$

$$H_{k,t} = \frac{1}{q} \begin{bmatrix} -dx\sqrt{q} & dy\sqrt{q} & 0 \\ dy & -dx & -q \end{bmatrix}$$

Jacobiano de h

# Filtro de Informação: Implementação

$$\lambda = \begin{bmatrix} (f(k, 1) - \mu_{x,t}) & (f(k, 2) - \mu_{y,t}) \end{bmatrix}$$

$$r = \lambda \lambda^T$$

$$rx = \lambda(1)$$

$$ry = \lambda(2)$$

$$z_{k,t} = \begin{bmatrix} \sqrt{r} \\ atan2(ry, rx) - \mu_{\theta,t} \end{bmatrix}$$

OBS: normalizar os ângulos de z(2) e h(2).

$$h(\bar{\mu}_t) = \begin{bmatrix} \sqrt{q} \\ atan2(dy, dx) - \mu_{\theta,t} \end{bmatrix}$$

# Filtro de Informação: Implementação

$$\Omega_t = \Omega_t + H_{k,t}^T Q_t^{-1} H_{k,t}$$

$$\xi_t = \xi_t + H_{k,t}^T Q_t^{-1} [z_{k,t} - h(\bar{\mu}_t) + H_{k,t} \bar{\mu}_t]$$

Fim do processamento de features.

Fim do ciclo t. Calcule a nova pose do robô:  $\mu_t = \Omega_t^{-1} \xi_t$

E atue o robô com a diferença  $\Delta_\mu$  (POST /motion/pose):  $\Delta\mu = \mu_t - \bar{\mu}_t$

Espere  $\Delta t$  para iniciar um novo ciclo - use `tic()` e `toc()` - e inicie o ciclo t+1.

# Filtro de Informação: Implementação

Todas as operações com angulos devem ser normalizadas, por exemplo:

```
Zkt(2) = NormAngle(Zkt(2))
hmu(2) = NormAngle(hmu(2))
Csi(3) = NormAngle(Csi(3))
```

```
function nangle = NormAngle(angle)
    angle = mod(angle, 2*pi);
    if angle > pi
        angle = angle - 2*pi;
    elseif angle < -pi
        angle = angle + 2*pi;
    end
    nangle = angle;
end
```

## LOCALIZAÇÃO COM FILTRO DE PARTÍCULAS

# Filtro de Partículas

O Filtro de Partículas pertence à classe dos métodos estocásticos designados como "Monte Carlo".

São métodos que utilizam amostragem para estabelecer o valor de uma grandeza que não dispomos uma forma fechada (expressão matemática) para expressá-la.

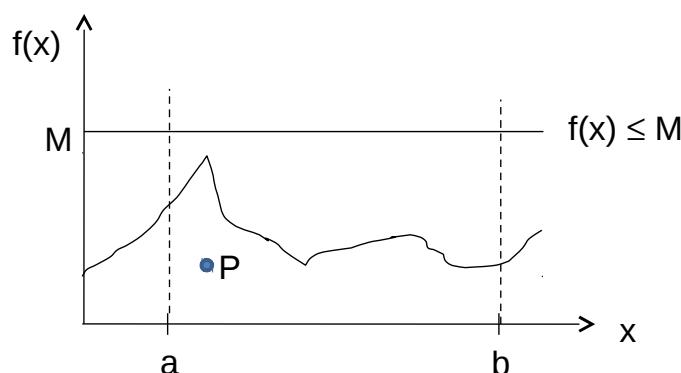
O Filtro de Partículas não utiliza linearizações nem requer que as incertezas sejam modeladas por distribuições normais.

Por exemplo, seja  $f$  uma função complicada que precisamos calcular sua integral no intervalo  $[a b]$ .

IA368W - Prof. Eleri Cardozo

29

# Filtro de Partículas



Geramos  $K$  pontos  $P = (x, w)$  tal que  $x$  possui distribuição uniforme entre  $[a b]$  e  $w$  possui distribuição uniforme entre  $[0 M]$  (as amostras).

Verificamos quantos pontos satisfazem a condição  $f(x) \geq w$  ( $L$  pontos).

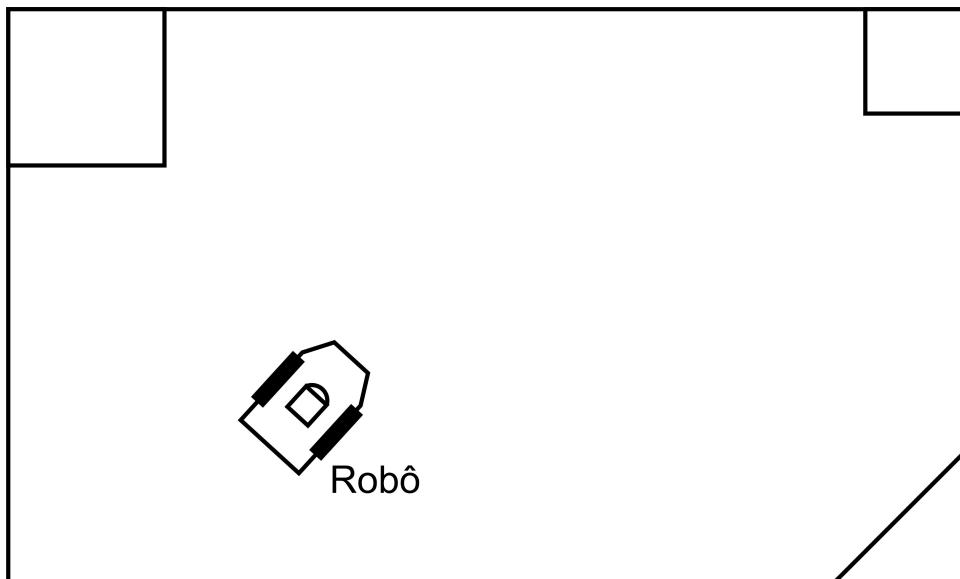
$$\int_a^b f(x)dx \approx \frac{L}{K}(b-a)M$$

