IIC2685 Robótica Móvil
I – 2022

Capítulo 4.2

Modelos de Sensor

Profesor: Gabriel Sepúlveda V.
grsepulveda@ing.puc.cl

Agenda

- Entender qué es un modelo probabilístico de sensor
- Dar los fundamentos de tres tipos de modelos de sensor
  - Beam-based sensor model
  - Likelihood Fields sensor model
  - Map-matching sensor model

Recordemos el Capítulo 5.1

- Cuando queremos estimar el "estado del mundo" x_t dadas las acciones *u* y observaciones *z* ocurridas hasta el tiempo *t*:

$$
\begin{aligned}
Bel(x_t) &= P(x_t | u_1, z_1, \ldots, u_t, z_t) \\
&= \eta P(z_t | x_t) \int_{x_{t-1}} P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}
\end{aligned}
$$

Recordemos el Capítulo 5.1

- Cuando queremos estimar el "estado del mundo" x_t dadas las acciones *u* y observaciones *z* ocurridas hasta el tiempo *t*:

$$
\begin{aligned}
Bel(x_t) &= P(x_t|u_1, z_1, \ldots, u_t, z_t) \\
&= \eta P(z_t|x_t) \int_{x_{t-1}} P(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}
\end{aligned}
$$

Factor de normalización

Sensor model

Motion model

"Belief" anterior

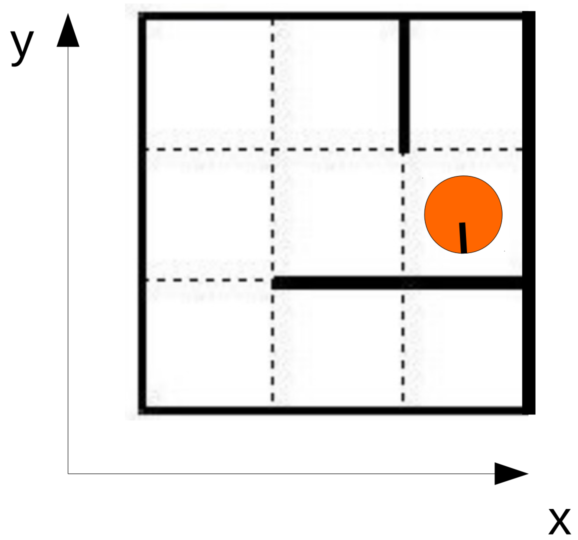
Sensor Model

- **Modelo de sensor:** describe el proceso de generación de mediciones dentro de un ambiente físico.
- El ambiente físico o entorno, es representado por un “mapa” *m*, tal que:

$$P(z|x) \rightarrow P(z|x, m)$$

- **Problema:** dado el estado del mundo (x_t) y el mapa (*m*), ¿ cuál es la distribución de la observación del robot ?
  - En otras palabras, ¿ qué tan probable es observar *z* ?
- Veamos un ejemplo simple en *MazeWorld* !

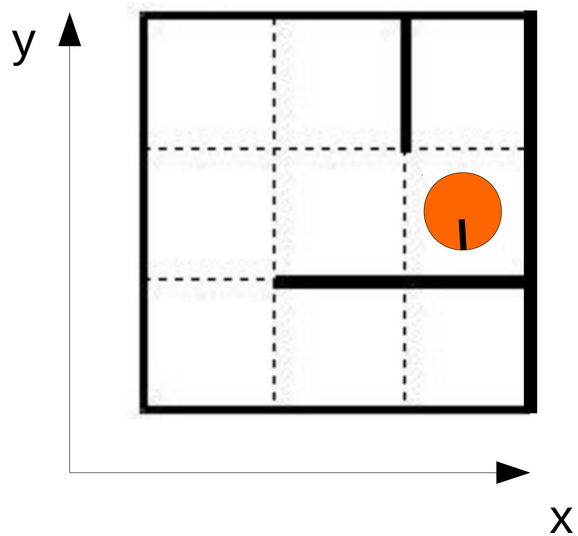
Maze World: Laberinto



- La pose estará dada por (x,y,θ). θ puede ser (up, right, down, left)
- El robot sólo ve la existencia/ausencia de las cuatro (posibles) paredes alrededor.
- ¿ Cuántos estados posibles existen para *el mundo MazeWorld* ?

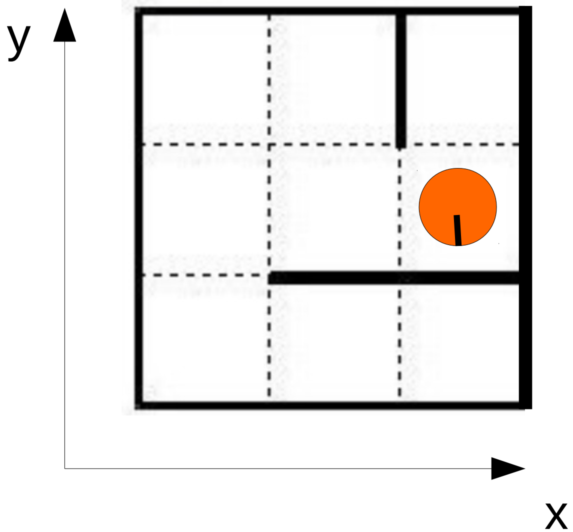
Maze World: Laberinto

- En total, son 9x4 = 36 estados posibles (9 posiciones x 4 orientaciones)
- Si el robot "observa" que tiene una pared al frente y otra a su izquierda, mientras que atrás y a su derecha no hay pared, ¿ cuál sería P(z|x,m) ?



Maze World: Laberinto

- En total, son 9x4 = 36 estados posibles (9 posiciones x 4 orientaciones)
- Si el robot "observa" que tiene una pared al frente y otra a su izquierda, mientras que atrás y a su derecha no hay pared, ¿ cuál sería P(z|x,m) ?

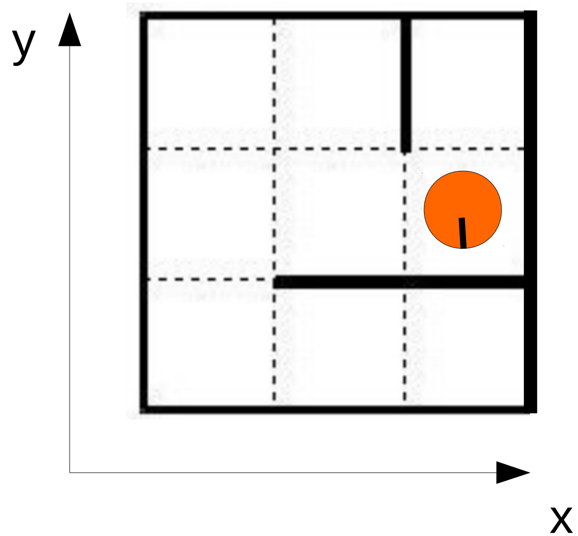


Estados posibles (x,y,θ):

(0,0,l), (0,2,u), (1,2,r), (2,1,d)

Maze World: Laberinto

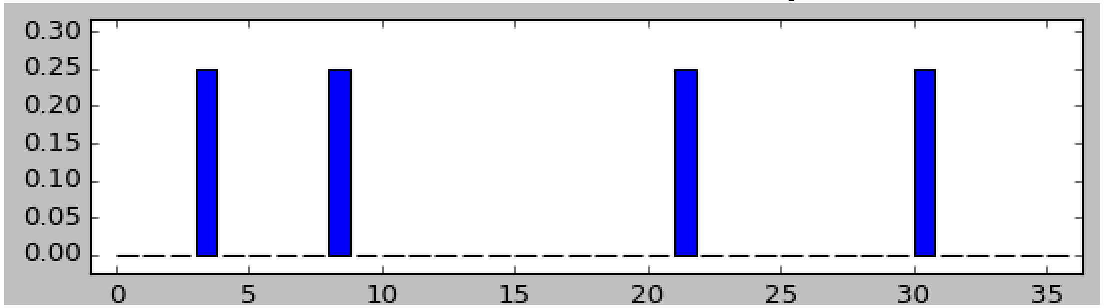
- En total, son 9x4 = 36 estados posibles (9 posiciones x 4 orientaciones)
- Si el robot “observa” que tiene una pared al frente y otra a su izquierda, mientras que atrás y a su derecha no hay pared, ¿ cuál sería P(z|x,m) ?



Estados posibles (x,y,θ):

(0,0,l), (0,2,u), (1,2,r), (2,1,d)

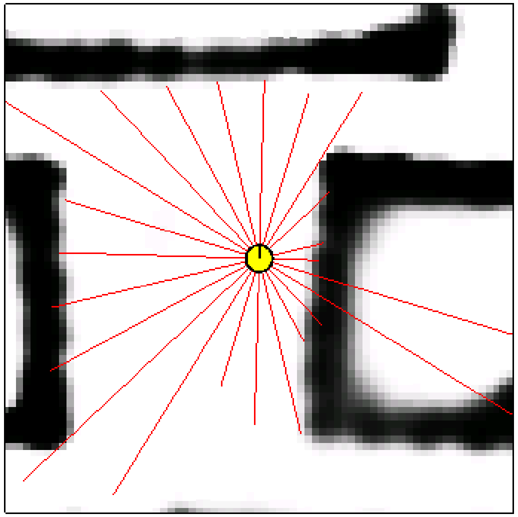
P(z=up_wall & left_wall | x, m)

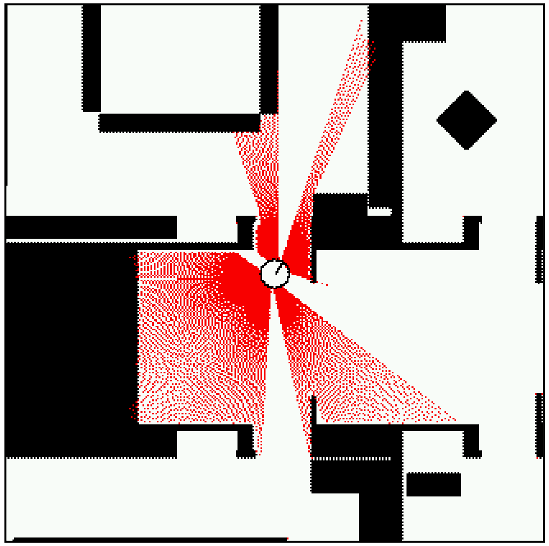


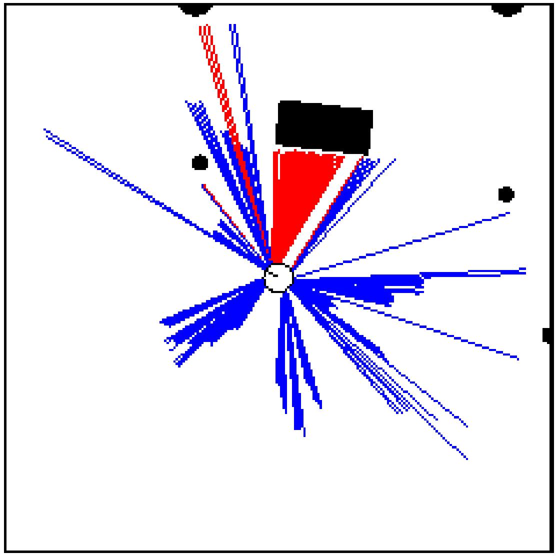
Estados totales del mundo (*x*)

Sensor Model

- En la práctica, tanto x, y y la orientación θ son continuos
- Pero la metodología es la misma: obtener la distribución de la observación z.







