Práctica 10

Localización de robots

10.1. Objetivo

Conocer e implementar una técnica de localización utilizada en aplicaciones reales, específicamente para los robots *Rhino* y *Minerva* en los museos *Deutsches Museum Bonn* y *National Museum of American History*, respectivamente [1]. Se utilizará Processing como herramienta de visualización de la técnica de localización.

10.2. Introducción

La localización robótica ha sido considerado como uno de los problemas fundamentales de la robótica. El objetivo de la localización es lograr estimar la posición del robot en un ambiente, apoyándose en un mapa de éste y con lecturas de sensores.
Las técnicas que se han desarrollado hasta la fecha tratan de resolver uno de los dos tipos de localización que hay:

- *Localización local o rastreo.* Estas técnicas tratan de compensar errores odométricos durante el movimiento del robot. Son técnicas auxiliares que refinan la estimación que se tiene de la posición del robot en el entorno todo el tiempo, si la pierden es casi imposible volver a recuperarla.

- *Localización global.* Estas técnicas están diseñadas para encontrar la posición estimada del robot globalmente, es decir, no es necesario tener un aproximado inicial de su posición. Estas técnicas pueden resolver el problema de localizar al robot al momento de encenderlo, al igual que permiten que se lleve al robot a una posición aleatoria del entorno durante su operación y recuperar su posición.

Como se podrán dar cuenta, las técnicas de localización global son más poderosas que las locales. Para esta práctica desarrollaremos una técnica de localizacion global de Markov.

10.3. Localización de Markov

La localización de Markov utiliza un sistema probabilístico que mantiene una distribución de probabilidad de la posición del robot sobre todo el entorno. Es decir, la probabilidad de que el robot se encuentre en cada posición del entorno en un tiempo dado. Por

[1] [Dieter Fox (1999)]

ejemplo, el robot puede iniciar con una distribución de probabilidad uniforme representando que no tiene idea de dónde se encuentra en el entorno, esto es, cada posición en el entorno tiene la misma probabilidad de que el robot se encuente en ella. En el caso en el que el robot esté muy seguro de su posición, la distribución de probabilidad se convierte en una distribución unimodal centrada en la posición del robot.

10.3.1. Ejemplo

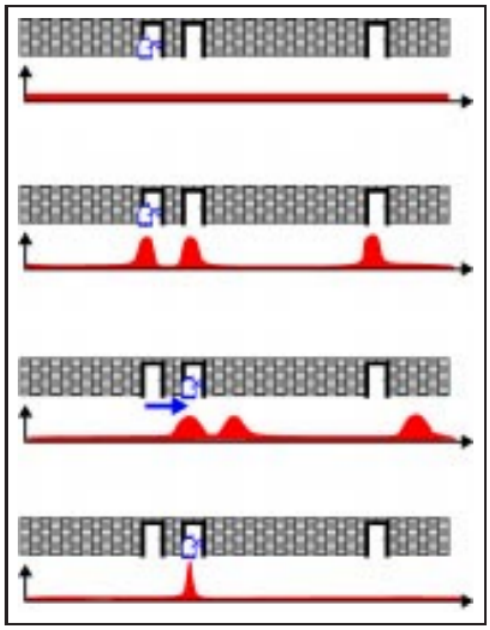

Figura 10.1: La idea básica de localización de Markov, entorno. La gráfica de color rojo representa la distribución de probabilidad.

Veremos un ejemplo sencillo para ilustrar los conceptos de la localización de Markov. Consideremos el entorno mostrado en la figura 10.1. Para simplificar, asumamos que el robot sólo puede moverse en una dimensión (enfrente-atrás).

Ahora, supongamos que posicionamos el robot en algún lugar aleatorio del entorno, pero no le informamos al robot cuál es su posición. La localización de Markov representa este estado de "confusión" como una distribución de probabilidad uniforme sobre todo el conjunto de posibles posiciones del entorno, como lo muestra la primer gráfica de la figura 10.1. Asumamos que el robot hace una medición con sus sensores y determina que está al lado de una puerta. La localización de Markov modifica la distribución de probabilidad de tal manera que las posiciones que se encuentran a un lado de puertas tengan mayor probabilidad, esto queda ilustrado en la segunda gráfica de la figura 10.1. Notemos que la distribución resultante es multimodal([2]), ya que la información obtenida por los sensores es insuficiente para determinar exactamente la posición del robot. También notemos que las posiciones que no se encuentran a un lado de una puerta aún tienen una probabilidad mayor que cero, esto es porque las mediciones de los sensores contienen ruido. Ahora, supongamos que el robot se mueve un metro hacia el frente. La localización de Markov incorpora este movimiento para que la distribución de probabilidad cambie como se representa en la tercer gráfica de la figura 10.1. Finalmente, asumamos que el