IA368W - Prof. Eleri Cardozo

30

# Filtro de Partículas

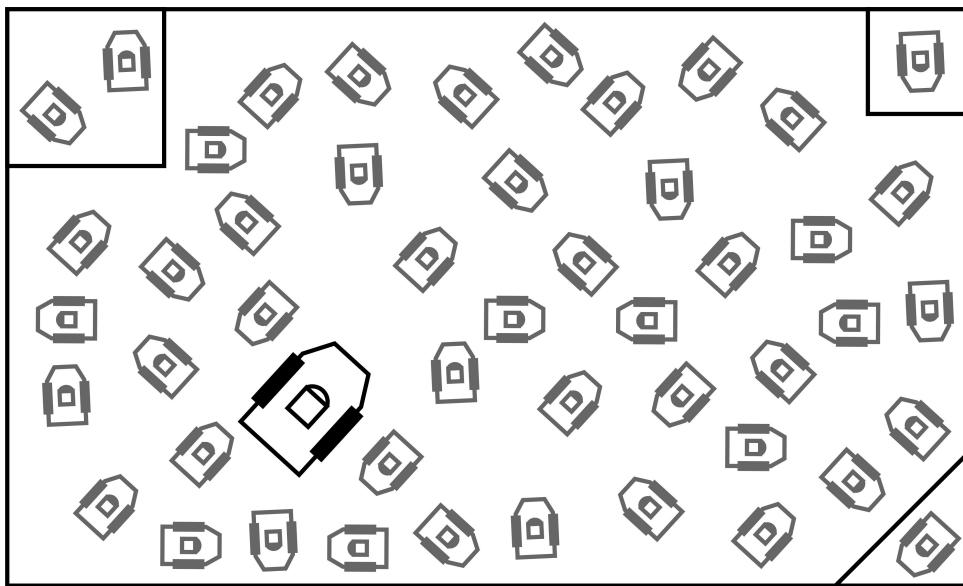
Imagine que um robô é deixado em um local desconhecido de um mapa do qual ele dispõe.



# Filtro de Partículas

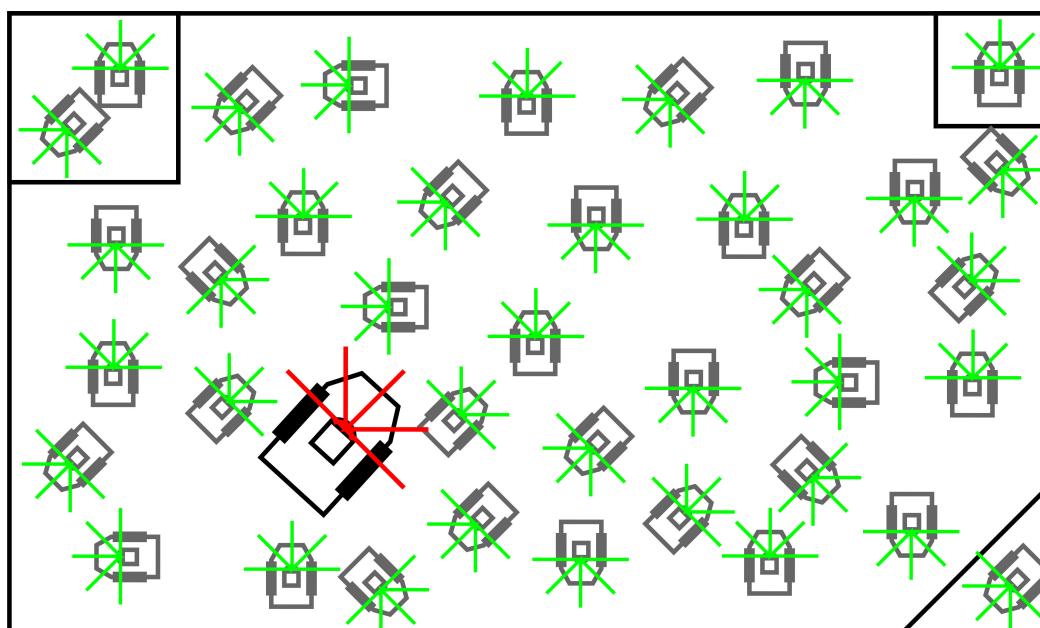
- Como o robô não sabe onde está, ele supõe que pode estar em qualquer lugar do mapa.
- Isto leva a uma distribuição uniforme onde todas as posições do mapa possuem a mesma probabilidade de serem a posição verdadeira.
- Nesta abordagem o mapa não é discretizado, e sim, contínuo.
- Então, o **Filtro de Partículas** espalhará de maneira uniforme uma quantidade **M** de partículas pelo mapa disponibilizado.
- Cada partícula será uma simulação de um robô munido dos mesmos sensores que o robô real.
- A distribuição será uniforme em x, y e  $\theta$  (a orientação também variará entre as partículas).

