Auto-localização

Robótica

Prof. Carlos Carreto

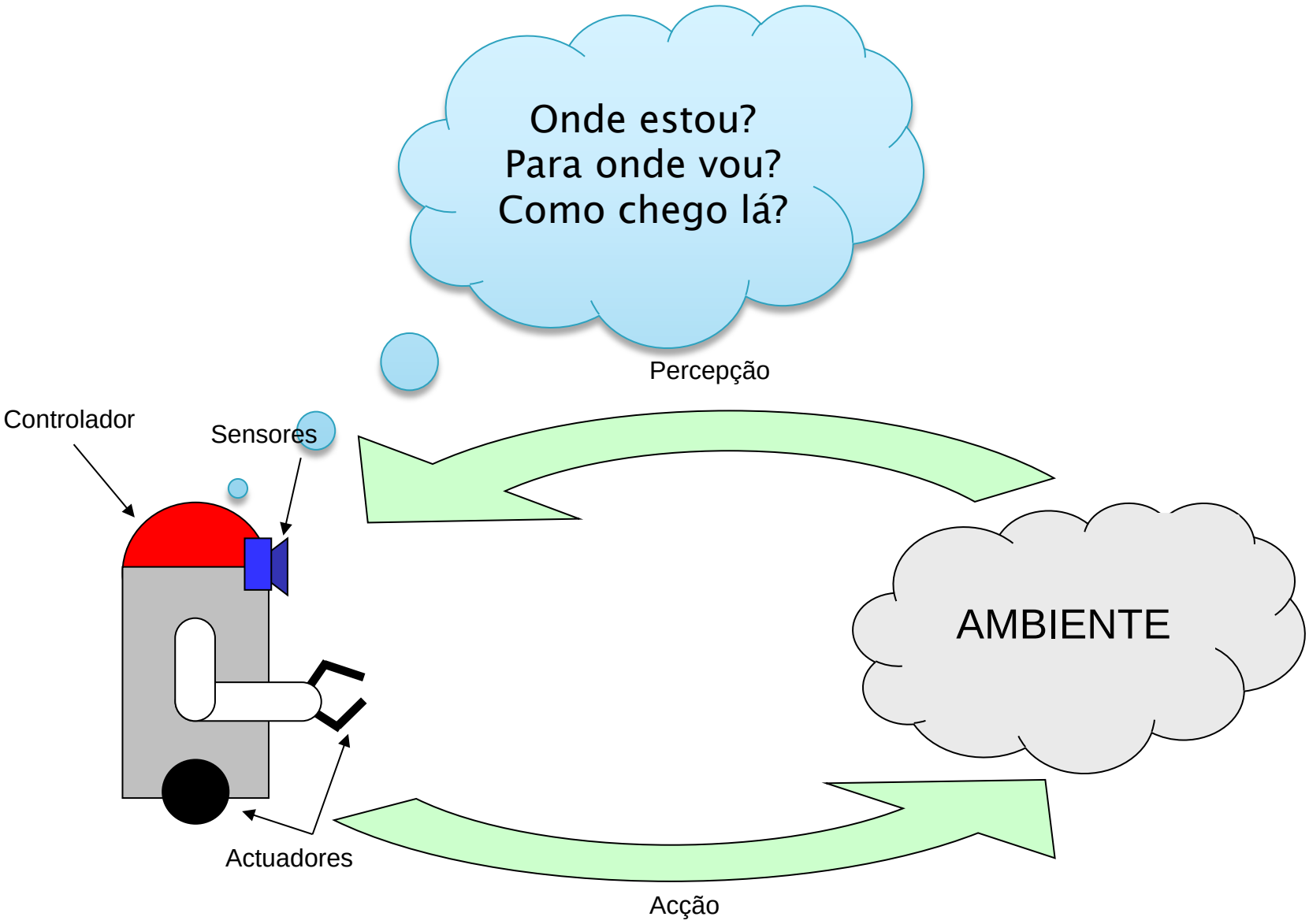
2012/2013

Engenharia Informática
Instituto Politécnico da Guarda

Sumário

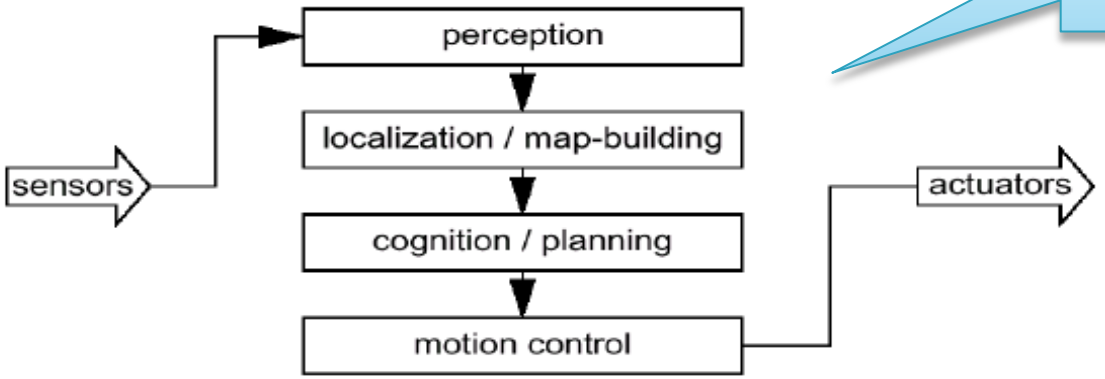
- Auto-localização
  - O Problema da Navegação
  - Estimativa da Pose
- Tipos de Medidas de Estimativa
  - Medidas Relativas
  - Medidas Absolutas
  - Medidas Mistas
- Dead Reckoning e Odometria
- Marcadores de Posição
- Localização Probabilística

O Problema da Navegação

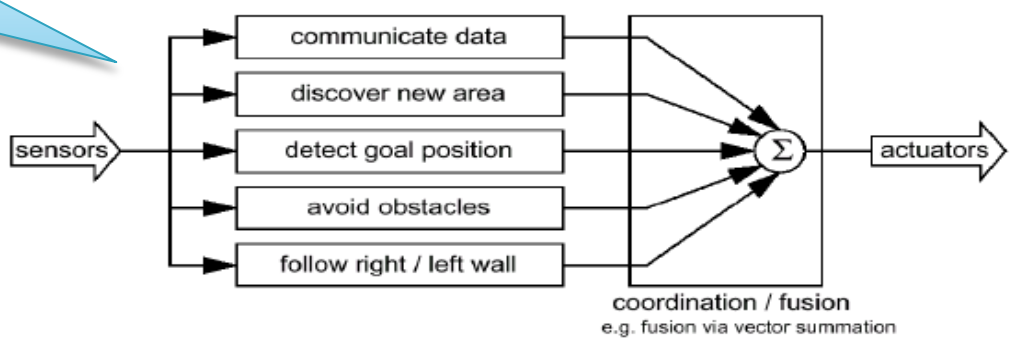


O Problema da Navegação

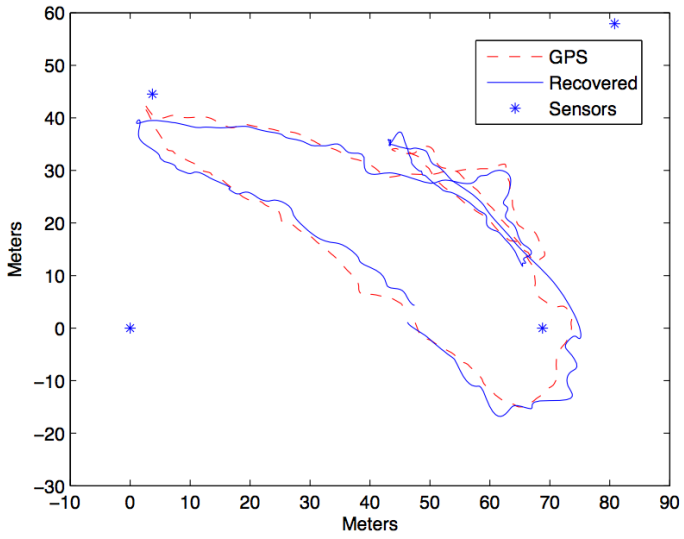
Navegação Baseada em Auto-localização



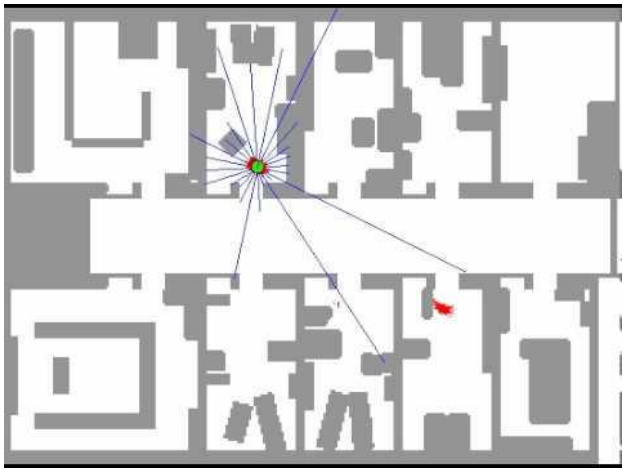
Navegação Reactiva (sem Auto-localização)



Auto-localização



Em relação a um referencial cartesiano fixo.



Em relação a um mapa do ambiente.