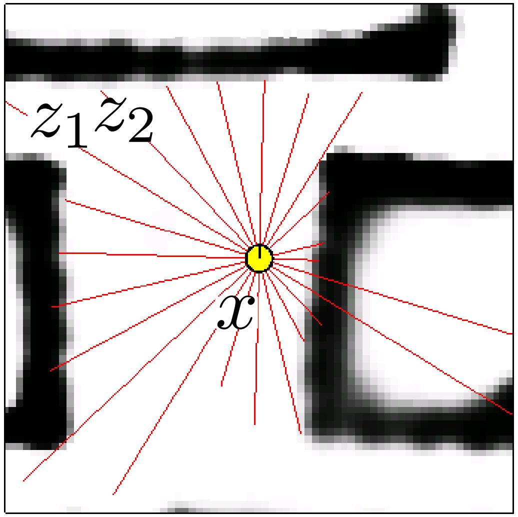
Beam-based Model

- Supondremos que disponemos de *K* mediciones (por ejemplo las de un Rangefinder)

$$z = \{z_1, z_2, \ldots, z_K\}$$

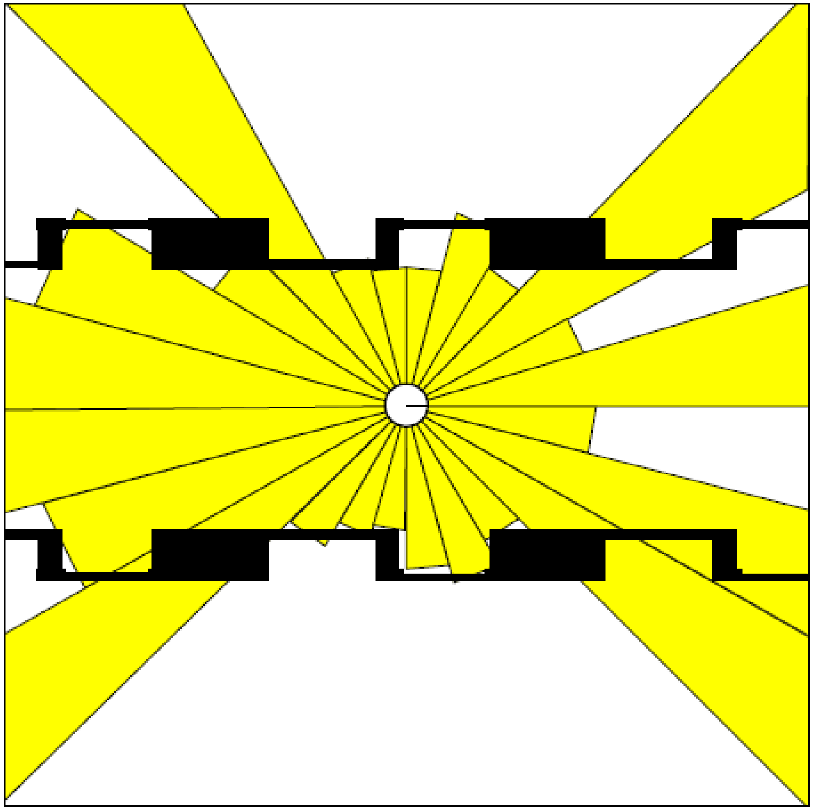
- Supondremos que las mediciones son condicionalmente independientes dada la posición del robot (x)

$$P(z|x,m) = \prod_{k=1}^{K} P(z_k|x,m)$$



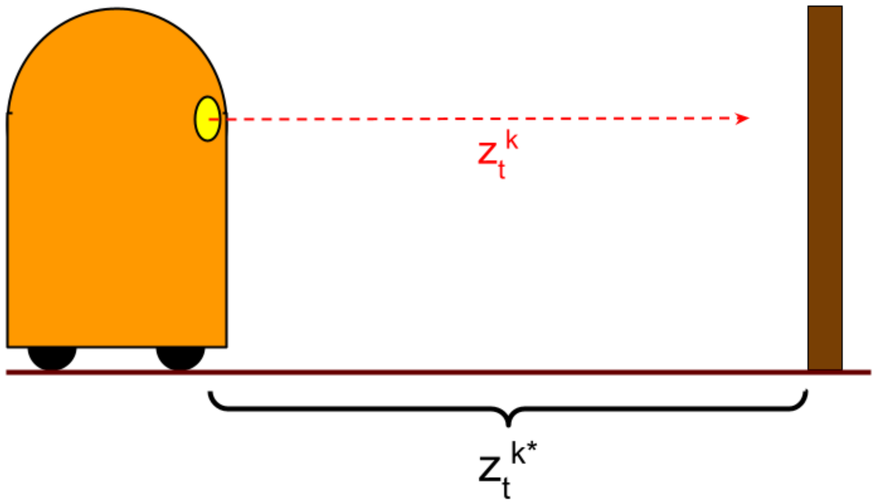
Beam-based Model

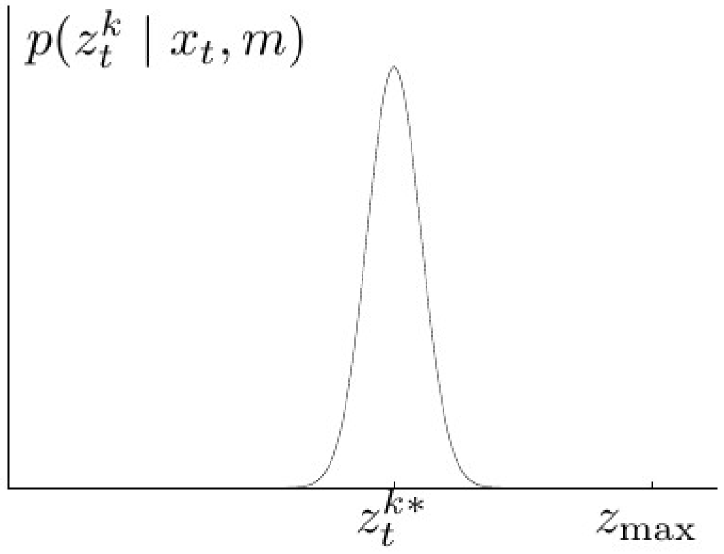
- Beam-based model modela la medición incluyendo el ruido del sensor según cuatro fuentes de reflexión:
  1) Obstáculos fijos (ej: paredes)
  2) Objetos inesperados (ej: personas)
  3) Fallas de medición (zmax)
  4) Mediciones aleatorias (ruido)
     - En medición
     - Random [0, zmax]



Beam-based Model

1) Distancia a obstáculo más ruido de medición local



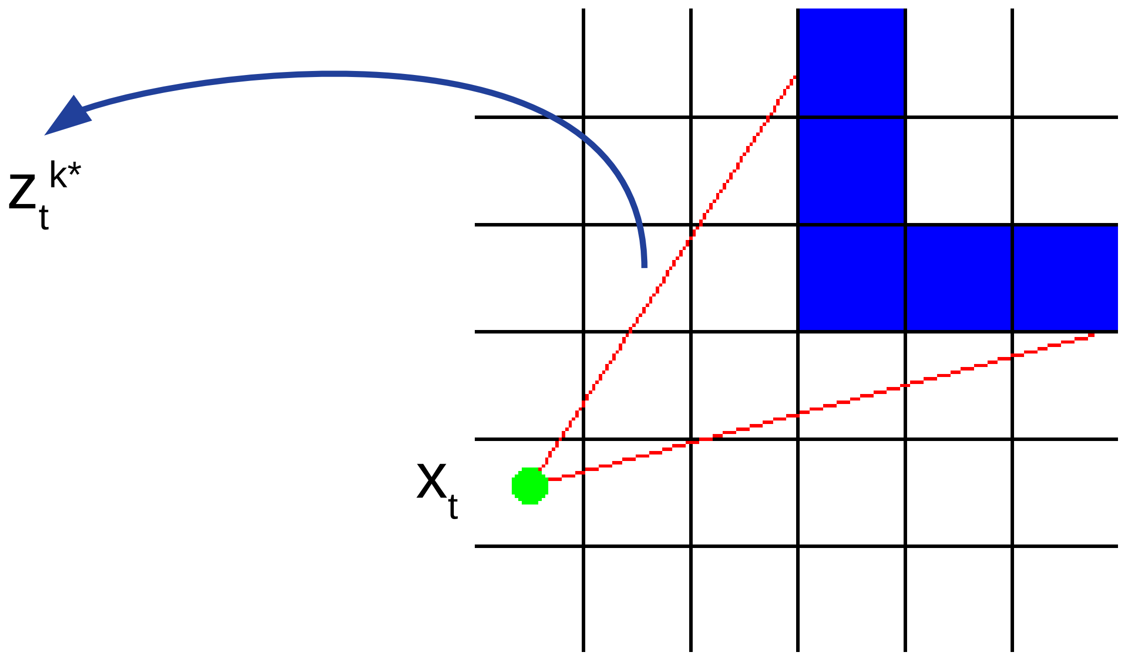


- **z_t^{k*}** es calculado a partir de ***x_t*** y ***m*** por medio del proceso de *ray casting*