# Filtro de Partículas



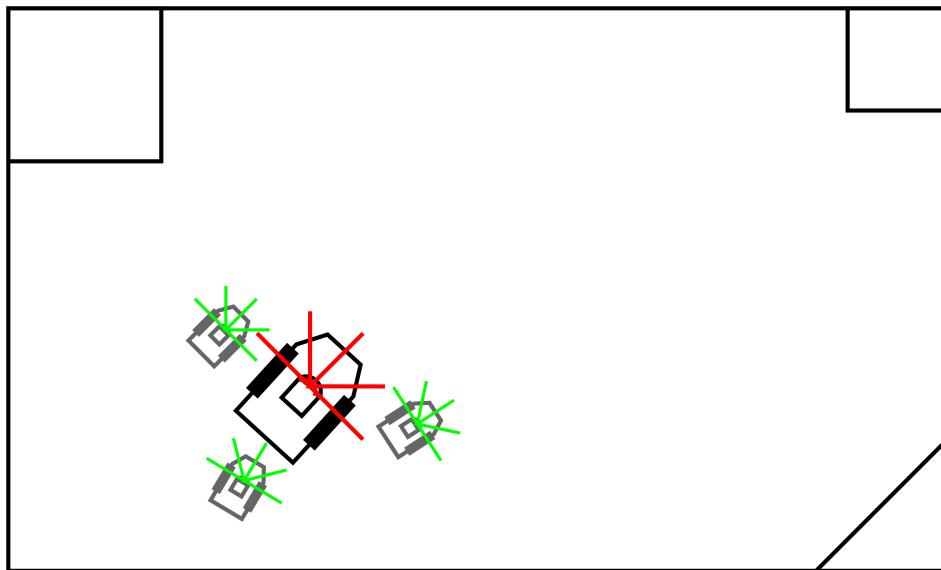
Múltiplas hipóteses de onde e como o robô real pode estar inserido no ambiente. Como saber qual hipótese melhor representa a pose do robô?

# Filtro de Partículas



- O robô real realiza uma medida de seus sensores.
- Todas as partículas farão o mesmo procedimento (simulado).

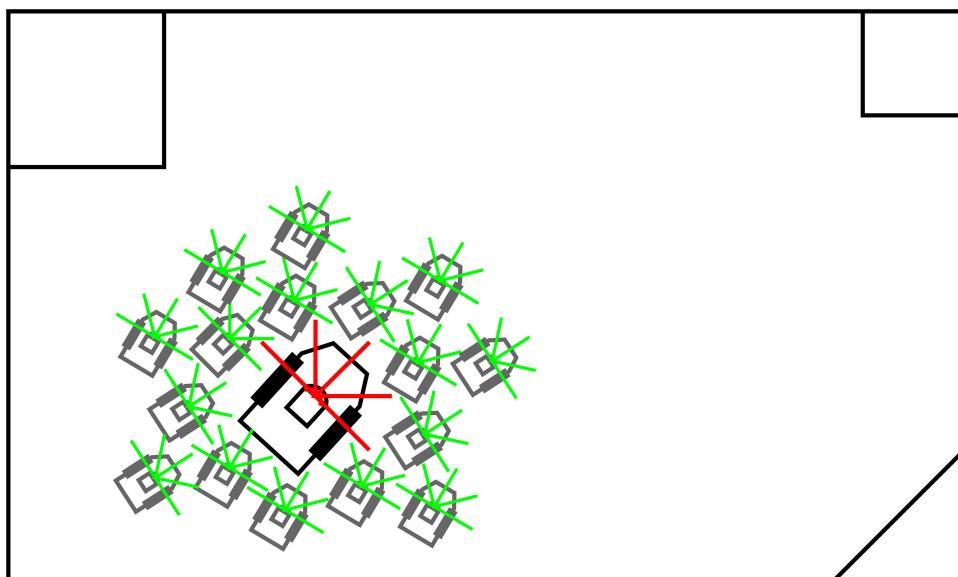
# Filtro de Partículas



Compara-se as medidas dos sensores das partículas com as medidas do sensor real.

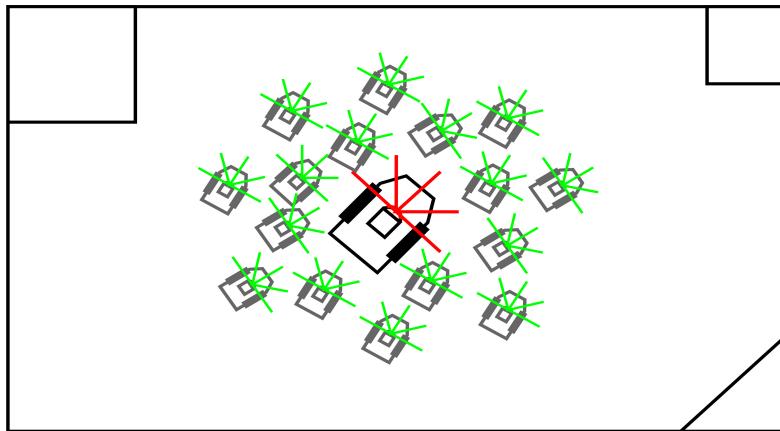
Guarda-se as partículas cujas medidas são similares às do robô real. As demais partículas são descartadas.