Pose 2D
Posição: x, y
Orientação: θ

Pose 3D
Posição: x, y, z
Orientação: θx, θy, θz

Auto-localização

- Estimativa da Pose (Pose Maintenance):
  - **Medidas relativas** – a pose é estimada com base na última pose conhecida e na medição do deslocamento do objecto (ex. Dead Reckoning – Estimativa do Deslocamento Relativo).
  - **Medidas absolutas** – a pose é determinada com base na detecção de pontos de referência do ambiente cujas posições são conhecidas (marcadores de posição).
  - **Medidas mistas** – a pose é estimada recorrendo a medidas relativas e medidas absolutas.

Auto-localização

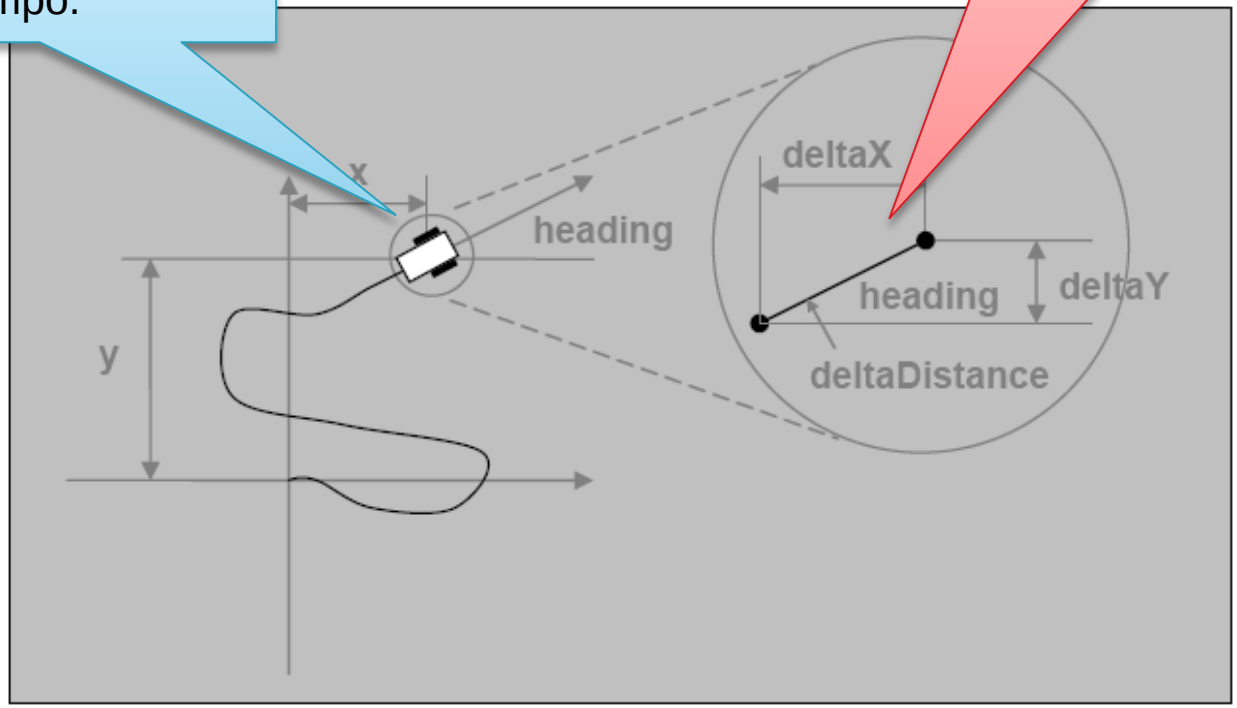
Medidas Relativas
(Dead Reckoning)

Auto-localização

Medidas Relativas (Dead Reckoning)

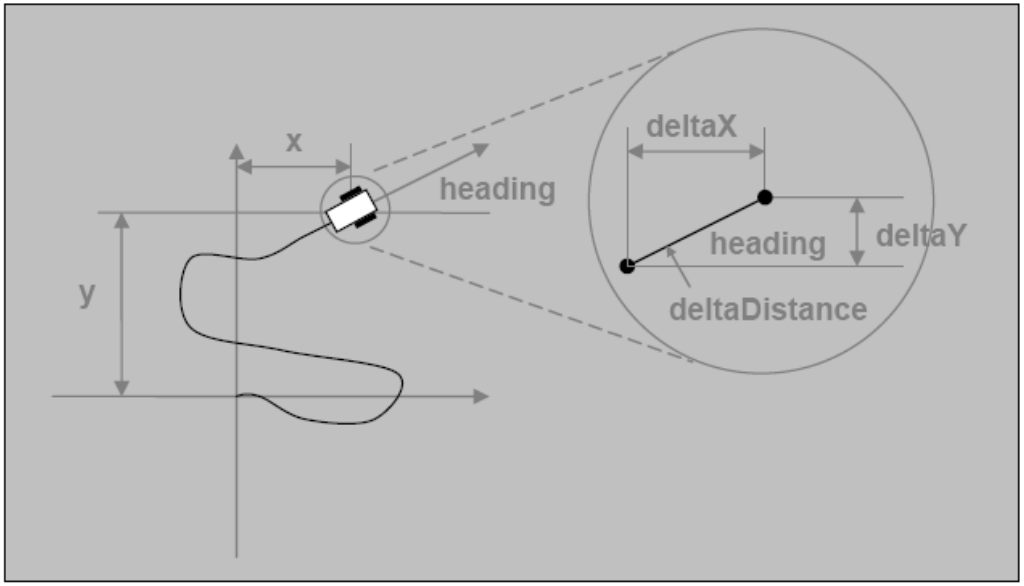
A pose do robô é estimada somando à última pose conhecida, o deslocamento sofrido pelo robô em cada unidade de tempo.

O deslocamento do robô pode ser estimado usando vários métodos. Um dos mais usados é a **Odometria**.



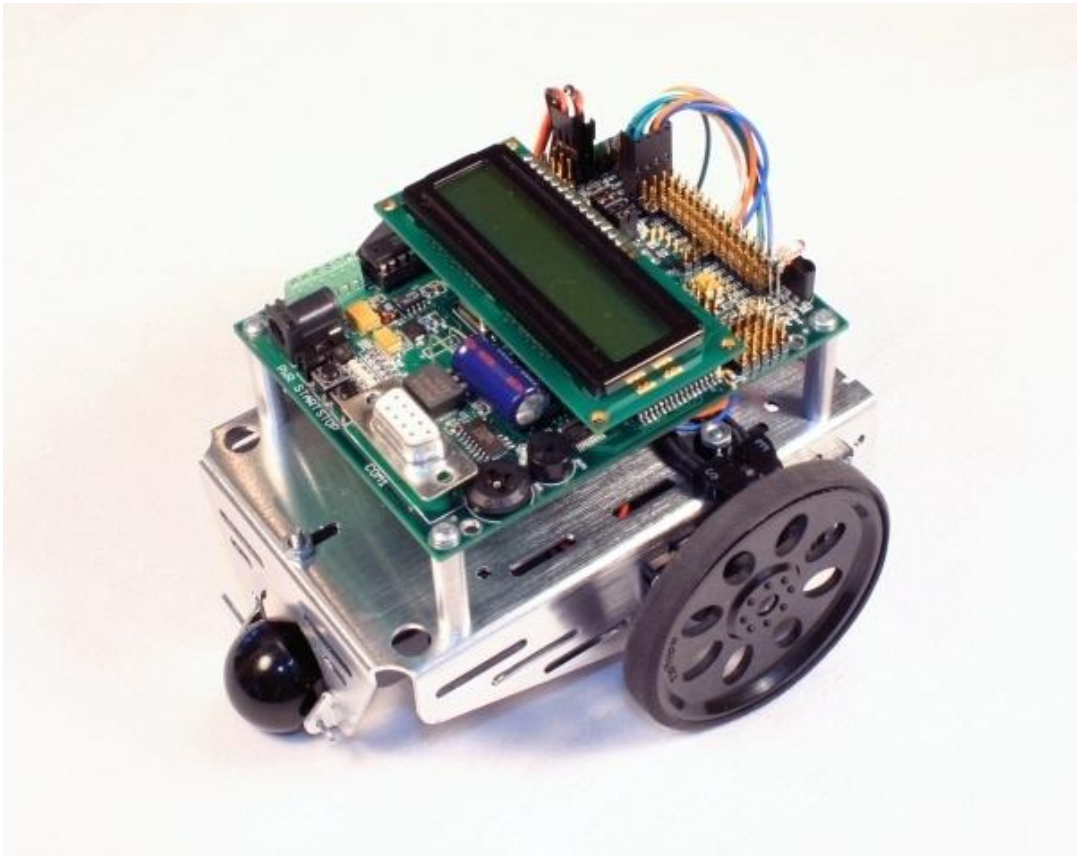
Auto-localização

Medidas Relativas (Dead Reckoning)



Em cada instante de tempo (muito pequeno), consideramos que o robô roda sobre si próprio um ângulo **heading** e percorre uma distância **deltaDistance**.

Aplicando o teorema de Pitágoras, podemos determinar o **deltaX** e o **deltaY** que somados aos valores anteriores de **x** e **y** dão uma estimativa da nova pose do robô.