Beam-based Model

1) Distancia a obstáculo más ruido de medición local

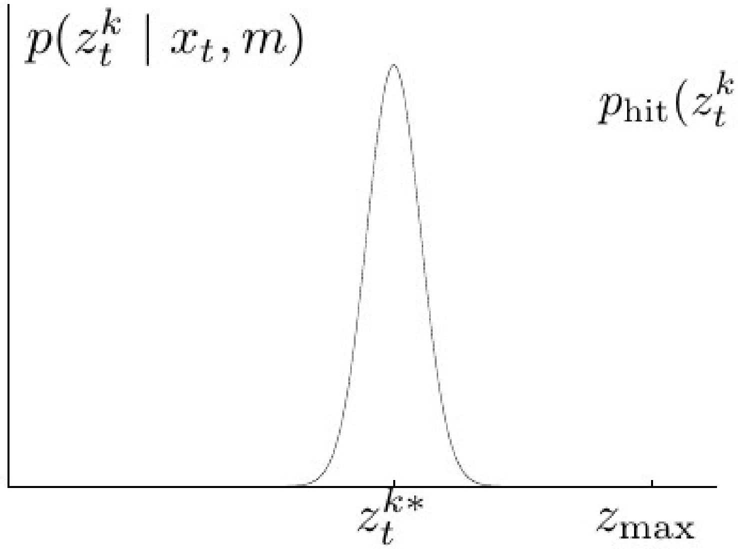
Ray Casting: proceso mediante el cual se cualcula la distancia desde el sensor hasta el obstáculo, dentro del mapa



Mapa

Beam-based Model

1) Distancia a obstáculo más ruido de medición local



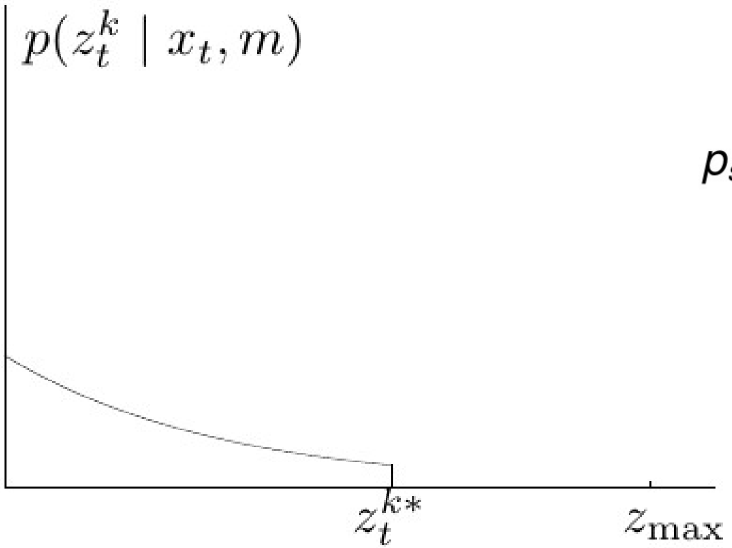
$$p_{\text{hit}}(z_t^k \mid x_t, m) = \begin{cases} \eta\, \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) & \text{if } 0 \leq z_t^k \leq z_{\max} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2) = \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}}\, e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{\text{hit}}^2}}$$

$$\eta = \left(\int_0^{z_{\max}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{\text{hit}}^2)\, dz_t^k \right)^{-1}$$

Beam-based Model

2) Obstáculos inesperados (o cortos)



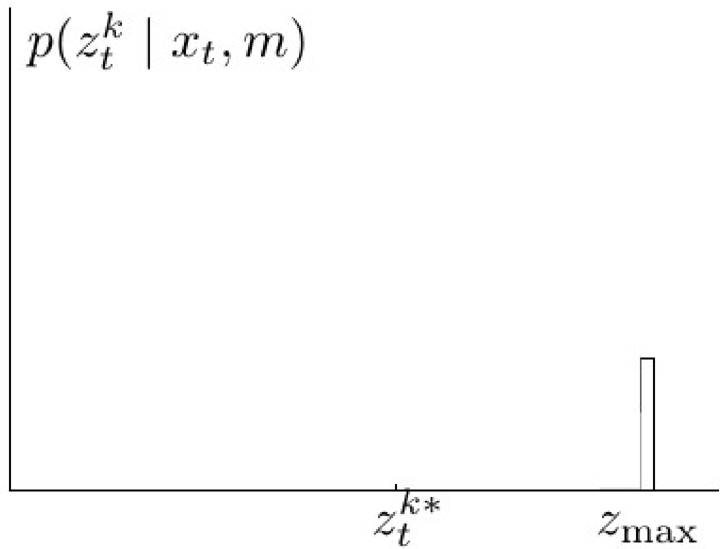
$$p_{short}(z_t^k|x_t, m) = \begin{cases} \eta \cdot \lambda_{short} \cdot e^{\lambda_{short} \cdot z_t^k} & ;\text{ si } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & ;\ \textit{otros casos} \end{cases}$$

$$\eta = \frac{1}{1 - e^{-\lambda_{\text{short}} z_t^{k*}}}$$

- Rangos siempre menores a (más cortos que) z_t^{k*}
- Verosimilitud decrece con la distancia al medir objetos inesperados

Beam-based Model

3) Fallas (zmax)

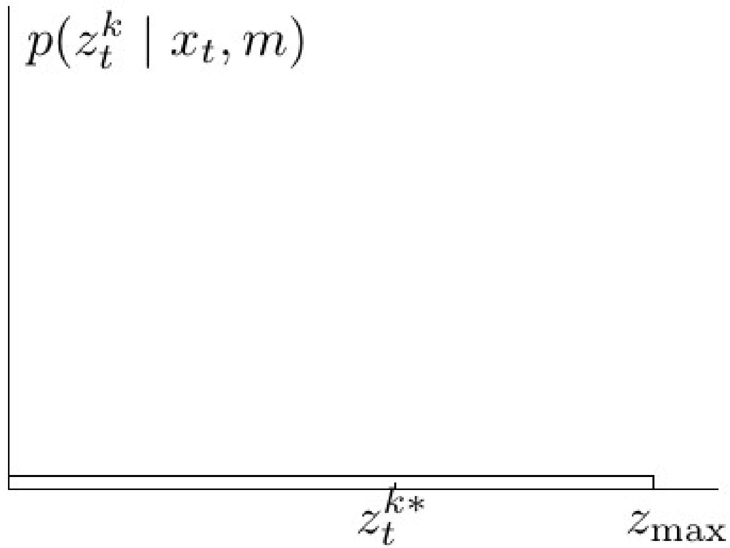


$$p_{max}(z_t^k|x_t, m) = \begin{cases} 1 & ; \text{si } z = z_{max} \\ 0 & ; \textit{otros casos} \end{cases}$$

- Medición con sonar sobre superficies en ángulos grandes
- Medición laser de objetos negros o absorventes de luz
- Medición laser de objetos bajo luz brillante
- Medición de objetos ubicados más allá de máximo alcance z_{max}

Beam-based Model

4) Medición aleatoria



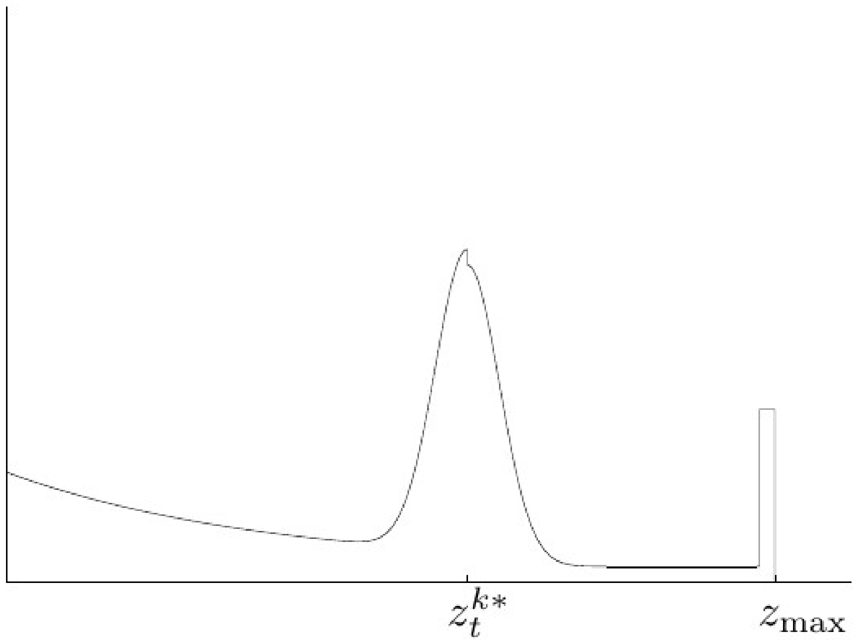
$$p_{rand}(z_t^k|x_t, m) = \begin{cases} \frac{1}{z_{max}} & ;\text{si } 0 \leq z_t^k < z_{max} \\ 0 & ;\ otros\ casos \end{cases}$$

- Errores inexplicables y aleatorios

Beam-based Model

- El resultado es la suma ponderada de todas las contribuciones

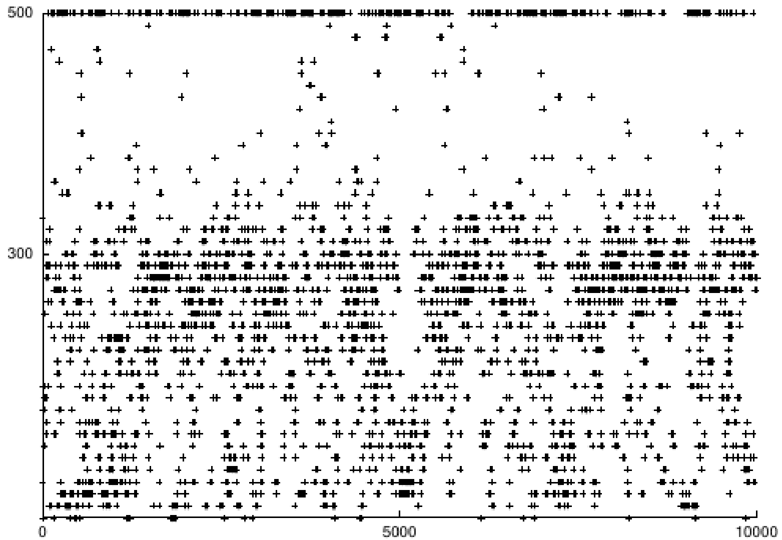
$$p(z_t^k \mid x_t, m) \quad = \quad \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\text{max}} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k \mid x_t, m) \\ p_{\text{short}}(z_t^k \mid x_t, m) \\ p_{\text{max}}(z_t^k \mid x_t, m) \\ p_{\text{rand}}(z_t^k \mid x_t, m) \end{pmatrix}$$



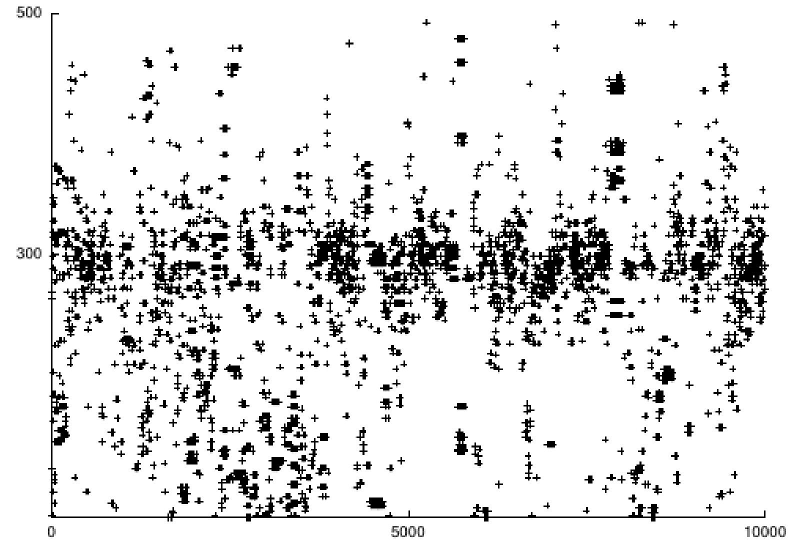
Beam-based Model

¿ Qué tan bien se ajusta nuestro modelo a datos reales ?