# Filtro de Partículas



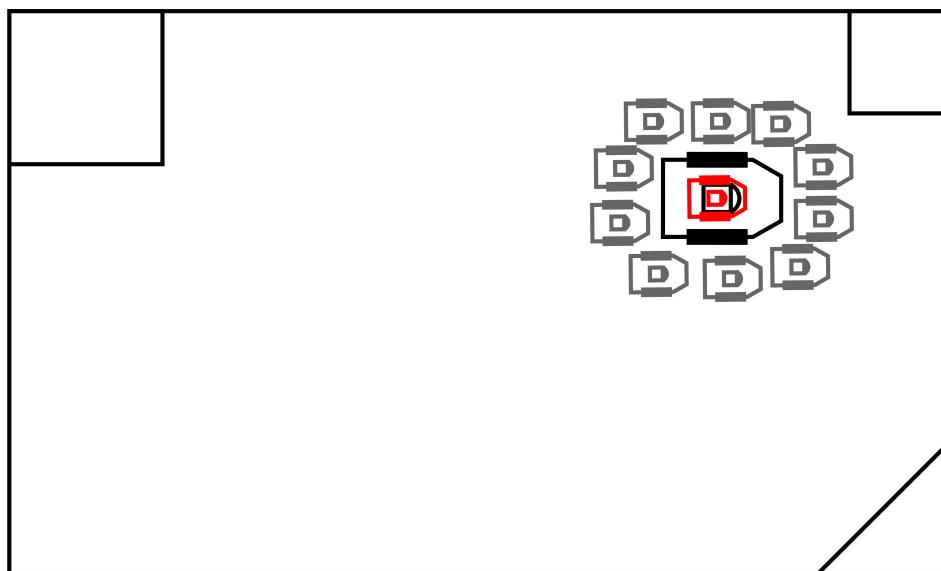
- Como foram eliminadas K partículas, introduz-se novas partículas no lugar das partículas eliminadas (reamostragem).
- As partículas que mais se repetirão são as que possuem medidas mais próximas das medidas do robô real.

# Filtro de Partículas



- Movimenta-se o robô e todas as partículas pelo ambiente de maneira igual e repete-se os passos anteriores de obtenção e comparação de medidas e reprodução das melhores partículas.
- Os ambientes serão circulares, ou seja, durante a movimentação se a partícula sair por um lado do ambiente, ela reaparecerá do outro lado.

# Filtro de Partículas



Repetindo-se estes passos iterativamente, as partículas tenderão a ficar todas com mesma pose, e a média (ponderada) das poses de todas as partículas fornecerá uma boa aproximação da pose do robô real.

# Filtro de Partículas

**Passo 1: Inicialização** - gere P partículas aleatoriamente pelo ambiente (ex: P=2000).

**Passo 2: Sensoriamento** - obtenha as medidas dos sensores de distância do robô e das partículas.

**Passo 3: Ranqueamento** - compute o peso (importância) de cada partícula.

**Passo 4: Seleção** - selecione as N melhores partículas de acordo com os respectivos pesos (ex: N=50).

**Passo 5: Reamostragem** - gere S partículas a partir das partículas N selecionadas (ex: S=200).

**Passo 6: Deslocamento** - desloque o robô e as S partículas com o mesmo movimento imposto ao robô.

**Passo 7: Perturbação** - Acrescente uma perturbação na pose de cada partícula para representar erros de deslocamento e adicione R novas partículas geradas aleatoriamente (ex: R=100). Vá para o passo 2.

# Algoritmo do Filtro de Partículas

## **Passo 2: Sensoriamento**

Para o robô real utilize o recurso de distâncias do laser, por exemplo 7 medidas: `/perception/laser/n/distances?range=-90:90:30`

Para as partículas simule a leitura a partir da pose da partícula e da geometria (mapa) do ambiente (mais adiante).

# Filtro de Partículas

## Passo 3: Ranqueamento

Obtenha o peso de importância  $w$  de cada partícula:

$$w = [1 \ 1 \ \cdots \ 1]_{(n \times 1)}^T$$

para  $i = 1:n$  (partículas)

para  $j = 1:l$  (sensores)

$$gauss = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{1}{2} \eta \left(\frac{sensor_{partícula}(i, j) - sensor_{real}(j)}{\sigma}\right)^2\right)$$

$$w(i) = gauss * w(i)$$

fim

$\sigma$  é o desvio padrão do sensor  
(distância apenas).

$\eta$  é um fator que evita  $\exp \rightarrow 0$

$$w = \frac{w}{\sum_{i=1}^n w(i)} \quad (\text{normalização})$$

IA368W - Prof. Eleri Cardozo

41

# Filtro de Partículas

## Passo 3: Ranqueamento (alternativa):

Obtém o peso de importância  $w$  de cada partícula

$$w = [1 \ 1 \ \cdots \ 1]_{(n \times 1)}^T$$

para  $i = 1:n$  (partículas)

para  $j = 1:l$  (sensores)

$$dist = \eta (sensor_{partícula}(i, j) - sensor_{real}(j))^2 + dist$$

$$w(i) = \frac{w(i)}{dist}$$

fim

$\eta$  é um fator de normalização  
que evita  $w(i) \rightarrow 0$ .

