Universidad de San Andrés, Depto. de Ingeniería

Práctica 2 - Modelos Probabilísticos, y Filtros Discretos

I-402 - Principios de la Robótica Autónoma

Prof. Dr. Ignacio Mas, Tadeo Casiraghi, Bautista Chasco

19 de marzo de 2025

Fecha límite de entrega: 04/04/25, 23:59hs.
Modo de entrega: Enviar por el Aula Virtual del Campus todo el código comentado y los gráficos (.jpg ó .pdf), todo en una sola carpeta comprimida.

1. Muestreo de distribuciones de probabilidad

Implementar tres funciones en Python que generen muestras de una distribución normal $\mathcal{N}(\mu, \sigma^2)$. Los parámetros de entrada de dichas funciones son la media μ y la varianza σ^2 de dicha distribución. Como única fuente de aleatoriedad, utilizar muestras de una distribución uniforme.

1. En la primer función, generar las muestras de la distribución normal sumando muestras de 12 distribuciones uniformes.

2. En la segunda función, usar "muestreo con rechazo"

3. En la tercera función, usar el método de la transformación de Box-Muller. El método de Box-Muller permite generar muestras de una distribución normal usando dos muestras uniformemente distribuidas $u_1, u_2 \in [0, 1]$ según la ecuación:

$$x = cos(2\pi u_1)\sqrt{-2\log u_2} \tag{1.1}$$

Comparar los tiempos de ejecución de las tres funciones usando la función de Python `timeit`.
Además, comparar el tiempo de ejecución de las tres funciones con el de la función `numpy.random.normal` de Python.

2. Modelo de movimiento basado en odometría

Implementar un modelo de movimiento basado en odometría siguiendo estos pasos:

1. Crear una función en Python que implemente un modelo de movimiento basado en odometría. los tres argumentos de entrada deberán ser:

$$\mathbf{x_t} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \qquad \mathbf{u_t} = \begin{pmatrix} \delta_{r1} \\ \delta_{r2} \\ \delta_t \end{pmatrix} \qquad \alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}, \tag{2.1}$$

   donde $\mathbf{x}_t$ es la pose actual del robot, $\mathbf{u}_t$ es la lectura de odometría obtenida por los sensores del robot y α son los parámetros de ruido del modelo de movimiento (varianza de la distribución). La función deberá devolver la nueva pose del robot $\mathbf{x}_{t+1}$.

   La implementación de la función deberá incluir el error de obtención de la odometría. Usar los métodos de muestreo del ejercicio 1 para generar muestras aleatorias normalmente distribuidas para introducir ruido en el modelo de movimiento.

2. Evaluar el modelo de movimiento desarrollado generando 5000 muestras con los siguientes argumentos:

$$\mathbf{x_t} = \begin{pmatrix} 2{,}0 \\ 4{,}0 \\ 0{,}0 \end{pmatrix} \qquad \mathbf{u_t} = \begin{pmatrix} \frac{\pi}{2} \\ 0{,}0 \\ 1{,}0 \end{pmatrix} \qquad \alpha = \begin{pmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}01 \\ 0{,}01 \end{pmatrix}, \tag{2.2}$$

   Graficar las posiciones $(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})$ para las 5000 muestras en un solo gráfico.

3. Filtro Discreto

Queremos estimar la posición de un robot que se mueve es un mundo unidimensional acotado. El mundo tiene 20 celdas y el robot se encuentra inicialmente en la celda 10. El robot no puede salir del area especificada.
En cada paso, el robot puede ejecutar el comando 'avanzar un paso' o 'retroceder un paso'. Este movimiento puede tener cierto error, por lo que el resultado puede no siempre

ser el esperado. De hecho, el robot se comporta de la siguiente manera: al ejecutar 'avanzar un paso' el resultado será el siguiente:

- el 25 % de las veces, el robot no se moverá
- el 50 % de las veces, el robot avanzará una celda
- el 25 % de las veces, el robot se moverá dos celdas
- el robot en ningún caso se moverá en el sentido opuesto o avanzará más de dos celdas.

El modelo de 'retroceder un paso' es similar, pero en el sentido opuesto. Ya que el mundo es acotado, el comportamiento al intentar 'avanzar un paso' en los bordes es el siguiente:

- si el robot está en la última celda, al tratar de avanzar se quedará en la misma celda el 100 % de las veces
- si el robot está en la penúltima celda, al tratar de avanzar se quedará en la misma celda el 25 % de las veces, y se moverá a la próxima celda el 75 % de las veces.

Lo mismo sucederá en sentido contrario en el otro extremo del mundo al intentar retroceder.
Implementar un filtro de Bayes discreto en Python y estimar la posición final del robot (belief) después de ejecutar 9 comandos consecutivos de 'avanzar un paso' y luego 3 comandos de 'retroceder un paso'. Graficar el belief resultante de la posición del robot. Comenzar con el belief inicial de

```
bel = numpy.hstack((numpy.zeros(10),1,numpy.zeros(9))).
```