- Datos obtenidos por un sonar y un laser dentro de una oficina
- Alcance “real” a obstáculo: 300 [cm]
- Alcance máximo: 500 [cm]
- Número de lecturas: 10.000



Sonar



Laser

Beam-based Model

¿ Cómo determinar los parámetros de las distribuciones: z_{hit}, z_{short}, z_{max}, z_{rand}, σ_{hit} y λ_{short} ?

- **Solución 1:** Ajustar distribuciones "al ojo" hasta que sean coherentes con los datos reales
- **Solución 2:** Obtener datos y encontrar los parámetros con un algoritmo estimador de máxima verosimilitud (*ML estimator*)

$$P(Z|X, m, \Theta)$$

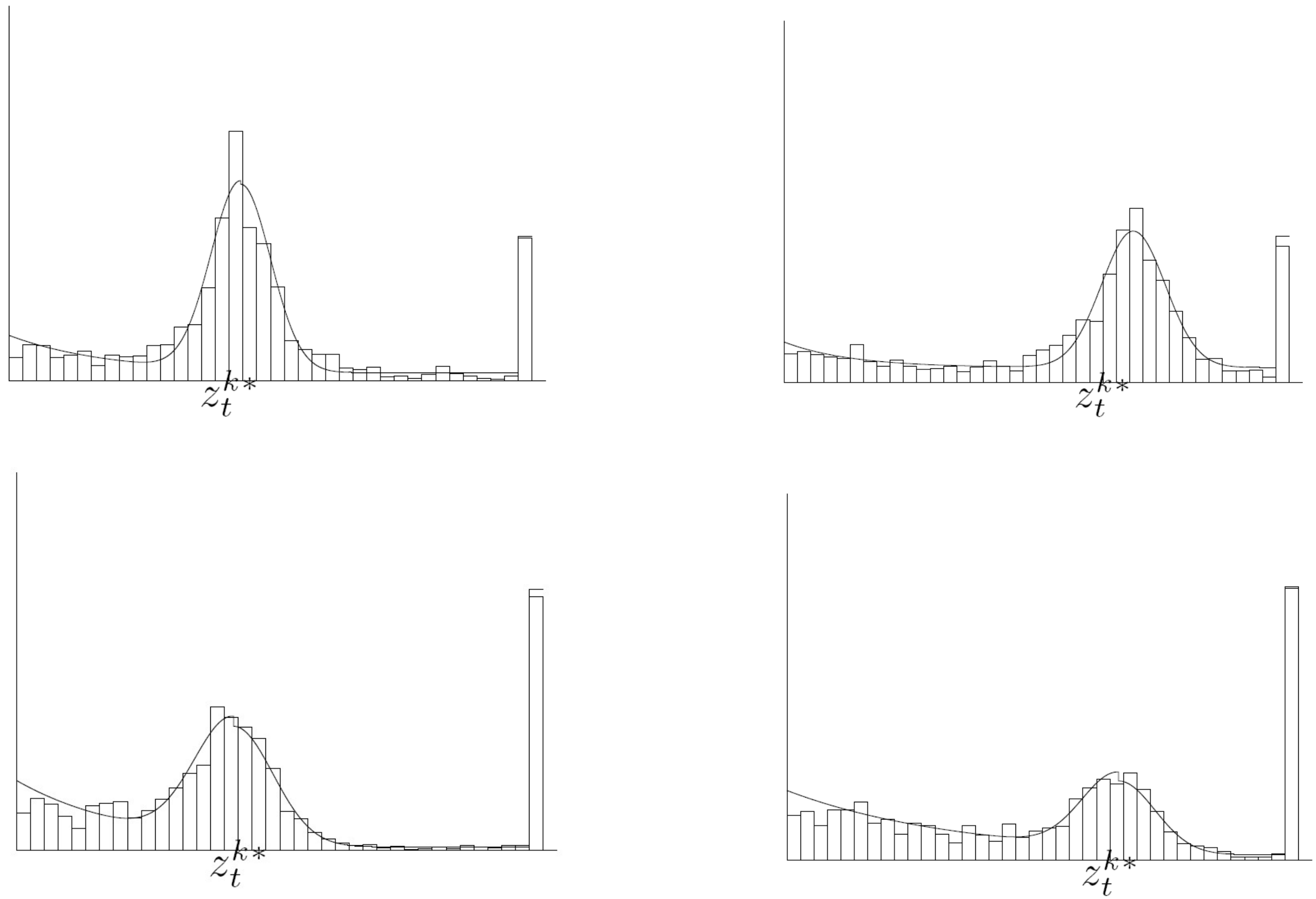
  - Datos: $Z = \{z_i\}$ y $X = \{x_i\}$
  - Mapa: m
  - Conjunto de parámetros a estimar: Θ

Beam-based Model

Aproximación de modelo Beam-based mediante estimador de ML

Laser

Sonar



300cm

400cm

Beam-based Model

- Notar que es un modelo independiente por cada “rayo” (beam)

$$P(z|x,m) = \prod_{k=1}^{K} P(z_k|x,m)$$

- Según lo anterior, el algoritmo toma la siguiente forma:

Algorithm beam_range_finder_model(z_t, x_t, m**):**

$q = 1$

for $k = 1$ *to* K *do*

compute z_t^{k*} *for the measurement* z_t^k *using ray casting*

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

$\quad + z_{\max} \cdot p_{\max}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$

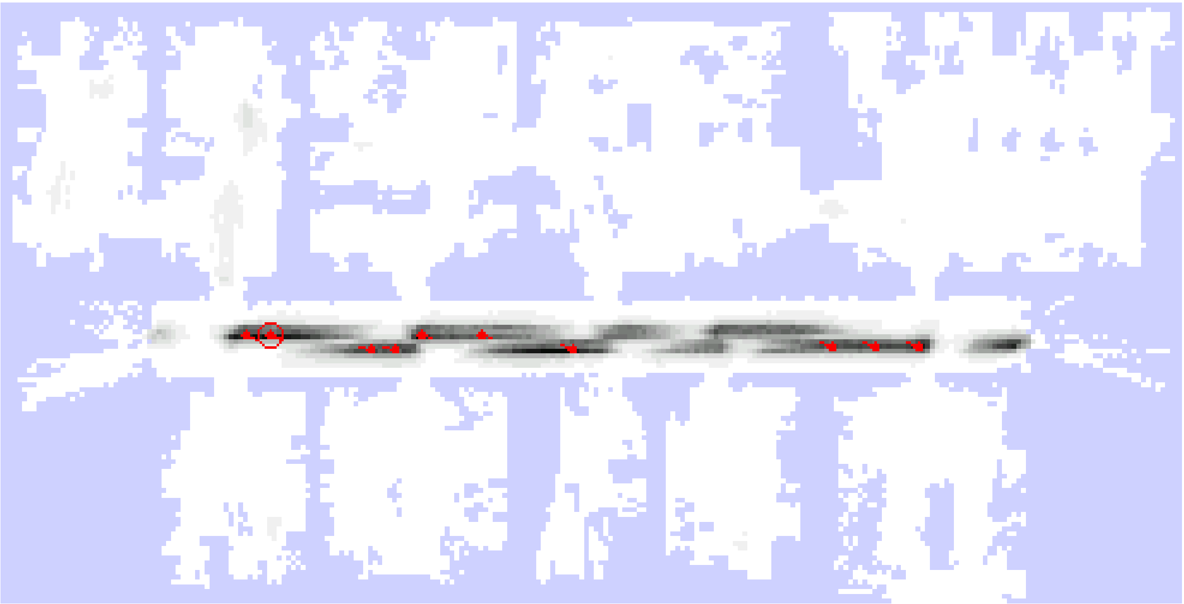
$q = q \cdot p$

return q

Beam-based Model

Cálculo de verosimilitud para todas las poses del mapa

z_t



$P(z_t|x_t,m)$

- Robot es ubicado en mapa de ocupación previamente adquirido
- Se utiliza un sensor de 180°
- Solo una medición es insuficiente para determinar pose exacta del robot (alta simetría en pasillo)

Beam-based Model - resumen

- Supone independencia condicional entre mediciones
- Modela las causas de las mediciones
  - Obstáculos (paredes), objetos móviles (personas), errores.
  - Supone independencia entre beams. Puede no ser cierto.
- Implementación
  - Aprender parámetros a partir de datos reales
  - Un modelo para cada “rayo”
  - … y potencialmente un modelo para cada distancia
- Problemas
  - Por las razones anteriores, no es eficiente y no es muy usado en la práctica, aunque provee del mayor nivel de robustez comparativa a otros modelos.
  - Las distribuciones no son “suaves” con respecto a pequeños cambios de x_t

Likelihood Fields Model

- ¿ Cómo mejorar la eficiencia del modelo Beam-based ?