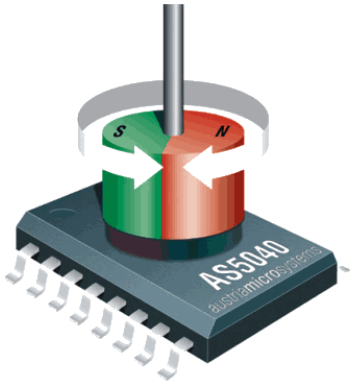
Implementação de Dead Reckoning e Odometria no robô de condução diferencial IntelliBrain-Bot

Odometria

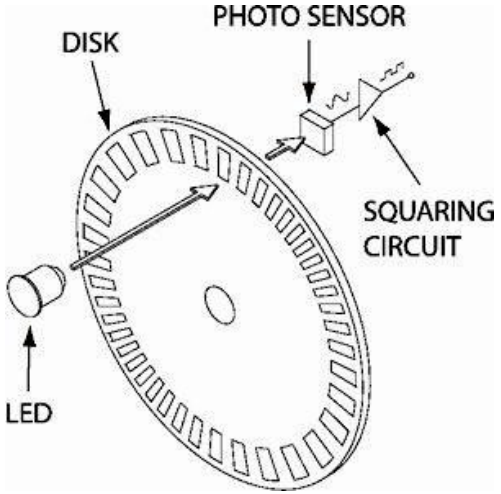
- Método para estimar o deslocamento do robô através da medição do movimento angular das rodas.
  - É um método totalmente auto-contido (não depende de partes externas ao robô).
  - Está sujeito a erros que pela natureza incremental do método, podem crescer muito depressa e de forma descontrolada.
  - A medição do movimento angular é normalmente realizada através de **shaft encoders** (codificadores de eixo).

Odometria

Encoder Industrial

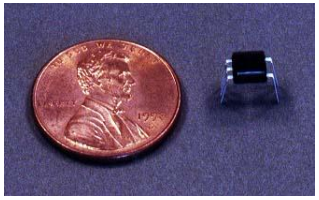


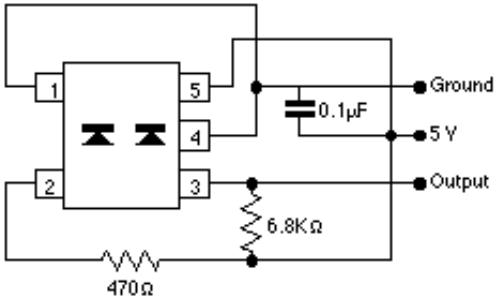
Encoder Magnético

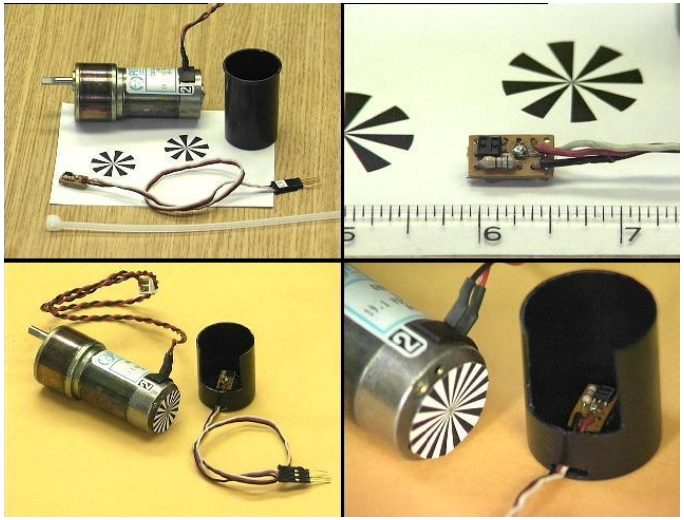


Encoder Óptico

Encoder Acoplado a um Motor

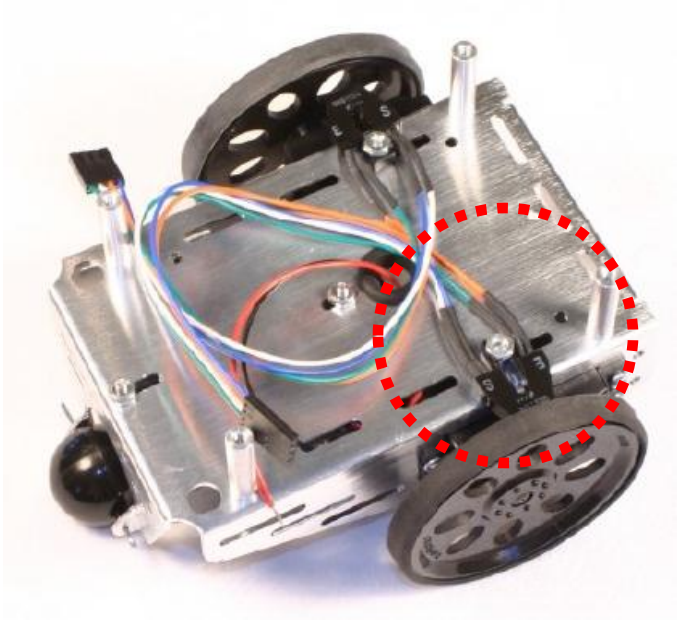


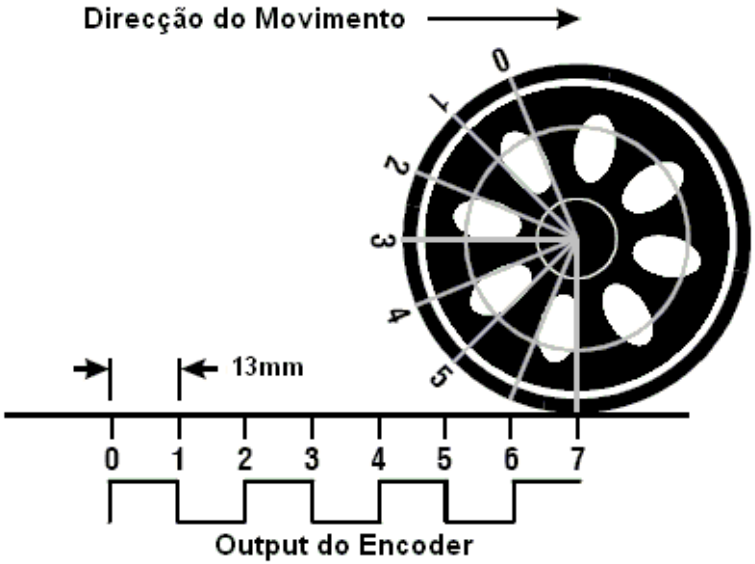




Encoder Caseiro

Odometria



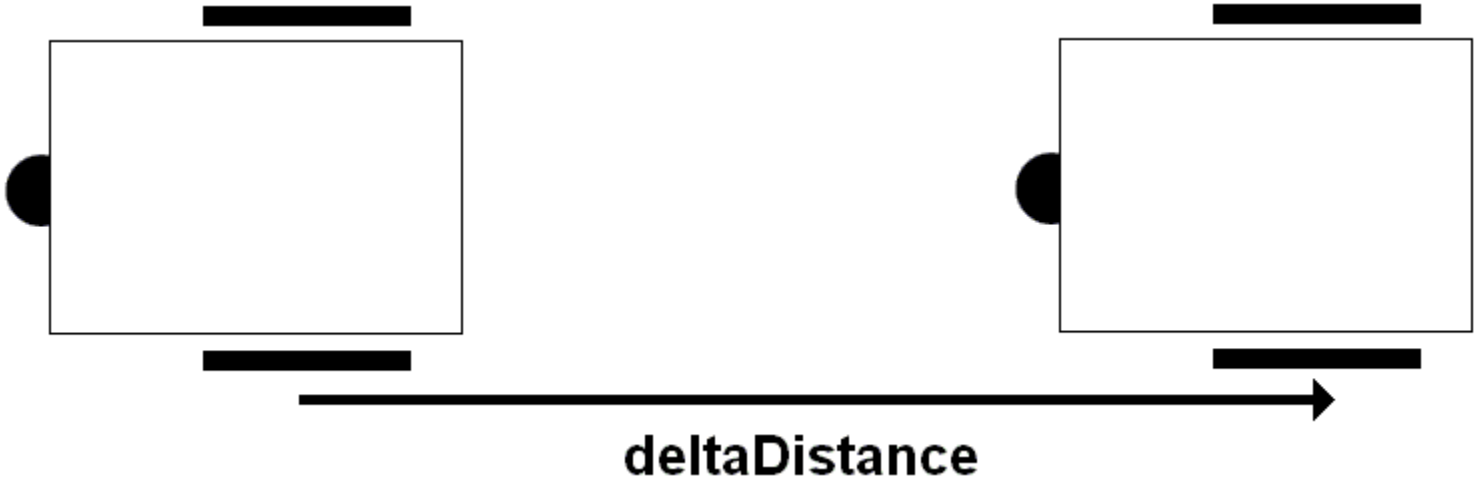


Ao fazer com que um sensor detecte o limiar dos buracos, é possível gerar um trem de pulsos à medida que a roda gira. Na roda da figura, o número de pulsos (**counts**) por volta, **countsPerRevolution**, é 16.