[2]Multimodal: Que tiene varios puntos máximos.

robot vuelve a tomar una medición con sus sensores. Incorporando esta información a la obtenida anteriormente, vemos cómo la distribución de probabilidad cambia (cuarta gráfica de la figura 10.1) para asignar una alta probabilidad a la posición del robot, mientras que todas las demás posiciones tienen una probabilidad casi nula.

10.4. Desarrollo

El desarrollo de la práctica se dividirá en dos partes: la simulación del ambiente en donde se encuentra el robot y del robot, y la implementación del modelo de Markov en este ambiente.

10.4.1. Simulación del ambiente y del robot

Primero, iniciemos con el contenido y especificación del mundo en el cual se moverá el robot. Este mundo contendrá obstáculos y será rectangular (imagínense un cuarto o una oficina), las paredes serán obstáculos y cualquier otra cosa que se pueda encontrar en un cuarto u oficina normal: sillas, mesas, escritorios, etc. Será representado por una matriz de celdas, donde cada celda será cuadrada, podrá ser obstáculo o no y tendrá la longitud de un lado especificado en *cm*.

El robot podrá ocupar cualquier posición del plano de Processing [3], no estará necesariamente en el centro de alguna celda. De hecho, se necesitará una función que, dado la posición del robot, nos permita saber dentro de que celda se encuentra. Esta función nos será útil para la implementación del modelo de Markov discretizado.

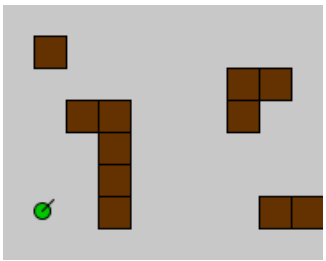

Figura 10.2: Representación del ambiente.

Movimiento del robot

El robot podrá moverse libremente en el ambiente, salvo que se encuentre de frente un obstáculo, o los bordes del mundo ya que son considerados paredes. Este movimiento será controlado poor el usuario, ya sea que utilizen las flechas direccionales de su teclado, alguna otra combinación de teclas, botones en la interfaz, o el mecanismo que ustedes prefieran. Lo importante es que el robot pueda moverse en línea recta y girar.

El usuario deberá poder especificar qué tanto se moverá el robot al momento de pulsar la tecla o botón requerido, ya sea si se desea cambiar de dirección o moverse en línea recta. Esto será muy importante, ya que necesitan saber la distancia real recorrida y la distancia real de giro para simular las medidas de los sensores.

[3] Cabe resaltar que para Processing el eje *y* se encuentra invertido, esto es, si aumentan el valor de *y* en el código, en la visualización esto se reflejará como si estuvieran caminando hacia abajo.

Simulación de sensores

Nuestro robot tendrá tres sensores:

1. láser: mide la distancia desde el robot hacia el primer obstáculo en línea recta del sensor. El robot tendrá 8 de estos, ubicados cada 45°.
2. odométrico: mide la distancia recorrida por el robot (en cm).
3. de giro: mide el giro del robot en grados.

Láser Como las mediciones de sensores en la vida real no son exactos, pueden cambiar debido a los materiales donde se refleja el láser, utilizaremos ruido gaussiano para simular este error.

Para cada sensor se necesitarán dos parámetros: la media (μ_{laser}) y la desviación estándar (σ_{laser}). La desviación estándar es un parámetro que tú debes fijar, éste representa la cantidad de ruido que se introducirá a la medición. Y la media estará dada por la distancia real medida desde el robot hacia el primer obstáculo en línea recta en la dirección del sensor.

Para simular la medición del sensor con ruido, se hará lo siguiente:

1. Obtener un número aleatorio utilizando `nextGaussian()` de la clase `java.util.Random`.
2. Multiplicar el número aleatorio obtenido por la desviación estándar (σ_{laser}).
3. Sumar a la media (μ_{laser}) el resultado anterior.