  Solución 1: Subsamplear mediciones hasta obtener un número menor de beams equiespaciados

  Solución 2: Evitando usar modelos por cada ángulo y distancia!:

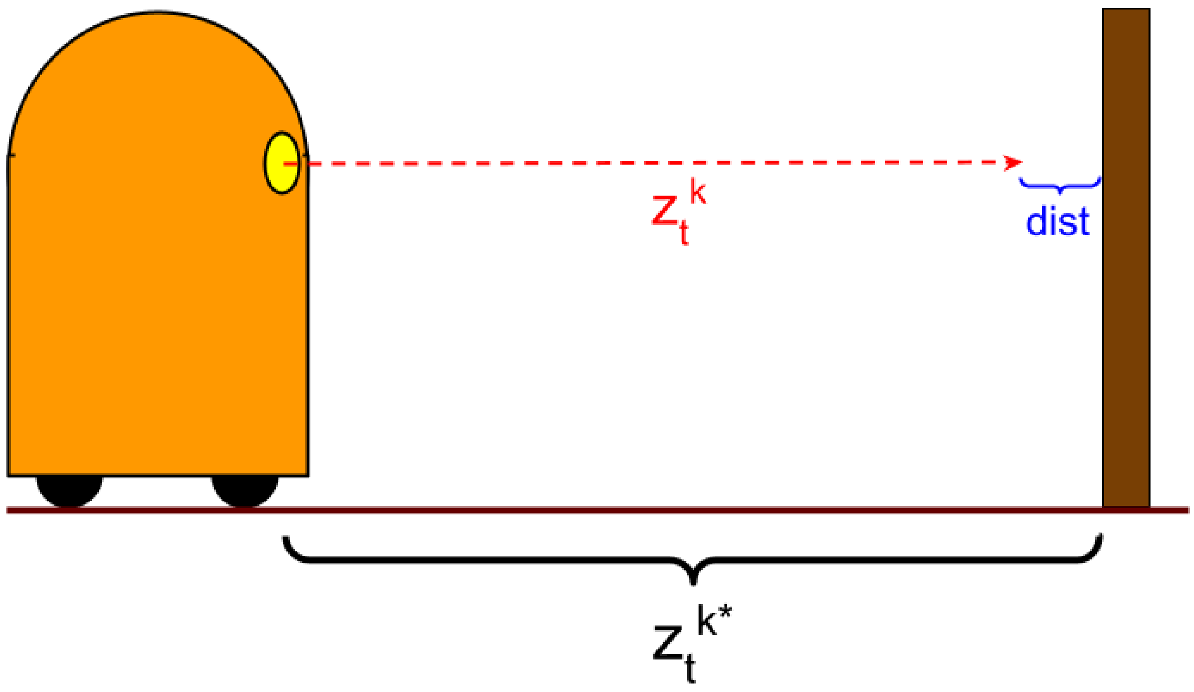
  Likelihood Fields model

Likelihood Fields Model

- Dada una medición z_t^k, se genera un modelo *ad-hoc* dado por:
  - Una distribución Normal con media igual a la distancia al obstáculo más cercano (range)
  - Una distribución Uniforme para mediciones random, y
  - Una “pequeña” masa de probabilidad para errores zmax
- Se supone nuevamente que las mediciones son condicionalmente independientes

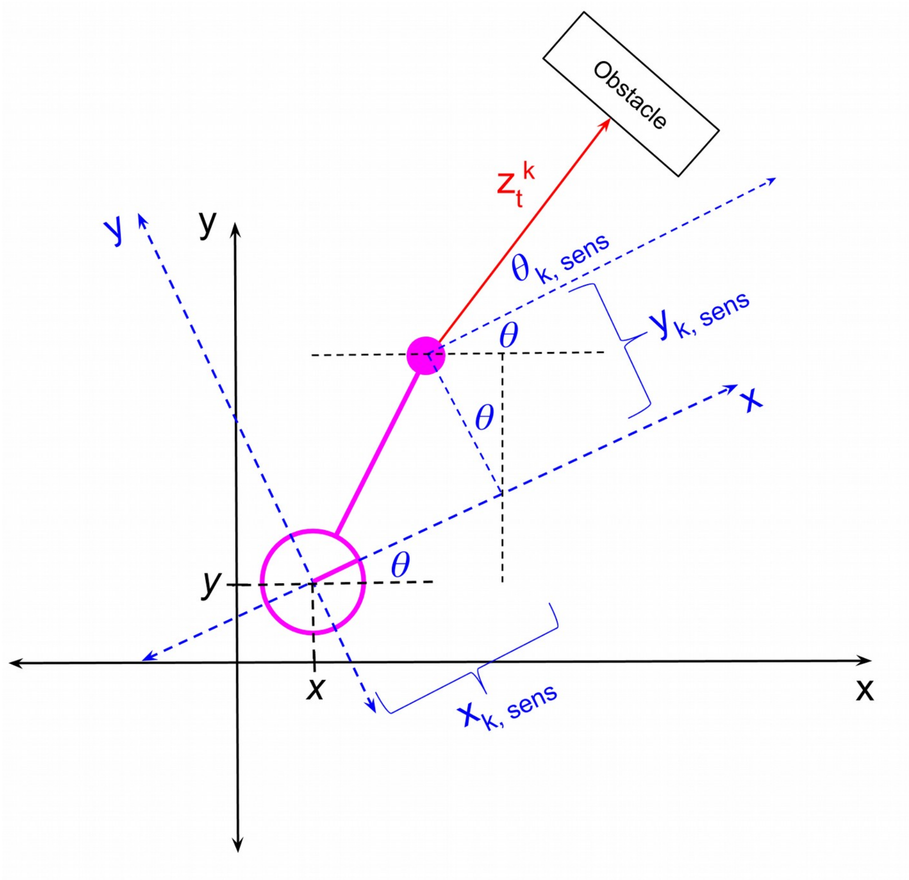
Likelihood Fields Model

- Idea: Obtener probabilidad que indique la cercanía (distancia) de la *punta del rayo* con el obtáculo más cercano



Likelihood Fields Model

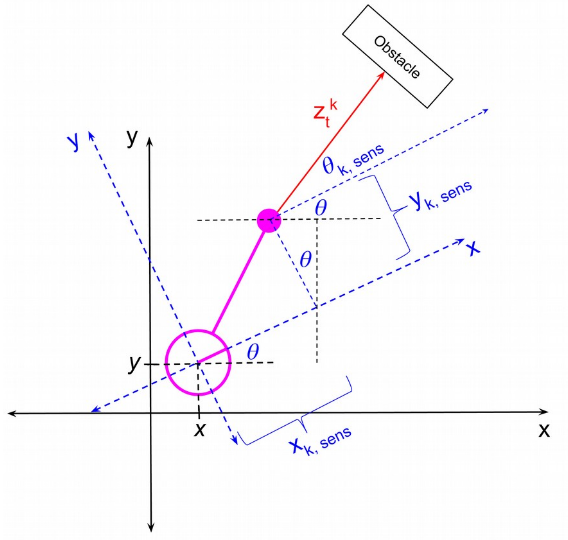
- Para determinar el obstáculo más cercano a z_t^k, primero se debe proyectar la medición desde el sistema de coordenadas del robot al sistema de coordenadas *global* del mapa



Likelihood Fields Model

- Como resultado, se obtiene la siguiente transformación:

$$\begin{pmatrix} x_{z_t^k} \\ y_{z_t^k} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{k,\text{sens}} \\ y_{k,\text{sens}} \end{pmatrix} + z_t^k \begin{pmatrix} \cos(\theta + \theta_{k,\text{sens}}) \\ \sin(\theta + \theta_{k,\text{sens}}) \end{pmatrix}$$



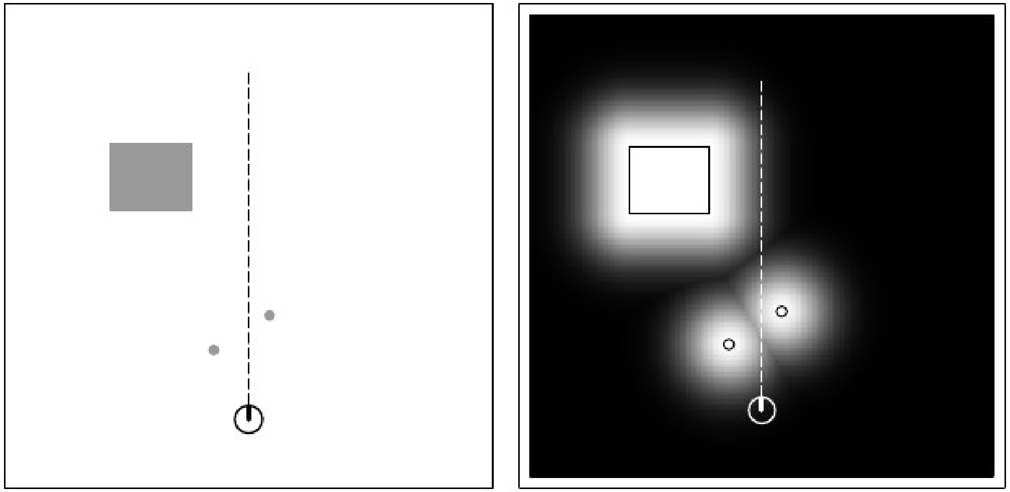
- Estas mediciones solo serán relevantes en los casos en que el sensor detecta un obstáculo
- Si $z_t^k = z_{max}$, la medición es descartada

Likelihood Fields Model

1) Ruido de medición

- Se busca en el mapa el obstáculo más cercano a $(x_{z_t^k} \;\; y_{z_t^k})^T$: *dist*
- El ruido del sensor es modelado por una Gaussiana centrada en cero, tal que:

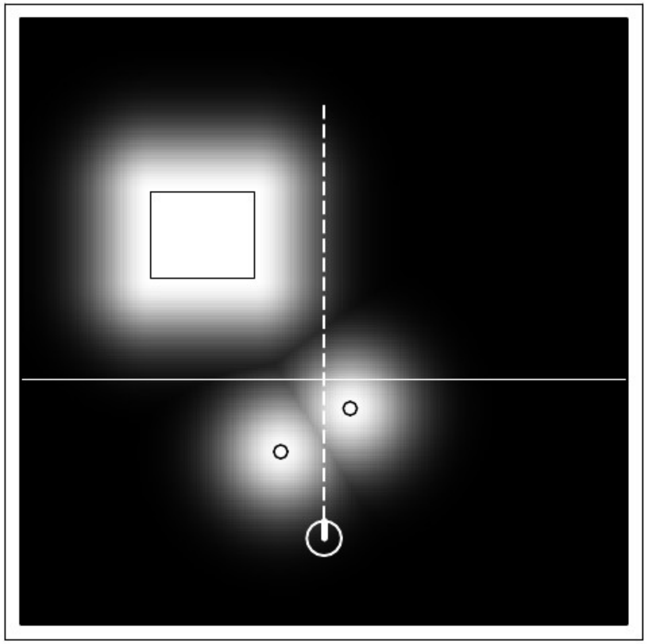
$$p_{\text{hit}}(z_t^k \mid x_t, m) \quad = \quad \varepsilon_{\sigma_{\text{hit}}}(dist)$$