A distância percorrida pela roda em cada pulso é dada por:

distancePerCount = Pi * diameterWheel / countsPerRevolution

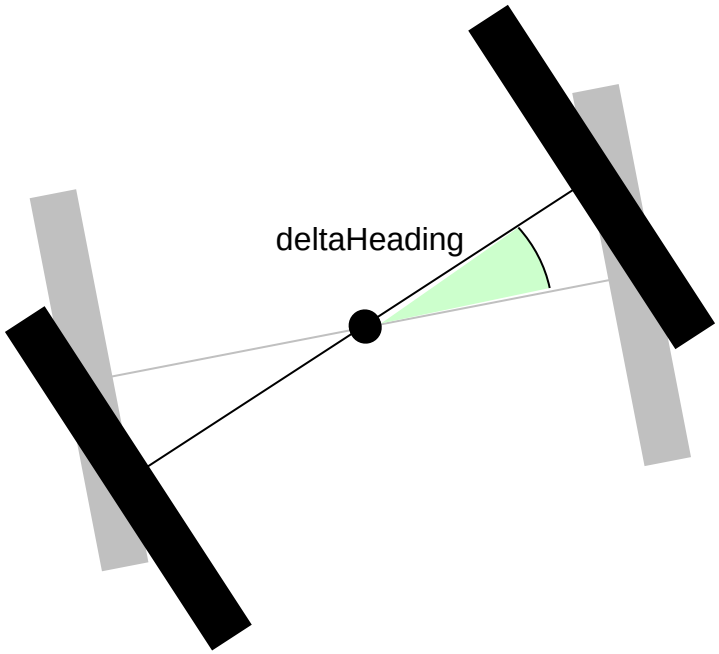
Odometria



Quando o robô se move em linha recta, a distância que percorre é simplesmente a média do número de pulsos dos encoders de cada roda, vezes a distância por pulso:

deltaDistance = (leftCounts + rightCounts) / 2 * distancePerCount

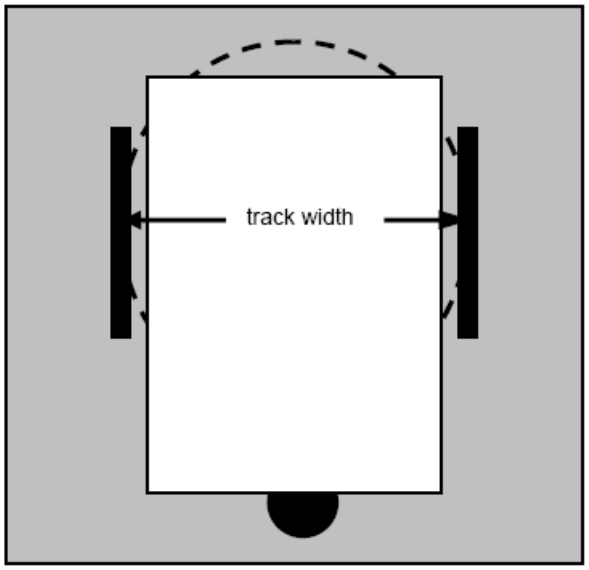
Odometria



O ângulo de rotação do robô é dado pela soma dos pulsos contados por cada roda (tendo em conta o sentido da contagem), vezes o ângulo que o robô roda por pulso:

deltaHeading = (rightCount – leftCount) * radiansPerCount

Odometria



Quando o robô roda sobre si próprio 360º, as rodas percorrem a circunferência traçado no desenho.

O perímetro dessa circunferência é:

circunferenceTw = Pi * trackWidth

O perímetro de uma roda é:

circunferenceWheel = Pi * wheelDiameter

Como o número de pulsos correspondente a **circunferenceWheel** é **countsPerRevolution**, o número de pulsos de uma única roda, correspondente à rotação de 360º da figura é:

countsPerRotation = (circunferenceTw / circunferenceWheel) * countsPerRevolution

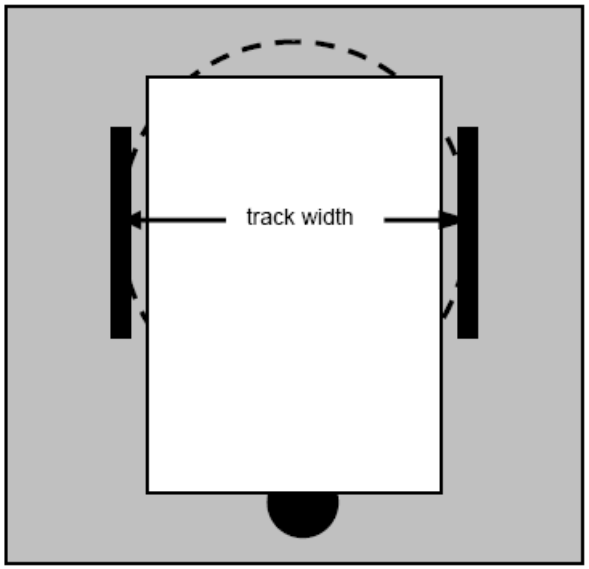
ou

countsPerRotation = (trackWidth/wheelDiameter) * countsPerRevolution

Para as duas rodas (ignorando para já o sentido de rotação), é

countsPerRotation = 2 * (trackWidth/wheelDiameter) * countsPerRevolution

Odometria



O número de radianos que o robô roda por cada pulso (considerando as 2 rodas), é:

radiansPerCount = 2 * Pi / countsPerRotation

Substituindo:

radiansPerCount = 2 * Pi / (2 * (trackWidth / wheelDiameter) * countsPerRevolution)

Que podemos simplificar para:

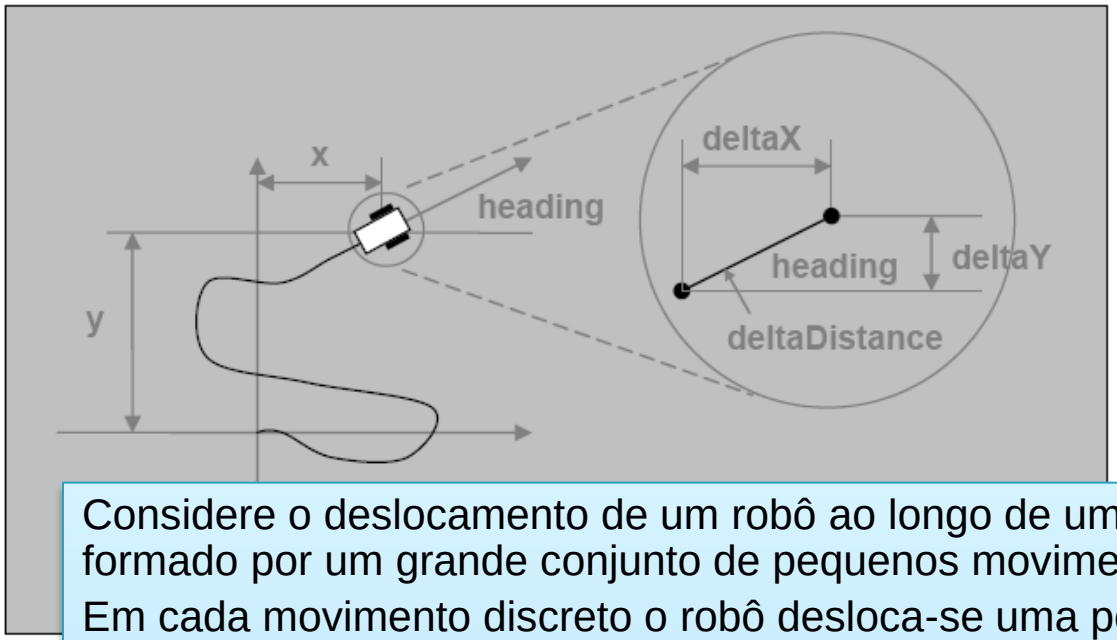
radiansPerCount = Pi / ((trackWidth / wheelDiameter) * countsPerRevolution)

ou

radiansPerCount = Pi * (wheelDiameter / trackWidth) / countsPerRevolution

Notar que **radiansPerCount** depende da geometria do robô e só é calculado uma vez.

Odometria



Considere o deslocamento de um robô ao longo de um trajecto arbitrário como sendo formado por um grande conjunto de pequenos movimentos discretos.

Em cada movimento discreto o robô desloca-se uma pequena distância **deltaDistance** na direcção **heading**.

Se assumirmos que a direcção **heading** não se altera significativamente durante essa pequena distância, a alteração da posição ao longo dos eixos **X** e **Y** é dada respectivamente por **deltaX** e **deltaY**.

deltaX = deltaDistance * cos(heading)

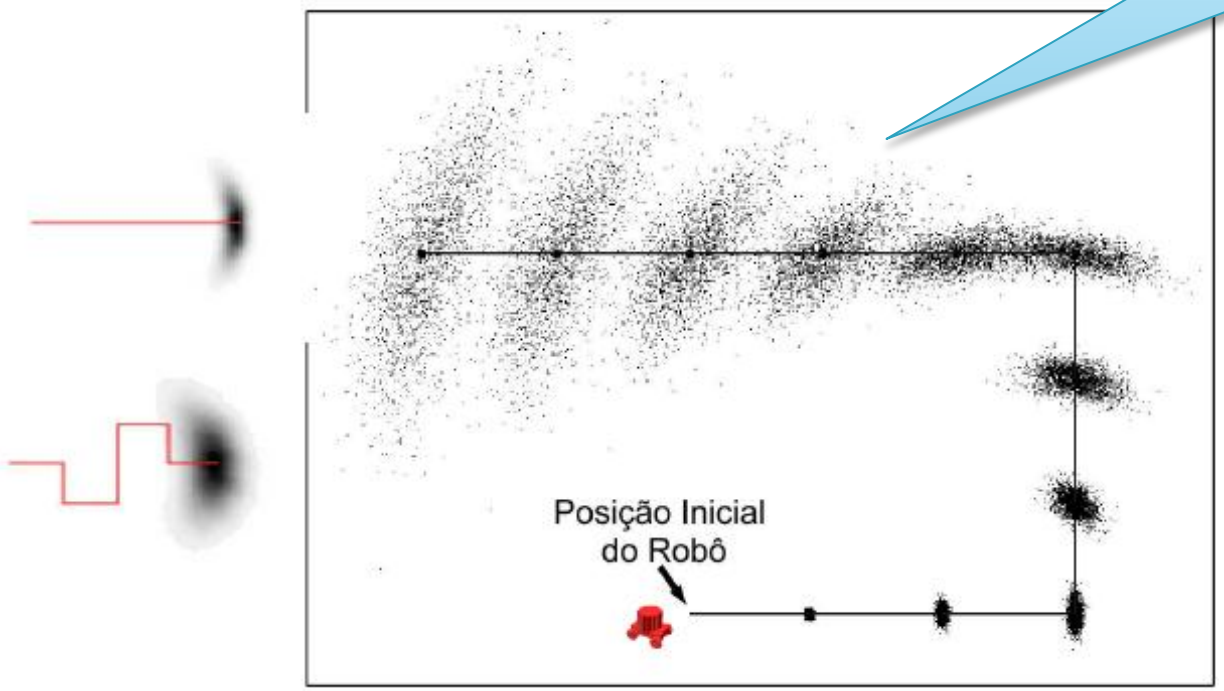
deltaY = deltaDistance * sin(heading)