fim

$$w = \frac{w}{\sum_{i=1}^n w(i)} \quad (\text{normalização})$$

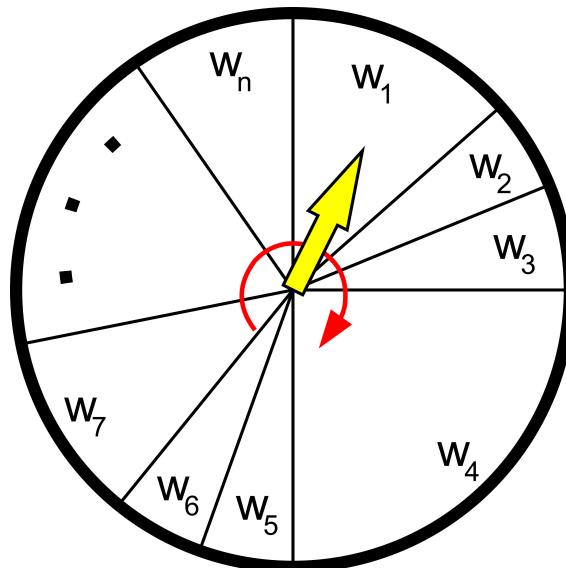
IA368W - Prof. Eleri Cardozo

42

# Filtro de Partículas

## Passo 5: Reamostragem

Utilize o algoritmo da “Roleta” para emular uma roleta aleatória que escolhera uma das  $n$  partículas utilizando as chances dos pesos.



Quanto maior o peso da partícula maior a chance dela ser escolhida para o próximo ciclo.

IA368W - Prof. Eleri Cardozo

43

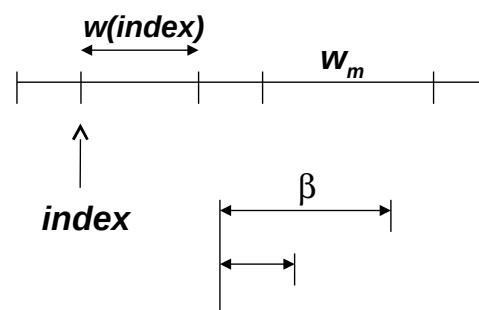
# Filtro de Partículas

**Algoritmo da roleta:** Gerar s novas partículas a partir das  $n$  melhores.

```

 $\beta = 0$ 
 $w_m = \max(w)$ 
(*)  $index = round(rand * n)$ 
para  $i = 1 : s$ 
     $\beta = \beta + 2w_m rand$ 
    enquanto  $\beta > w(index)$ 
         $\beta = \beta - w(index)$ 
    index =  $(index + 1) \bmod n$ 
fim
partículasream(i) = partículas(index) +  $\sigma_{ream} randn$ 
fim

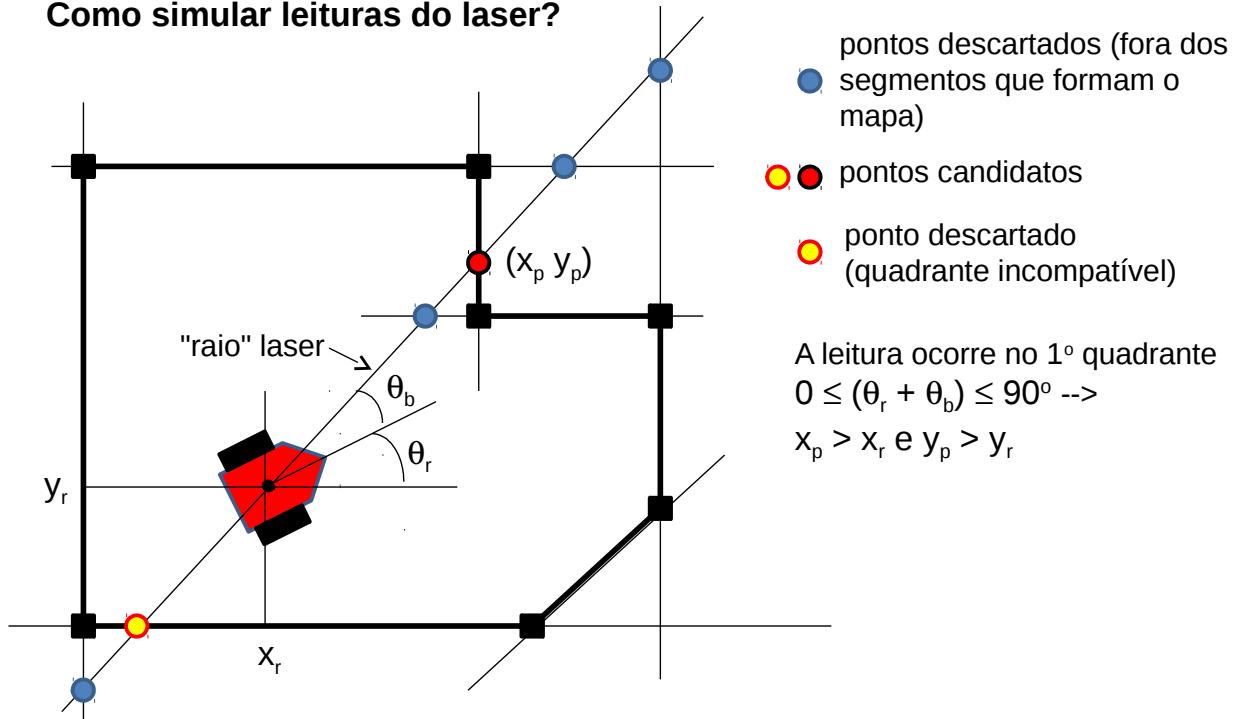
```



Cuidado com índices no Matlab:  
(\*) round pode retornar 0  
(\*\*) mod(n, n) = 0

# Filtro de Partículas

Como simular leituras do laser?