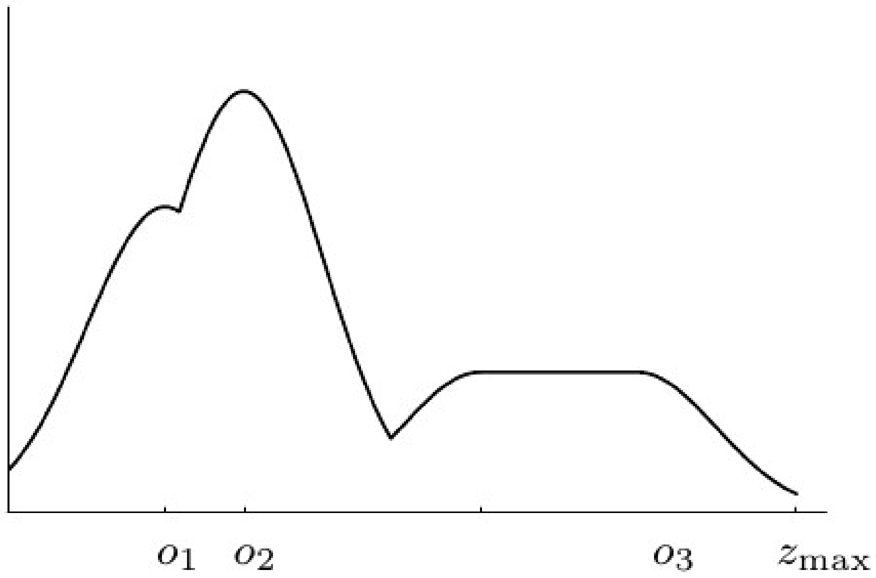


Likelihood Fields Model

1) Ruido de medición

- Bajo estas condiciones, la densidad p_{hit} se obtiene al superponer (y normalizar) el *campo de verosimilitud* con el eje del sensor





Likelihood Fields Model

2) Fallas

- Al igual que en el modelo Beam-based, se define como una distribución de masa puntual en p_{max}

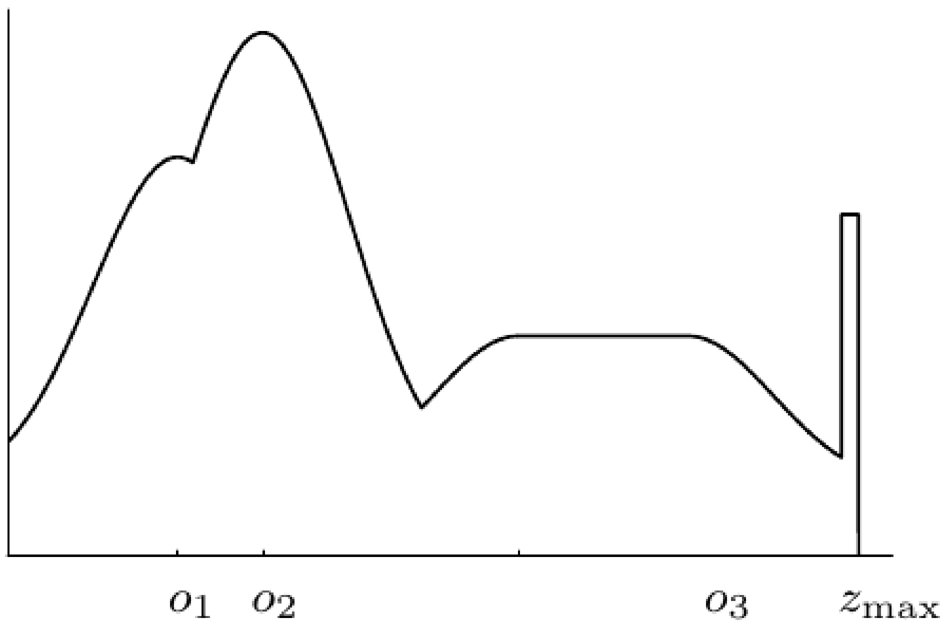
3) Mediciones aleatorias

- Distribución uniforme p_{rand}

Likelihood Fields Model

- Integrando las tres distribuciones se obtiene:

$$p(z_t^k | x_t, m) = z_{hit} \cdot p_{hit} + z_{rand} \cdot p_{rand} + z_{max} \cdot p_{max}$$



Likelihood Fields Model

- Algoritmo para el cálculo de la probabilidad de medición

Algorithm likelihood_field_range_finder_model(z_t, x_t, m)**:**

$q = 1$

for all k *do*

if $z_t^k \neq z_{\max}$

$x_{z_t^k} = x + x_{k,\text{sens}} \cos\theta - y_{k,\text{sens}} \sin\theta + z_t^k \cos(\theta + \theta_{k,\text{sens}})$

$y_{z_t^k} = y + y_{k,\text{sens}} \cos\theta + x_{k,\text{sens}} \sin\theta + z_t^k \sin(\theta + \theta_{k,\text{sens}})$

$dist \;\; = \min\limits_{x',y'} \left\{ \sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} \;\middle|\; \langle x', y' \rangle \text{ occupied in } m \right\}$

$q = q \cdot \left(z_{\text{hit}} \cdot \mathbf{prob}(dist \;\,, \sigma_{\text{hit}}) + \frac{z_{\text{random}}}{z_{\max}} \right)$

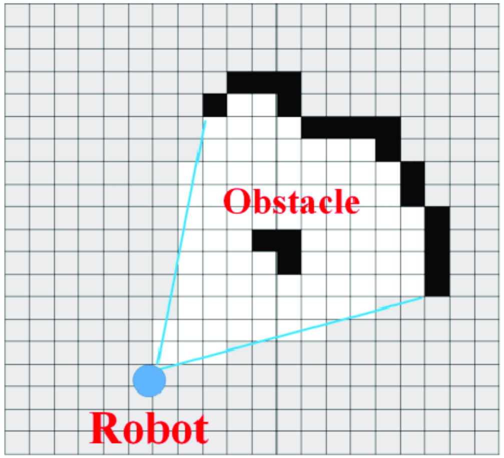
return q

Likelihood Fields Model - resumen

- No-paramétrico, no es necesario encontrar modelos. La lectura es el modelo en sí mismo
- Eficiente
- Produce distribuciones “suaves” con respecto a cambios de *x*, a diferencia de Beam-based model
- Problemas:
  - Asume ambiente estático, no modela causas de detecciones *cortas* (ej: personas)
  - Como consecuencia de la búsqueda del obstáculo más cercano, trata los sensores como si pudieran ver a través de las paredes

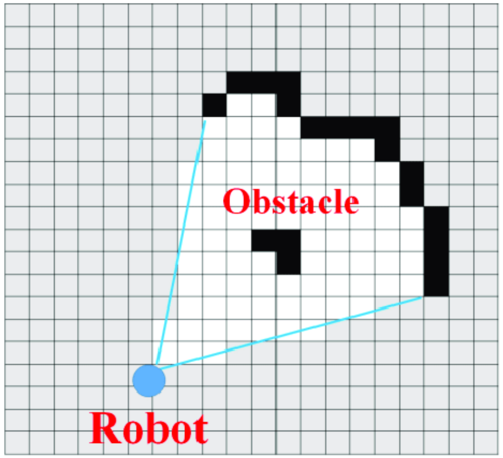
Map Matching Model

- Supongamos que podemos obtener un mini mapa local
  - Transformamos las mediciones z_t^k en un *mini* mapa m_{local}
  - Consideraremos un mapa en formato *Occupancy Grid Map*, que asigna valores de ocupación a las coordenadas (x, y) según algunos de los siguientes estados: *Ocupado*, *Libre* o *Desconocido*
- Mini-mapa debe tener las mismas características del mapa global (resolución, definición ocupado / desocupado / desconocido, etc).

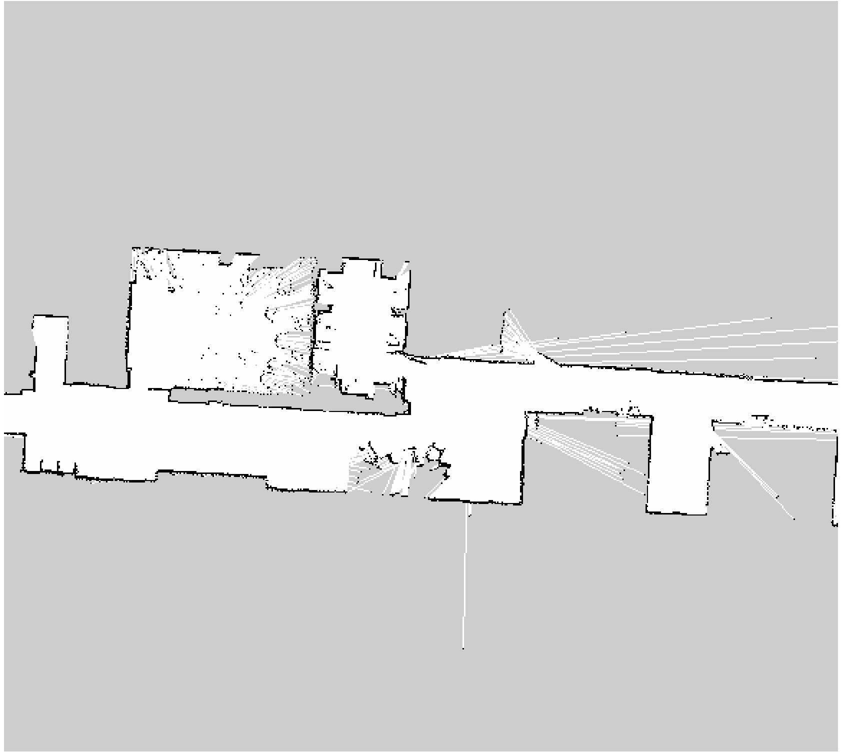


Map Matching Model

- Estrategia de solución: buscar la correlación entre el mapa local y el mapa global