Odometria

- Está sujeita a dois tipos de erros:
  - **Erros sistemáticos** – relacionados com as características e a calibração do sistema.
  - **Erros aleatórios** – podem ser descritos por modelos de erros na tentativa de os diminuir, mas provocarão sempre uma incerteza na determinação da posição do robô.
- Algumas fontes de erros da Odometria:
  - Resolução limitada (tempo discreto, distância por pulso)
  - Falta de precisão na caracterização da geometria do robô
  - Eixos desalinhados
  - Rodas com diâmetros diferentes
  - Derrapagens e bloqueios das rodas
  - etc

Odometria

Consequência dos erros - Os erros (sistemáticos e aleatórios), associados à estimativa do deslocamento, provocam erros na determinação da pose do robô.



Auto-localização
Medidas Absolutas
(Marcadores de Posição)

Auto-localização

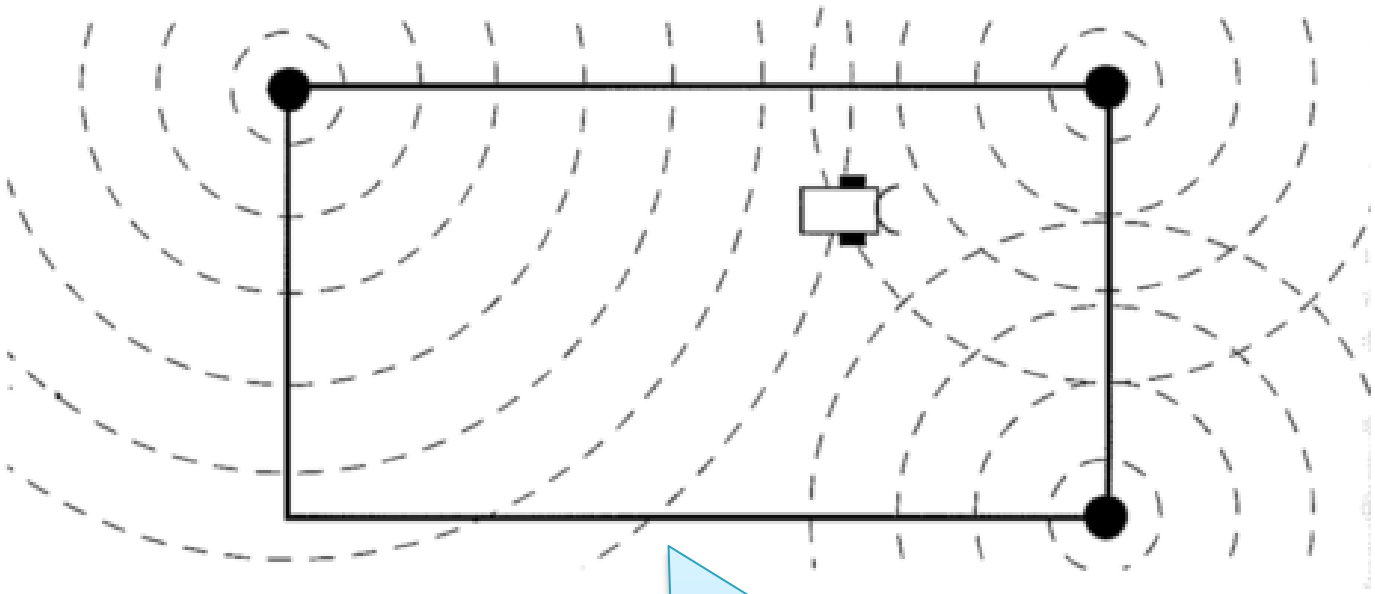
Medidas Absolutas (Marcadores de Posição)

- **Beacons** – São sinais de referência ou faróis (marcas activas que emitem algum tipo de feixe de energia) que permite ao robô estimar a sua posição aproximada.

- **Landmarks** – São marcas ou pontos de referência para o robô detectar e estimar a sua posição aproximada.
  - **Marcas naturais** – fazem parte do ambiente do robô.
  - **Marcas artificiais** – são inseridas no ambiente com o propósito de serem detectadas pelo robô.

Auto-localização

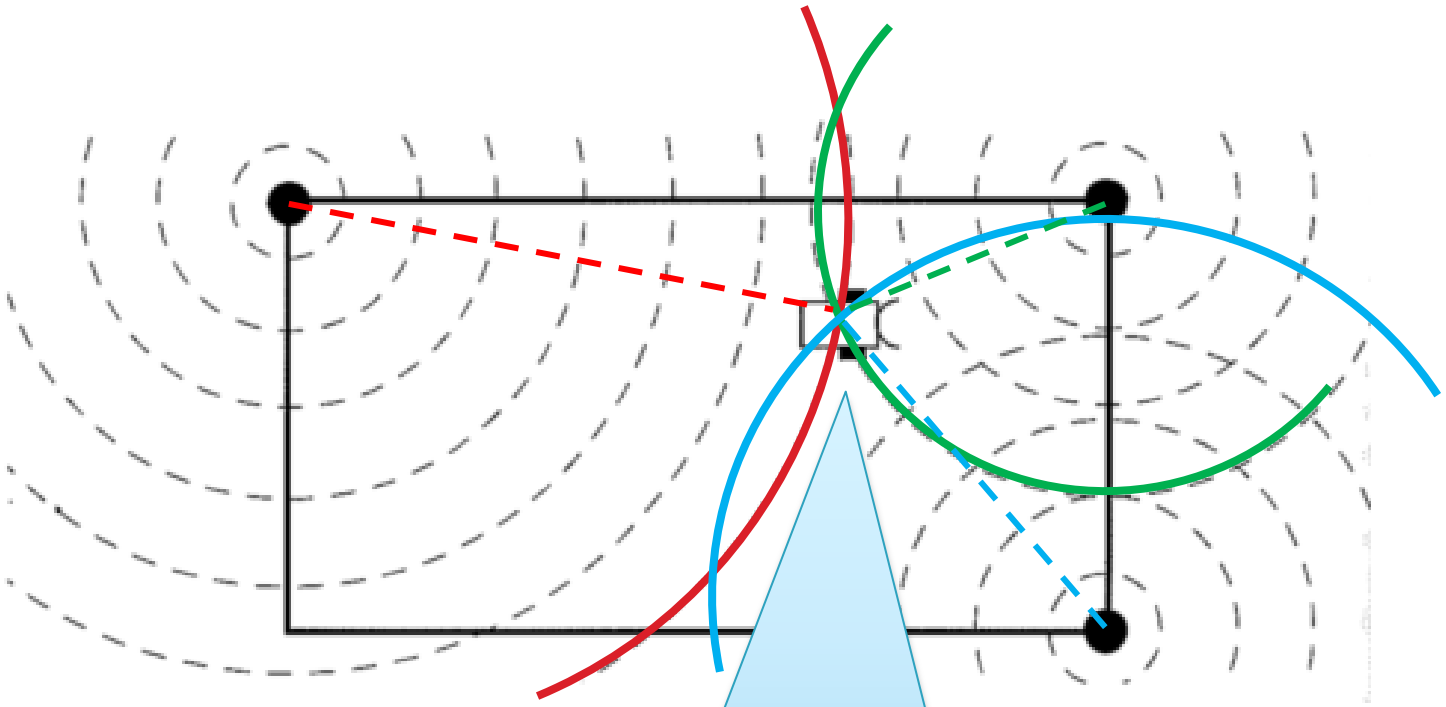
Medidas Absolutas (Marcadores de Posição)



Marcadores de posição como o sistema GPS (outdoor) ou faróis de infravermelhos, sonar, laser ou rádio (indoor/outdoor), podem ser usados para o robô estimar a sua pose através de métodos como **triangulação** e **trilateração**.