Odométrico La manera de simular la medición con ruido de éste sensor es casi idéntica al sensor láser, sólo que en lugar de medir la distancia hacia el obtáculo más cercano usaremos la distancia que el usuario proporcionó para mover al robot. Por ejemplo: Si el usuario decidió mover al robot $10cm$, la media ($\mu_{odometrico}$) sería $10cm$ y la desviación estándar ($\sigma_{odometrico}$) un parámetro especificado por ti.

De giro Similarmente al sesor odométrico, la media ($\mu_{de\ giro}$) sería los grados reales girados y la desviación estándar ($\sigma_{de\ giro}$) un parámetro especificado por ti.

10.4.2. Implementación del modelo de Markov

Representación interna del mundo El modelo de Markov en el que se basa esta práctica utiliza un modelo del mundo discretizado (como el que usamos para definir área libres y obstáculos). *Dentro de la mente del robot* el mundo será representado por una matriz de tres dimensiones, ya que tenemos tres grados de libertad para el movimiento del robot: $< x, y, \theta >$, donde θ es el ángulo hacia donde está viendo el robot. Estas tres direcciones estarán discretizadas.

Recapitulando, en la mente del robot el mundo será representado por un arreglo tridimensional $< x, y, \theta >$ donde en las coordenadas $< x, y >$ se tendrá la representación del mundo con celdas, como se especificó al incio de la sección pasada. Si el robot se

encuentra dentro de una celda, x y y serán las coordenadas del centro de ésta. Y en cuanto al eje θ, θ tomará valores desde 0° hasta 315°, en aumentos de 45°. Para cada uno de esos ángulos se tendrá una representación del plano $< x, y >$ pero con el robot mirando en la dirección discretizada θ. Como puedes ver, la visión del mundo que tiene el robot es mucho más simple de lo que realmente es, y esto puede conducir a que cometa errores.

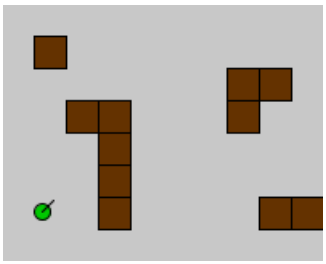

Figura 10.3: Representación del mundo implementado.

Como habíamos mencionado antes, se necesitará una función que permita obtener la celda en donde se encuentra el robot dadas las coordenadas de éste, pero también se requerirá otra función que nos diga en que dirección discretizada se encuentra viendo el robot. La función anterior deberá calcular el ángulo al que se encuentra viendo el robot, pero centrado sobre las posibles direcciones de movimiento. Por ejemplo: si el robot se encuentra con una dirección de 12°, la función deberá regresar el valor 0°, ya que 12° se encuentra entre (24.5°, -24.5°].

Inicialización Deberán posicionar al robot aleatoriamente en el mundo, después deberán calcular la probabilidad de que el robot se encuentre en alguna celda, llamémosle a esta probabilidad *creencia* y representa lo que el robot *cree* acerca de su posición en el mundo. Al inicio esta probabilidad será igual para toda posición porque el robot no sabe dónde está:

$$P(posición) = \frac{1}{\#\ de\ posiciones\ que\ no\ son\ obstáculo}$$

Después, deberán calcular la distancia desde el centro de cada celda al obstáculo más cercano en línea recta, para cada ángulo discretizado. Esta información la deberán guardar para cada posición de tal manera que eviten estarla calculando cada que sea considerada en el algoritmo siguiente.[4]

10.5. Notación

Definamos la posición discretizada del robot usando una variable tridimensional $l =< x, y, \theta >$, donde x y y son las coordenadas del centro de la celda donde se encuentra el robot y θ es la dirección discretizada en la que está viendo el robot. A partir de este momento consideraremos únicamente variables dicretizadas. Sea l_t la posición real del robot en el tiempo t, y L_t la variable aleatoria que modela la probabilidad de que el robot se encuentre en una posición l dada.