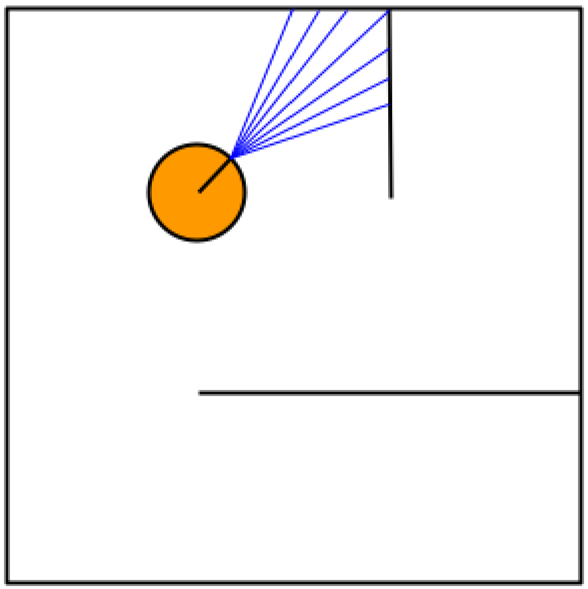
Mapa Local



Mapa Global

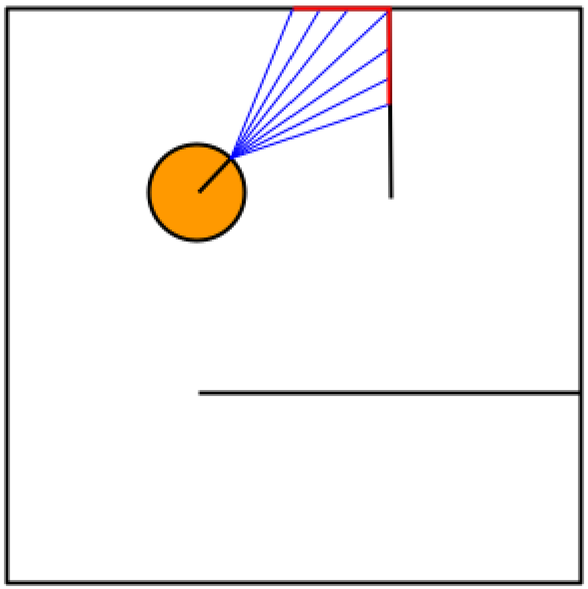
Map Matching Model

- Cálculo de correlación entre mapas local y global



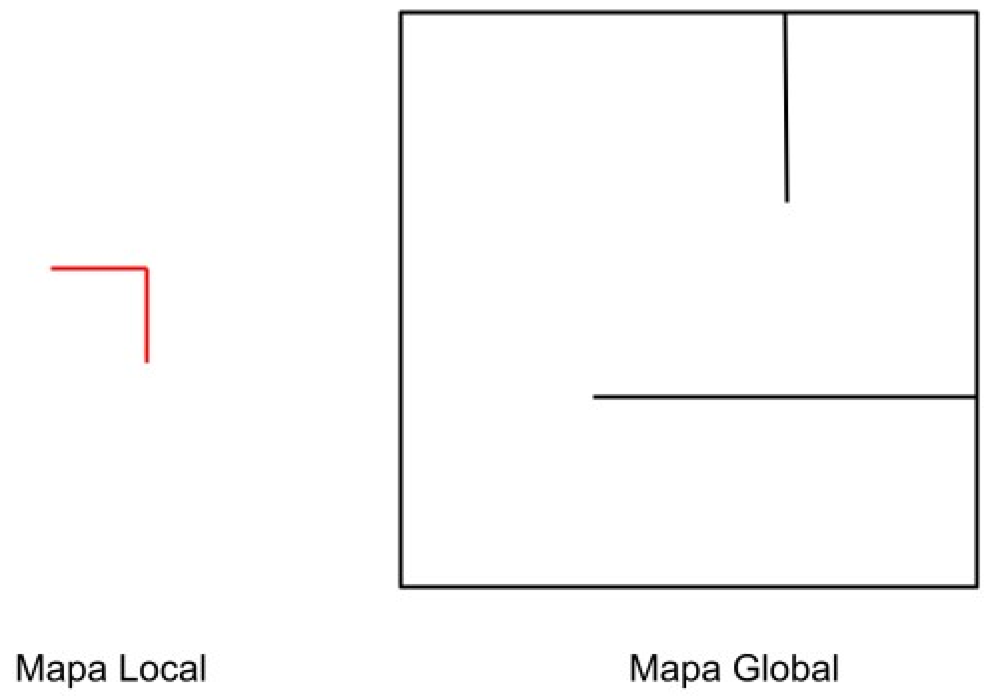
Map Matching Model

- Cálculo de correlación entre mapas local y global



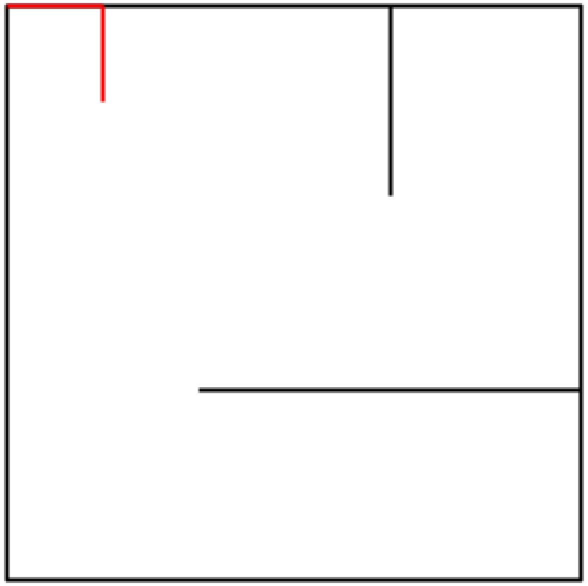
Map Matching Model

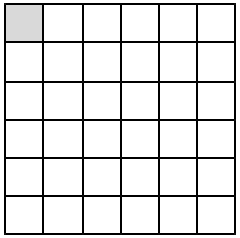
- Cálculo de correlación entre mapas local y global