Auto-localização

Medidas Absolutas (Marcadores de Posição)

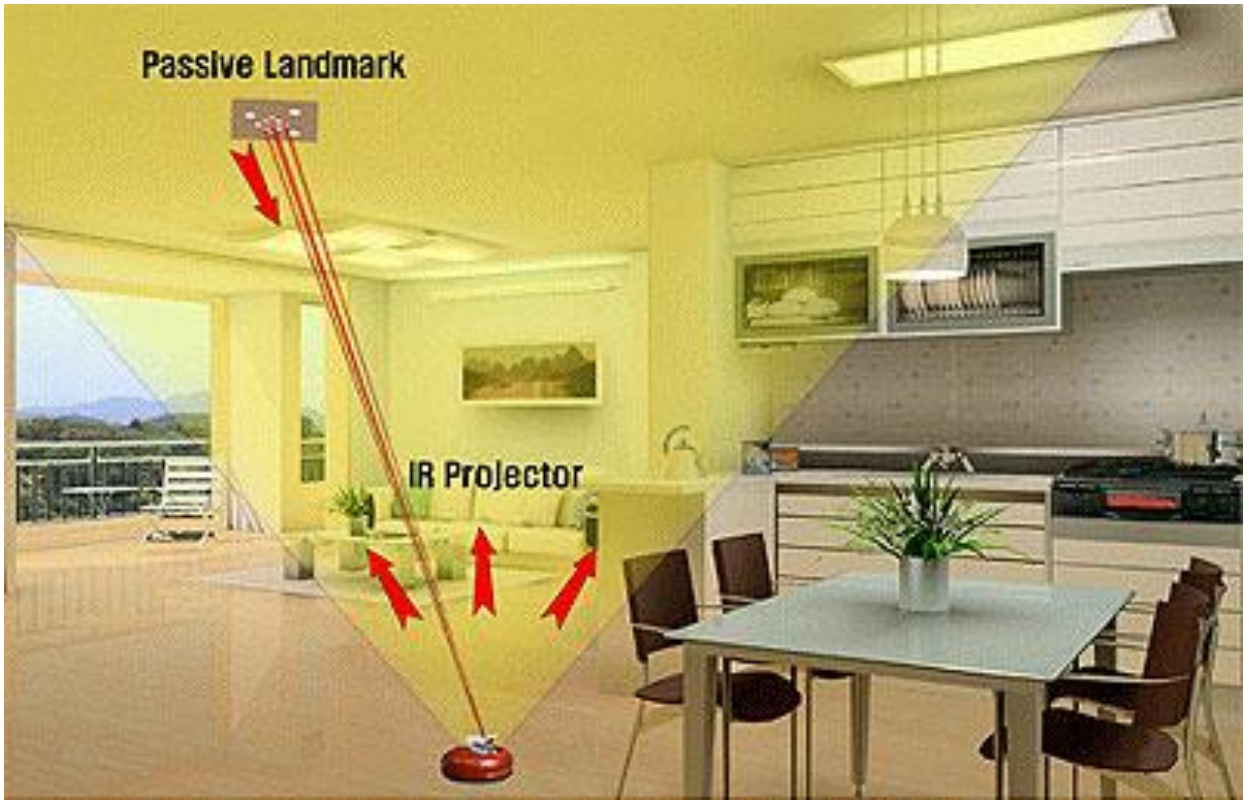


Exemplo de Trilateração – A posição do robô é estimada com base na distância entre este e um conjunto de marcadores de posição existentes no ambiente.

Auto-localização

Medidas Absolutas (Marcadores de Posição)

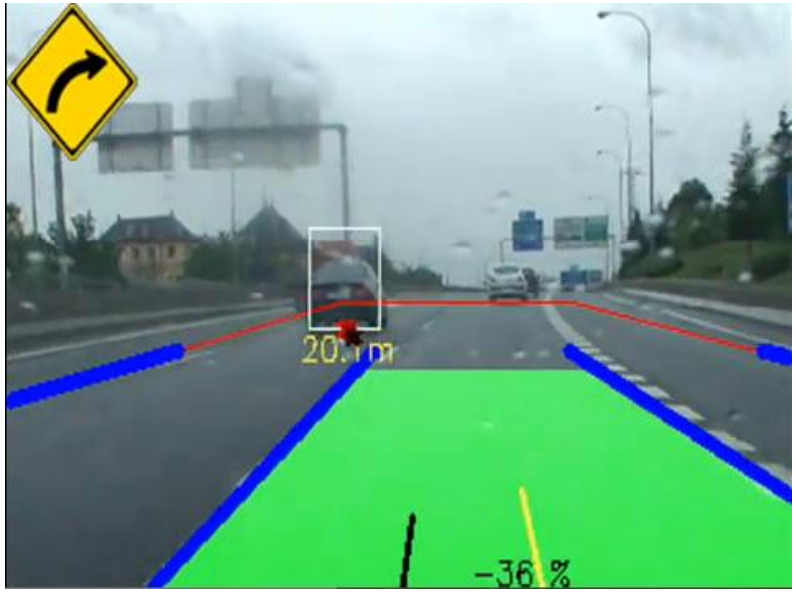
Hagisonic StarGazer Robot Localization System



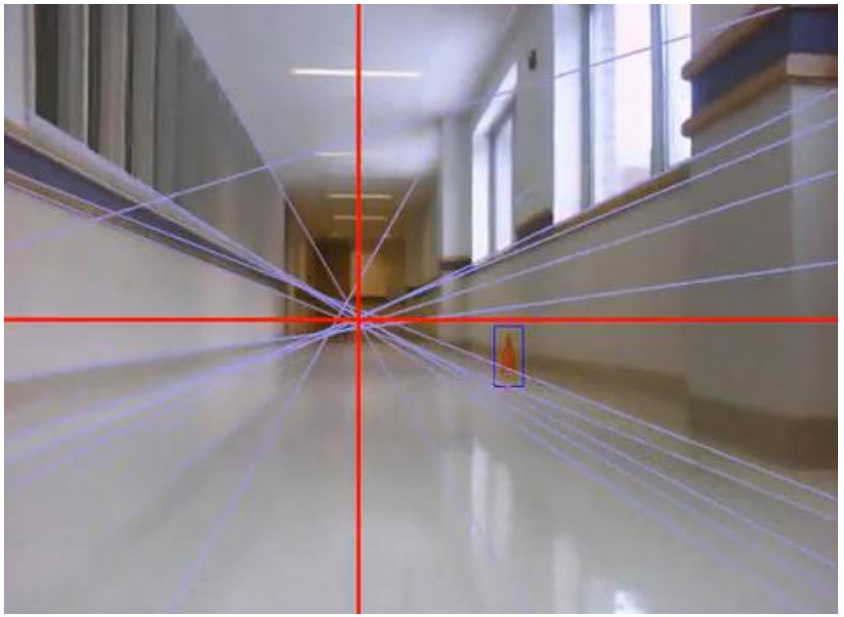
Auto-localização

Medidas Absolutas (Marcadores de Posição)

- **Sensor-based servoing (Servo-controlo)** – A posição do robô é estimada em relação a marcadores de posição a partir de medições resultantes da percepção sensorial.



http://www.youtube.com/watch?v=JmxDIuCIIcg

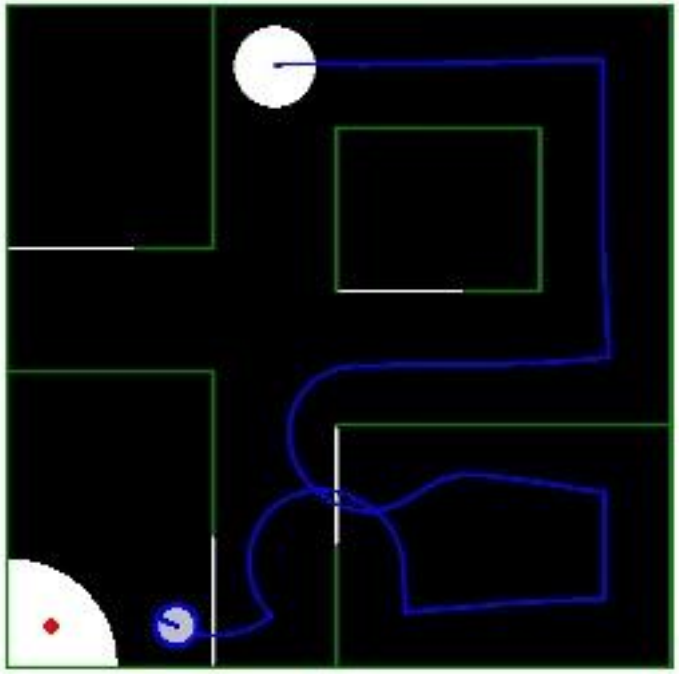


http://www.youtube.com/watch?v=nb0VpSYtJ_Y

Auto-localização

Medidas Absolutas (Marcadores de Posição)

- **Sensor-based servoing (Servo-controlo)** – A posição do robô é estimada em relação a marcadores de posição a partir de medições resultantes da percepção sensorial.