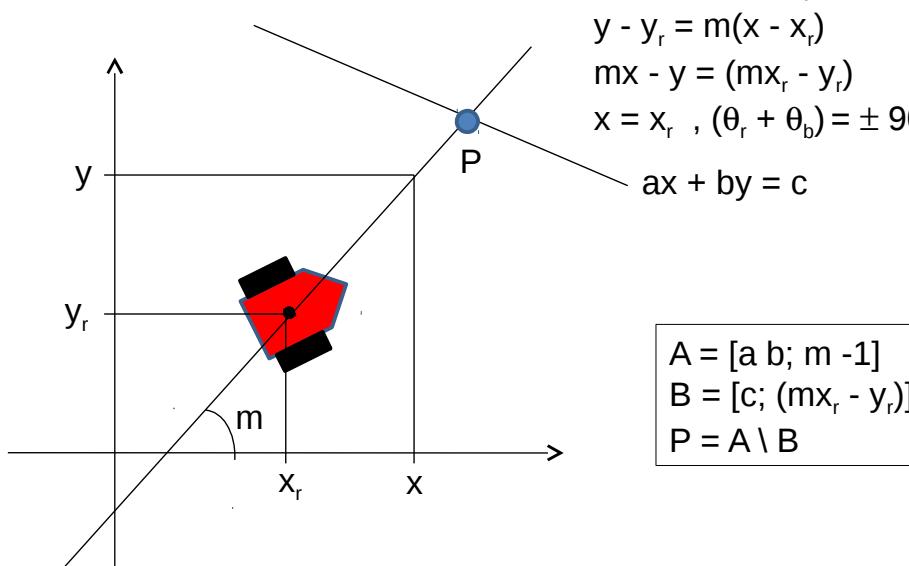


IA368W - Prof. Eleri Cardozo

45

# Filtro de Partículas

Intersecção de duas retas no Matlab

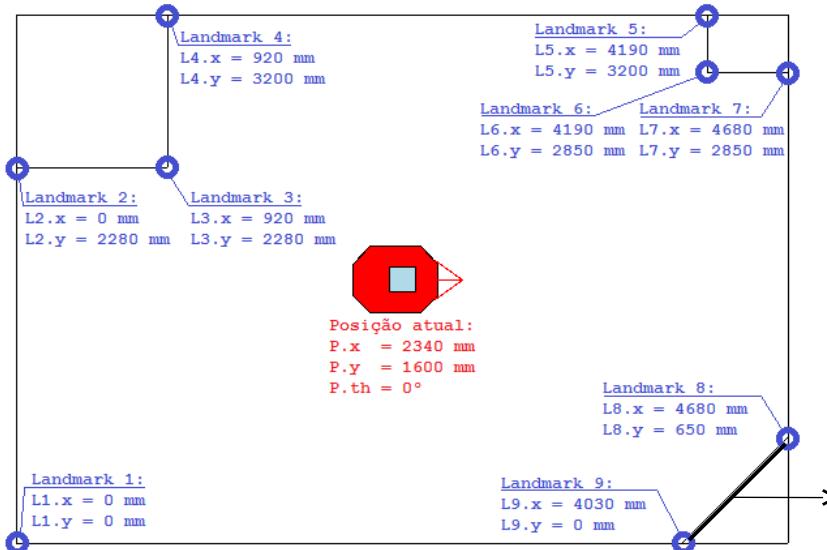


IA368W - Prof. Eleri Cardozo

46

# Filtro de Partículas

Segmentos de retas que compõem o mapa



$$ax + by = 1$$

$$4680a + 650y = 1$$

$$4030a + 0y = 1$$

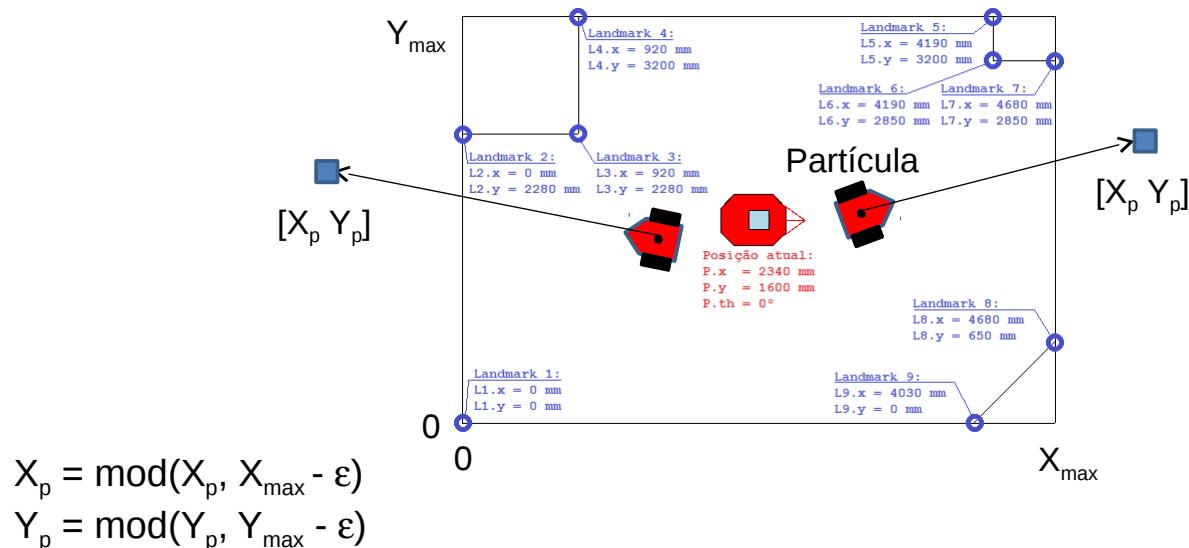
$$A = [4680 \ 650; 4030 \ 0]$$

$$B = [1; 1]$$

$$P = A \setminus B$$

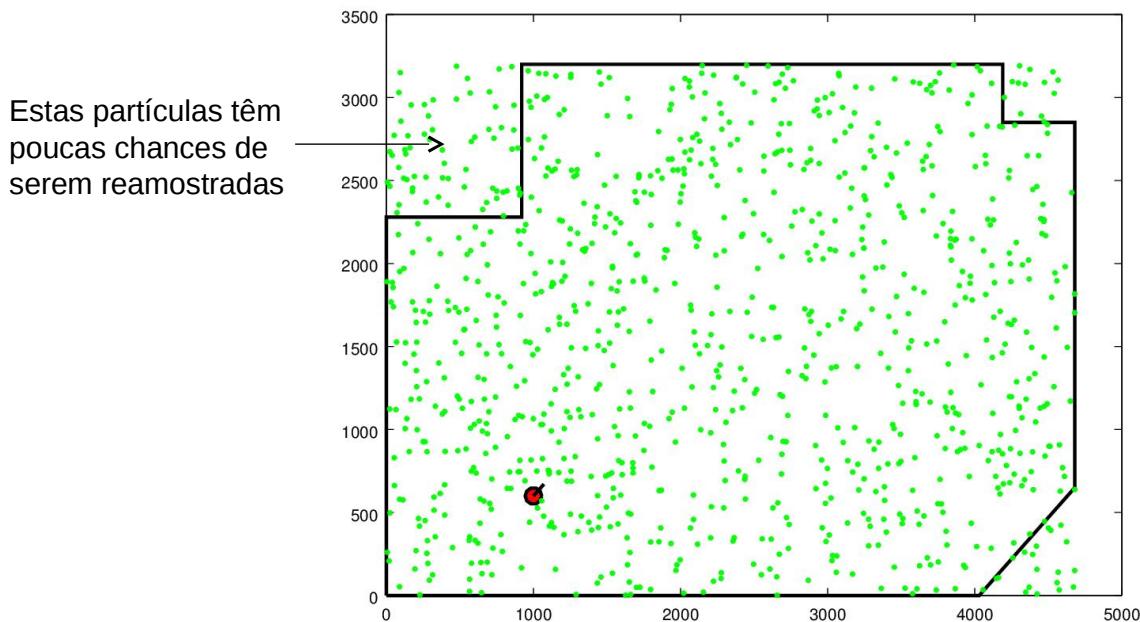