Map Matching Model

- Cálculo de correlación entre mapas local y global

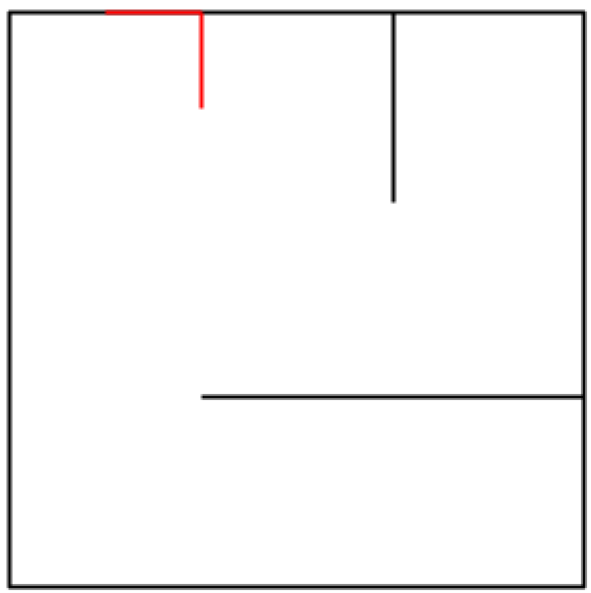


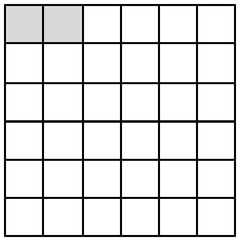


Matriz de Correlación

Map Matching Model

- Cálculo de correlación entre mapas local y global

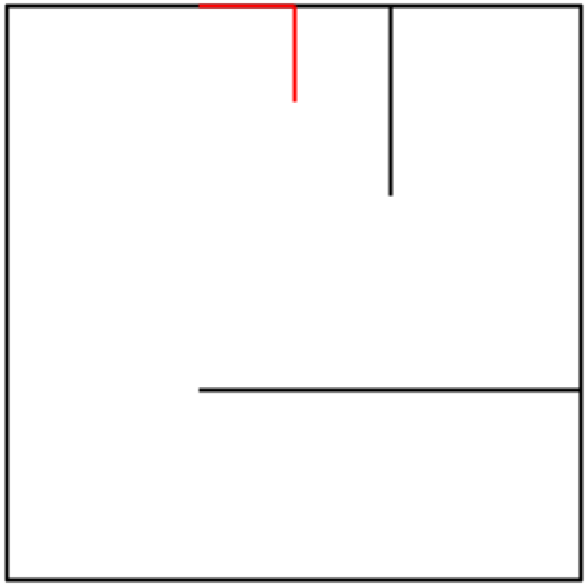


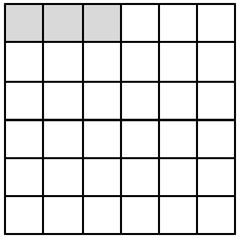


Matriz de Correlación

Map Matching Model

- Cálculo de correlación entre mapas local y global

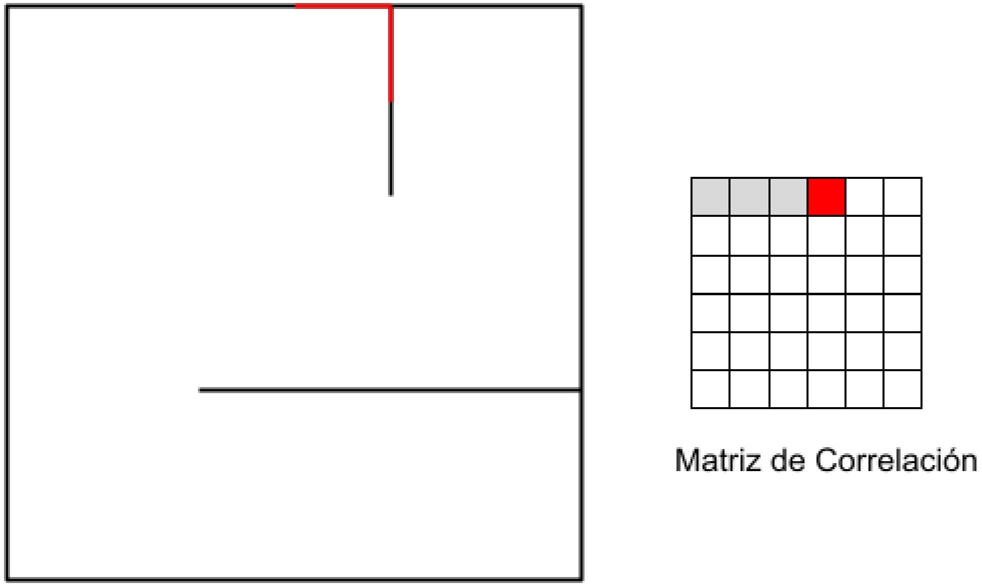




Matriz de Correlación

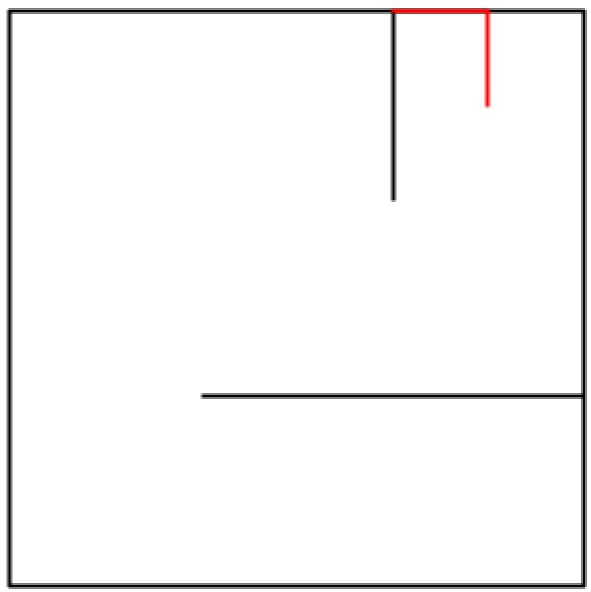
Map Matching Model

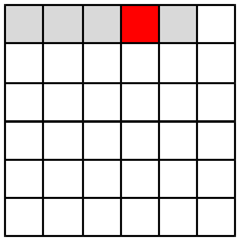
- Cálculo de correlación entre mapas local y global



Map Matching Model

- Cálculo de correlación entre mapas local y global

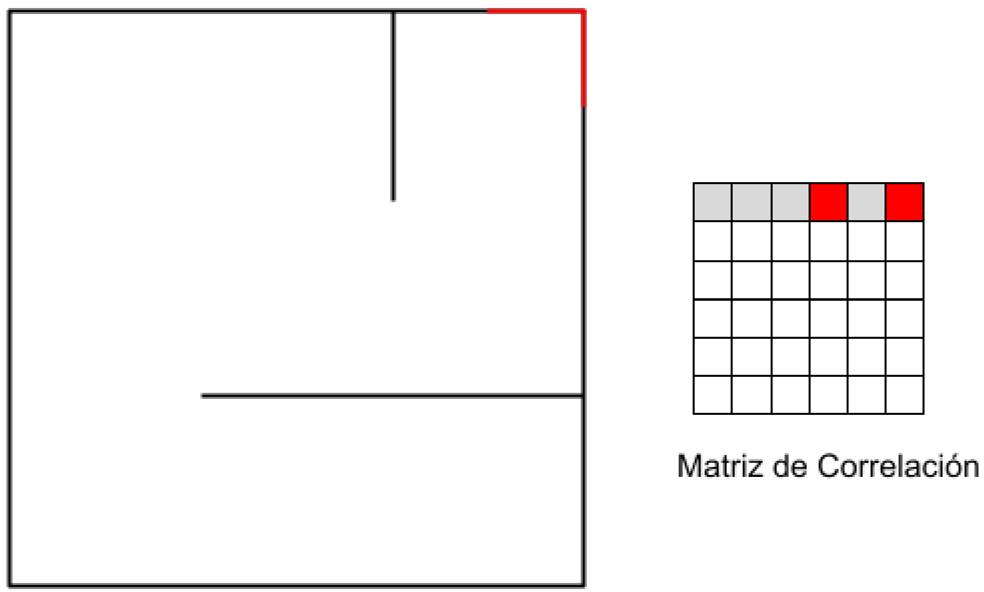




Matriz de Correlación

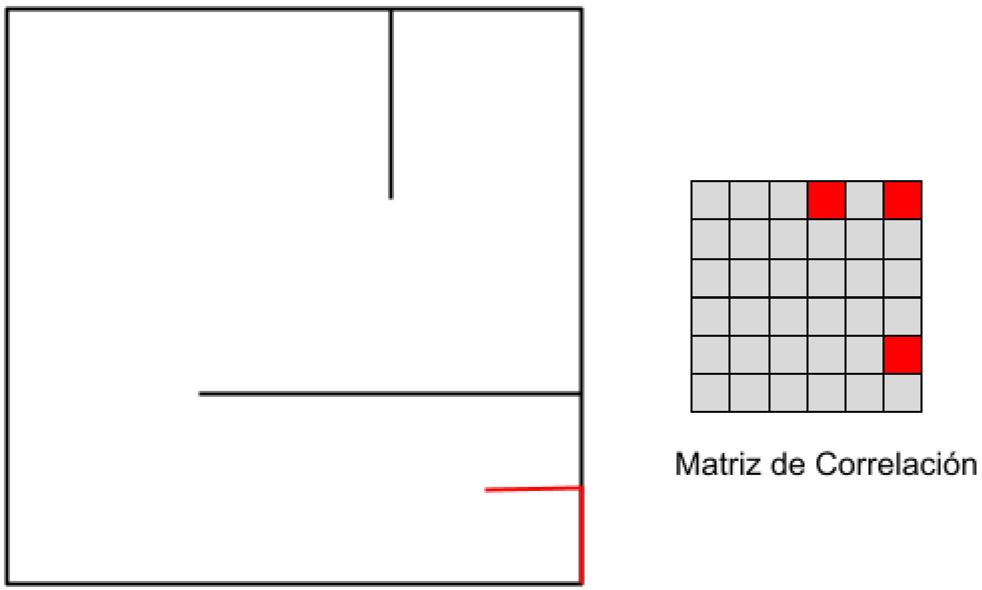
Map Matching Model

- Cálculo de correlación entre mapas local y global



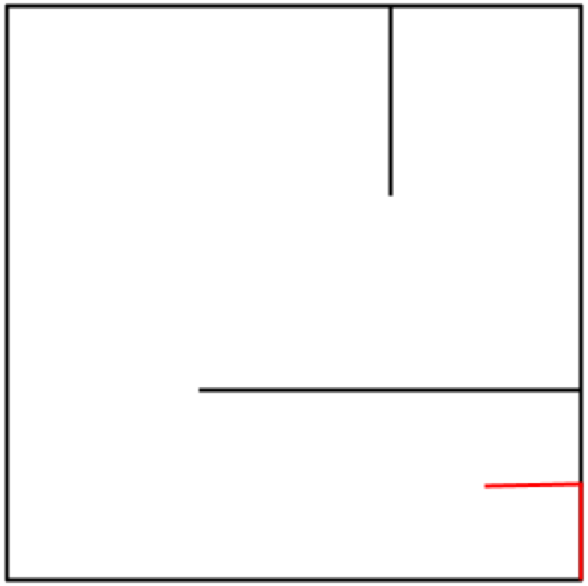
Map Matching Model

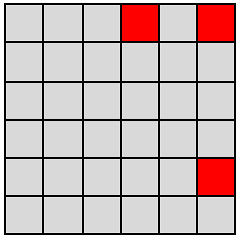
- Cálculo de correlación entre mapas local y global



Map Matching Model

- Cálculo de correlación entre mapas local y global





Matriz de Correlación

- La correlación debe ser calculada para distintas posiciones y ángulos

Map Matching Model

- Para encontrar P(m_{local}|x_t, m), se utiliza una función de correlación:

$$\rho_{m,m_{\text{local}},x_t} = \frac{\sum_{x,y}(m_{x,y}-\bar{m})\cdot(m_{x,y,\text{local}}(x_t)-\bar{m})}{\sqrt{\sum_{x,y}(m_{x,y}-\bar{m})^2\sum_{x,y}(m_{x,y,\text{local}}(x_t)-\bar{m})^2}}$$

$$\bar{m} = \frac{1}{2N}\sum_{x,y}(m_{x,y}+m_{x,y,\text{local}})$$

N: número de pares de celdas a correlacionar entre mapa local y global

- Dado que la función de correlación varía entre -1 y 1, se define:

$$p(m_{\text{local}} \mid x_t, m) = \max\{\rho_{m,m_{\text{local}},x_t}, 0\}$$

- Además, si el mapa local fue generado a partir de una medición:

$$p(z_t|x_t, m) = p(m_{local}|x_t, m)$$

Map Matching Model

- **Ejemplo 1:** Suponga que se desea comparar el siguiente conjunto de celdas, cuyo estado puede tomar el valor 1 para *ocupado*, o 0 para *desocupado, y tal que:*

$m = [0, 1] \quad :$

0	1

$m_{local} = [0, 1] \quad :$

0	1

De los datos se infiere que N = 2, entonces:

$$\overline{m} = \frac{1}{2 \cdot 2} \cdot [(0 + 0) + (1 + 1)] = 0.5$$

$$\rho_{m, m_{local}, x_t} = \frac{[(0 - 0.5) \cdot (0 - 0.5) + (1 - 0.5) \cdot (1 - 0.5)]}{\sqrt{[(0 - 0.5)^2 + (1 - 0.5)^2] \cdot [(0 - 0.5)^2 + (1 - 0.5)^2]}} = 1$$

Map Matching Model

- **Ejemplo 2:** Suponga que se desea comparar el siguiente conjunto de celdas, cuyo estado puede tomar el valor 1 para *ocupado*, o 0 para *desocupado, y tal que:*

$m = [0, 1]$:

0	1

$m_{local} = [1, 0]$:

1	0

De los datos se infiere que N = 2, entonces:

$$\overline{m} = \frac{1}{2 \cdot 2} \cdot [(0+1) + (1+0)] = 0.5$$

$$\rho_{m,m_{local},x_t} = \frac{[(0-0.5)\cdot(1-0.5) + (1-0.5)\cdot(0-0.5)]}{\sqrt{[(0-0.5)^2 + (1-0.5)^2]\cdot[(1-0.5)^2 + (0-0.5)^2]}} = -1$$

Map Matching Model

- En la práctica, ambos mapas son "suavizados" (Gaussian blur) para no tener correlaciones discontinuas.
- Muy usados en mapas métricos con sensores laser (Rangefinder)
  - Simple de calcular
  - Intuitivo
  - Considera el espacio libre a diferencia del modelo Likelihood-based

Bibliografía

- ***Probabilistic Robotics***, Thrun, S., Burgard, W., Fox, D.