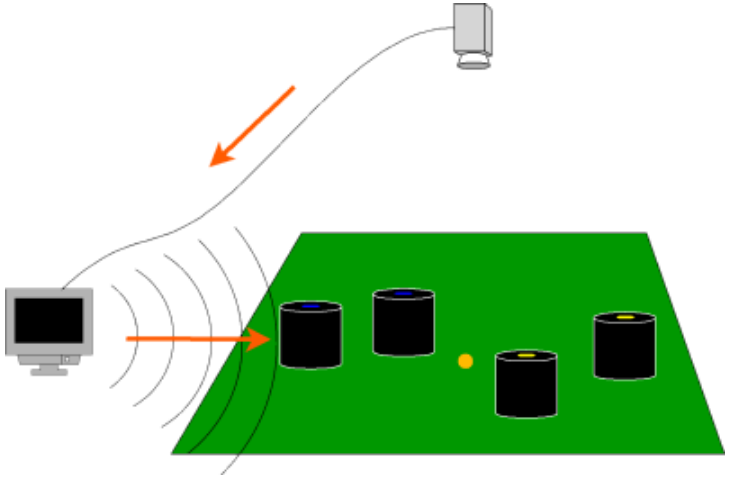


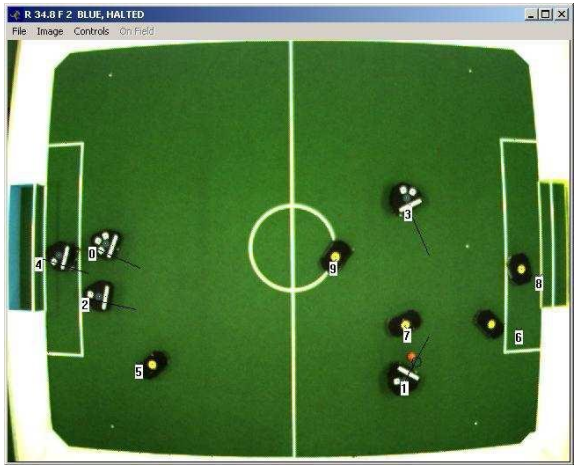
http://www.youtube.com/watch?v=E1x7Gv5Uzqc

Auto-localização

Medidas Absolutas (Marcadores de Posição)

- **Tracking Externo**

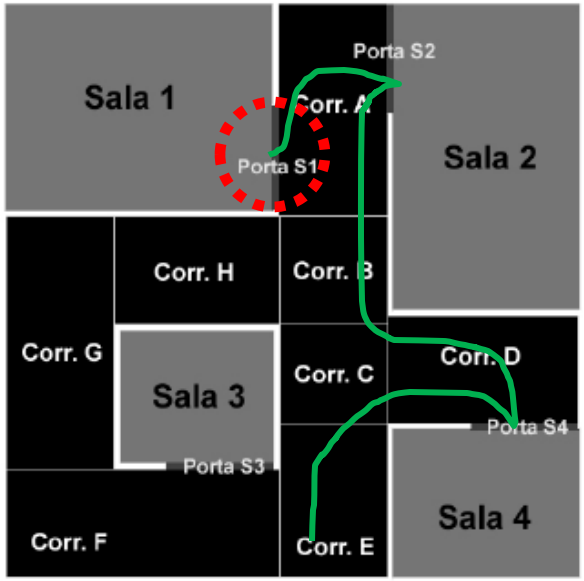


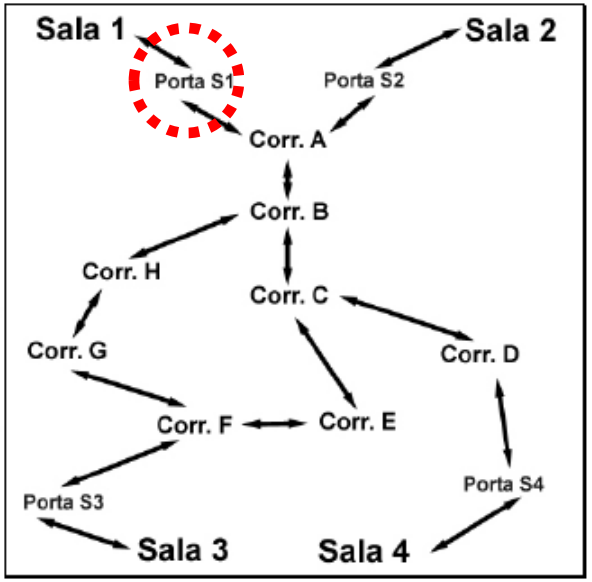


Auto-localização

Medidas Absolutas (Marcadores de Posição)

- **Localização Topológica** – O robô identifica marcas no ambiente para estimar a sua localização num mapa topológico que representa as regiões do ambiente e a conectividade entre elas.





Auto-localização

Medidas Absolutas (Marcadores de Posição)

- **Que marcadores podem ser usados no RB?**

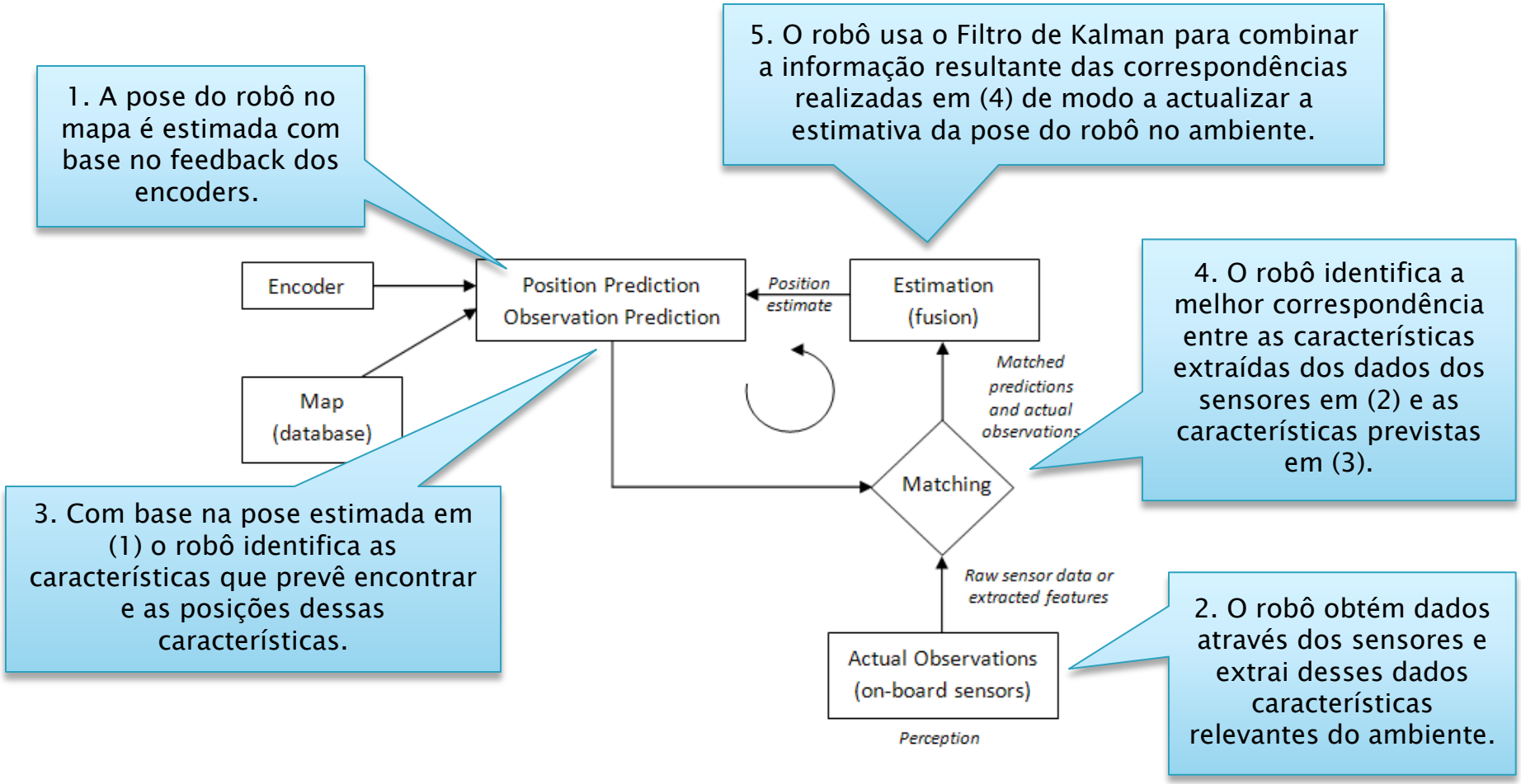
- Paredes
- Esquinas
- Linhas das entradas
- Linha da vela
- Alcatifas
- Mobília
- Base da vela
- Rampas
- Ponto de partida
- . . .

Localização Probabilística

- Usa técnicas que representam a localização como uma distribuição estatística na tentativa de ultrapassar a imprecisão dos sensores e dos métodos de localização.
- São baseadas no Filtro de Bayes.
  - $\text{Bel}(x_t) = P(x_t \mid z_1, z_2, \ldots, z_t)$.
  - Qual é a probabilidade de se estar na posição x se a história das medidas dos sensores é $z_1, z_2, \ldots, z_t$?"
- Combinam a informação sobre o deslocamento do robô com a integração de dados provenientes das medições dos sensores e de um mapa do ambiente.