# Filtro de Partículas

Ambiente circular



# Implementação do Filtro de Partículas

2000 partículas iniciais posicionadas aleatoriamente

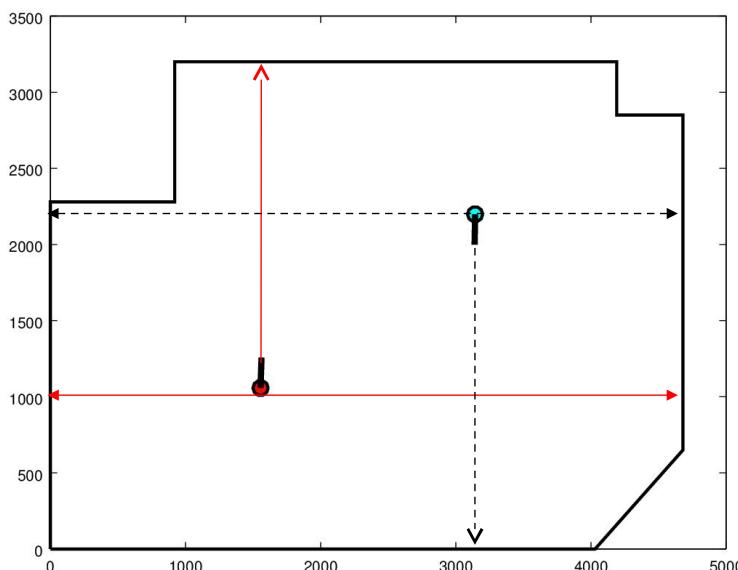


IA368W - Prof. Eleri Cardozo

49

## Filtro de Partículas Implementação

O Filtro de Partículas pode divergir devido a simetrias do ambiente:



Estratégias:

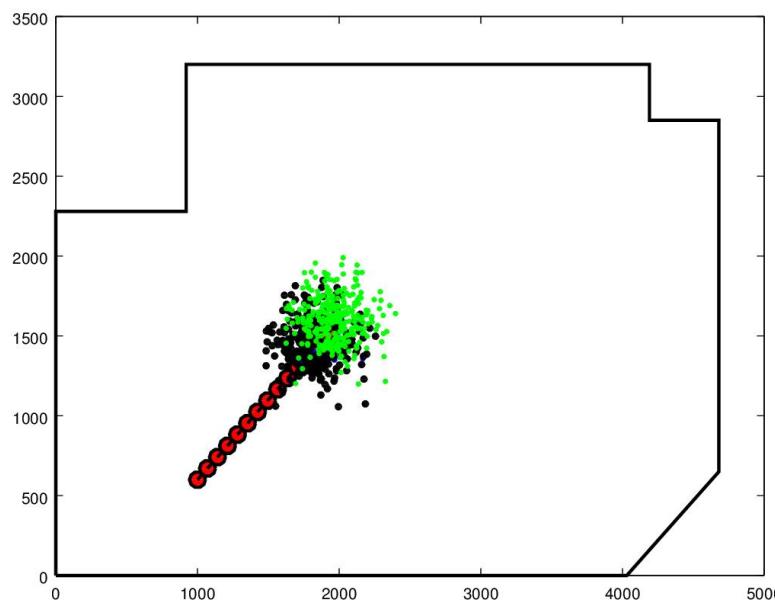
- gerar K partículas aleatórias em cada iteração.
- utilizar um número maior de partículas iniciais.