[4]Se sugiere guardar esta información como un atributo de la celda

Como el robot no sabe su posición exacta, su *creencia* o $Bel(L_t)$ es una ditribución de probabilidad $P(L)$ sobre el espacio de posibles posiciones. Esta *creencia* nos permite saber cuál es la probabilidad de que se encuentre en una celda l en el tiempo t, formalmente $Bel(L_t = l)$. Sea n el número de posiciones posibles.
También aprovecharé para definir a las lecturas de los sensores como: s_T la lectura del láser, θ_T la lectura del giro y a_T la lectura del sensor odométrico. Sean:

$P(s_T \mid l)$ es la probabilidad de que se haya obtenido una medición s_T si el robot se encuentra en la posición l. Calcularemos esta probabilidad como:

$$P(s_T \mid l) = \frac{1}{(\sqrt{2\pi}\sigma_{láser2})} * \mathrm{e}^{(\frac{-(s_T-\mu_{láser2})^2}{2\sigma^2_{láser2}})} \tag{10.1}$$

donde $\sigma_{láser2}$ modela, en parte, el error del sensor y el error por la discretización del mundo, y $\mu_{láser2}$ es la distancia desde el centroide de la celda de la posición l hacia el primer obstáculo.

$P(\theta \mid \theta', \theta_T)$ es la probabilidad de que el robot esté mirando en el ángulo θ de la posición l dado que se encontraba mirando en el ángulo θ' de la posición l' y se giró un ángulo θ_T. Supondremos que al girar, el robot no cambia de celda. Calcularemos esta probabilidad como:

$$P(\theta \mid \theta', \theta_T) = \frac{1}{(\sqrt{2\pi}\sigma_\theta)} * \mathrm{e}^{(\frac{[\theta_T-(\theta-\theta')]^2}{(\sigma_\theta)^2})} \tag{10.2}$$

donde σ_θ modela, en parte, el error del sensor y el error por la discretización del mundo. Esta variable será definida por ustedes.

$P(l \mid l', a_T)$ es la probabilidad de que el robot esté en la posición l dado que se encontraba en la posición l' y se avanzó a_T cms. Calcularemos esta probabilidad como:

$$P(l \mid l', a_T) = \frac{1}{(\sqrt{2\pi}\sigma)} * \mathrm{e}^{-\frac{1}{2}\frac{[(x'+a_T\cos\theta'-x)^2+(y'+a_T\,\mathrm{sen}\,\theta'-y)^2+(\theta'-\theta)^2]}{\sigma^2}} \tag{10.3}$$

con σ modelando el error del sensor.[5]

Una vez definida la notación, procederemos a ver el algoritmo completo de esta técnica de localización.

10.6. Algoritmo

El algoritmo es el siguiente:

for all *posicion l* en *mundo* **do** ▷ Inicialización
  $Bel(L_0 = l) \leftarrow P(L_0 = l) = \frac{1}{n}$
end for
for all *posicion l* en *mundo* **do**
  determinar las distancias hacia obstaculos

[5] Recuerden utilizar radianes, ya que las funciones de `Java` están definidas en radianes.

end for
while *true* **do** ▷ Se recibió comando o medición del láser
  if no se movio el robot **then** ▷ Se recibió medición del láser
    $\alpha_T \leftarrow 0$ ▷ Constante de normalización
    for all *posicion l* en *mundo* **do**
      $Bel(L_T = l) \leftarrow P(s_T \mid l) * Bel(L_{T-1} = l)$
      $\alpha_T \leftarrow \alpha_T + Bel(L_T = l)$
    end for
    for all *posicion l* en *mundo* **do** ▷ Ahora se normaliza la creencia
      $Bel(L_T = l) \leftarrow {\alpha_T}^{-1} * Bel(L_T = l)$
    end for
  end if
  if se movio el robot **then**
    if se obtuvo θ_T **then** ▷ Se obtuvo una lectura del sensor de giro
      for all *posicion l* en *mundo* **do**
        $Bel(L_T = l) \leftarrow [\sum_{\theta'} P(\theta \mid \theta', \theta_T) * Bel(L_{T-1} = l)]$
      end for
    end if
    if se obtuvo a_T **then** ▷ Se obtuvo una lectura del sensor odométrico
      for all *posicion l* en *mundo* **do**
        $Bel(L_T = l) \leftarrow [\sum_{l'} P(l \mid l', a_T) * Bel(L_{T-1} = l')]$
      end for
    end if
  end if
end while

10.7. Desarrollo e implementación

Esto se implementará en processing, recuerden que debe estar bien hecho y comentado, ya que este proyecto vale el 20 % de su calificación.
Su aplicación deberá mostrar el mundo, los obstáculos y la probabilidad de cada celda para cada ángulo usando distintos colores o gradientes de un mismo color.