Localização Probabilística

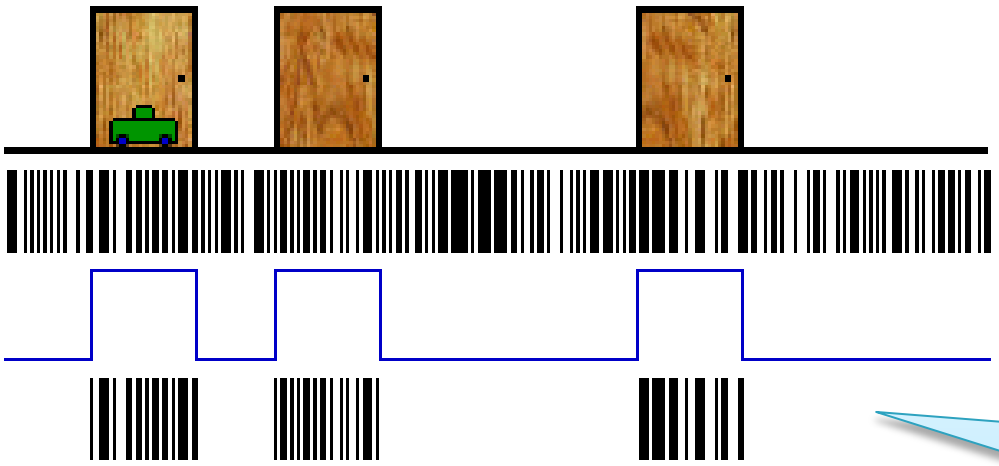
- **Filtro de Kalman**



Localização Probabilística

- **Técnicas de Monte Carlo (Filtro de Partículas)**

Exemplo 1D



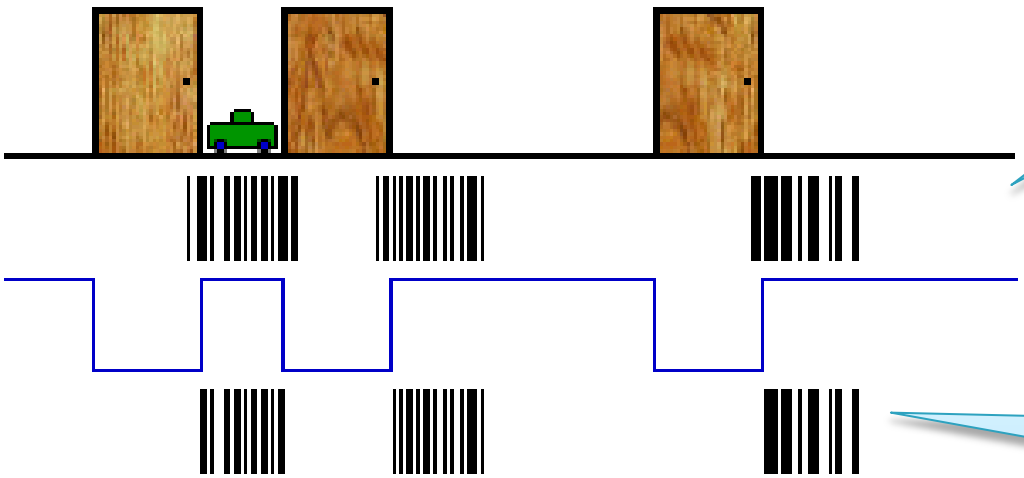
Inicialmente a posição do robô é representada por um conjunto de partículas distribuídas aleatoriamente, que representam a probabilidade do robô se encontrar em dada posição. .

O robô tem um sensor para detectar portas. Se detectar que está junto a uma porta, pode descartar as partículas que correspondem a posições entre as portas. A linha azul representa o filtro aplicado às partículas.

Localização Probabilística

- **Técnicas de Monte Carlo (Filtro de Partículas)**

Exemplo 1D



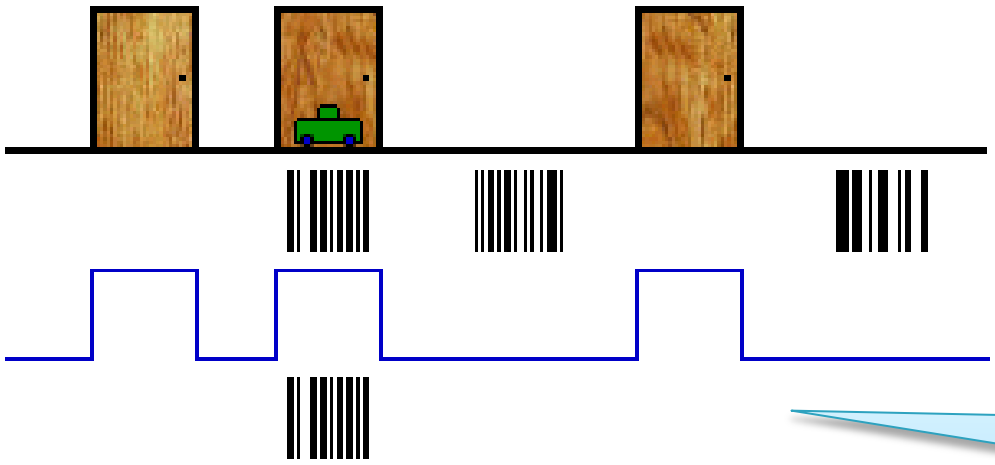
À medida que o robô se desloca, as partículas são também deslocadas da mesma quantidade.

Neste caso nenhuma porta é detectada pelo que devemos inverter a regra definida pela linha azul de modo a filtrar mais algumas partículas.

Localização Probabilística

- **Técnicas de Monte Carlo (Filtro de Partículas)**

Exemplo 1D



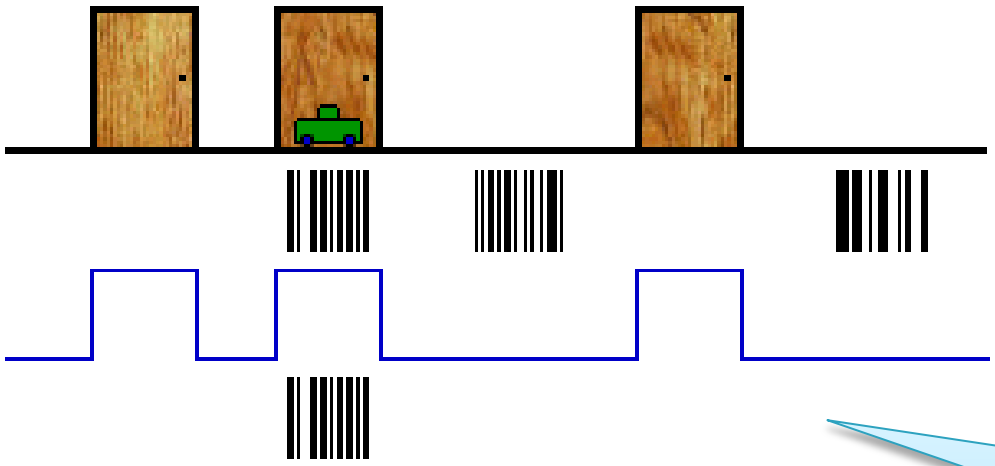
Quando o robô se move novamente, as partículas movem-se também e uma nova porta é detectada.

As coisas começam a ficar interessantes pois aplicando novamente o filtro conseguimos eliminar um grande número de partículas.

Localização Probabilística

- **Técnicas de Monte Carlo (Filtro de Partículas)**

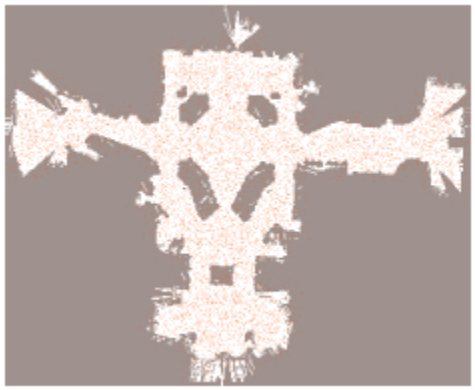
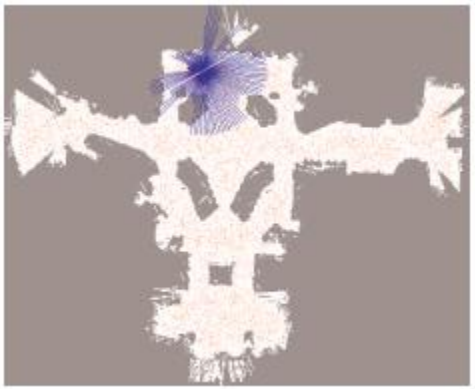
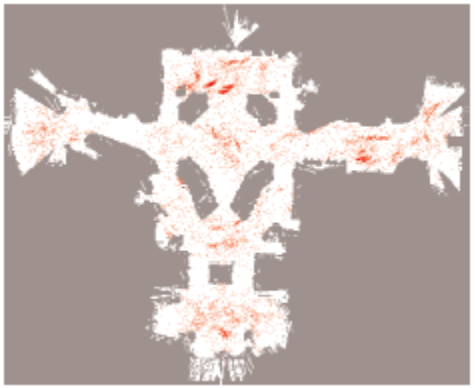
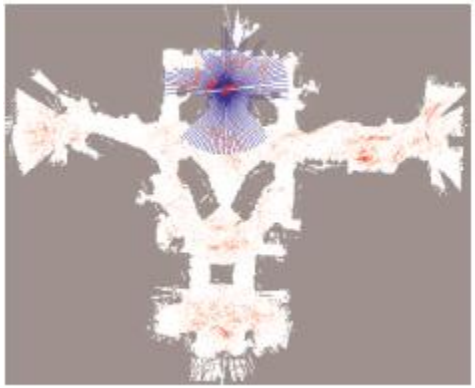
Exemplo 1D

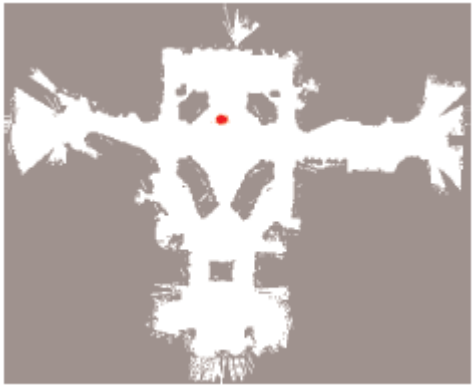


Se o processo continuar a ser repetido acabaremos com um número de partículas muito pequeno cujas posições são uma estimativa muito precisa da posição do robô.

Localização Probabilística

- **Técnicas de Monte Carlo (Filtro de Partículas)**



Bibliografia

- **Introduction to Autonomous Mobile Robots** (2004) Roland Siegwart and Illah Nourbakhsh
- **Computational Principles of Mobile Robotics** (2000) Gregory Dudek, Michael Jenkin
- **Probabilistic Robotics** (2005) Sebastian Thrun, Wolfram Burgard, Dieter Fox