IA368W - Prof. Eleri Cardozo

50

# Filtro de Partículas

## Implementação



Após 13 iterações:

Pose da odometria:  
[1848 1448 45]

Pose do FP:  
[1864 1440 44.8]

# Filtro de Partículas

## Implementação

### Critérios de parada:

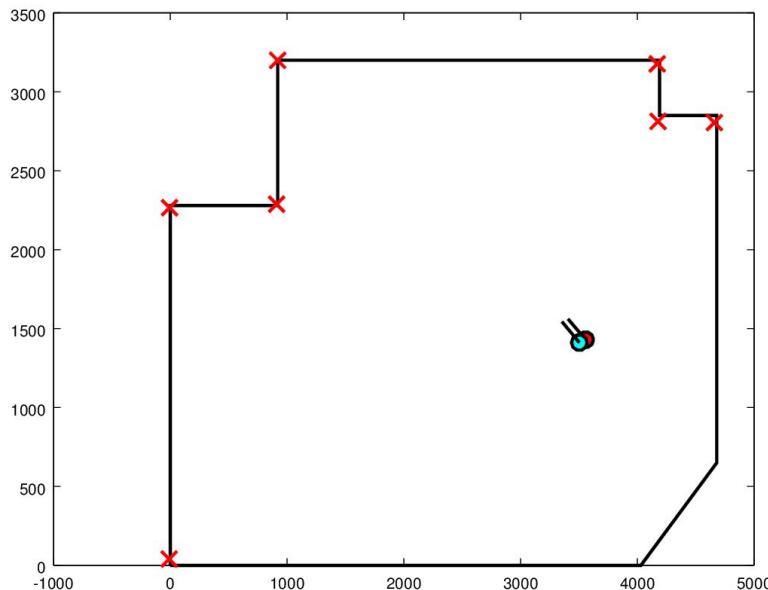
- Número fixo de iterações.
- Monitorar o número de partículas efetivas:  $N_{ef} = \frac{1}{\sum_i w_i^2}$
- Monitorar a covariância das partículas.

Ao final pode ser feita uma verificação adicional já que possuímos o mapa:

- ✓ Detecção de retas.
- ✓ Detecção de features.
- ✓ Localização local.

# Filtro de Partículas

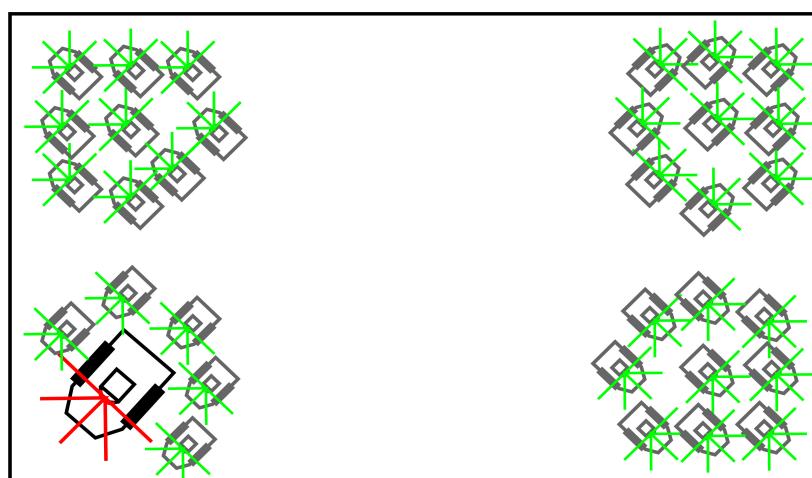
## Implementação



Detecção de landmarks utilizando o laser do robô e a pose dada pelo Filtro de Partículas.

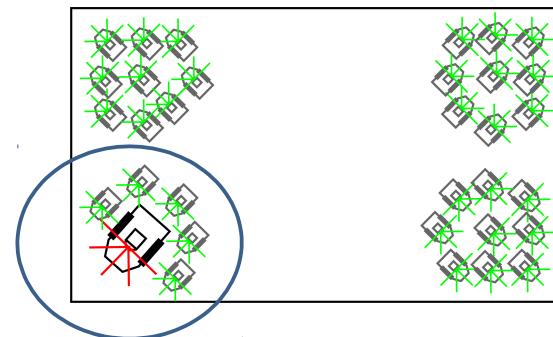
## Problemas com o Filtro de Partículas

**Ambiguidades:** mapas com elevados graus de simetria geram problemas de ambiguidade de localização.



# Aprimoramento do Filtro de Partículas

## Eliminação de Ambiguidades



- Utilizar um sensor adicional de localização:
  - bússola;
  - odometria (a partir de uma posição conhecida);
  - GPS (outdoor).
- Instrumentar o ambiente:
  - RFID (Radio Frequency Identification) em diferentes posições;
  - beacons (emissores de sinais);
  - refletores ópticos;
  - marcas visuais (cores, padrões).

## Resumo

Filtro	Hipóteses	Universo
Bayes	Única	Contínuo
Markov	Múltiplas	Discreto
Kalman	Única	Contínuo
Informação	Única	Contínuo
Partículas	Múltiplas	Contínuo