Universidad de Buenos Aires, Facultad de Ingeniería

Práctica 3 -Localización EKF y Mapeo

Robótica Móvil - Un enfoque probabilístico

Prof. Dr. Ignacio Mas

12 de octubre de 2017

Fecha límite de entrega: 23/10/17, 10hs.
Modo de entrega: Enviar por correo electrónico a `imas@fi.uba.ar` el código (.m) comentado y los gráficos (.jpg ó .pdf) y/o animaciones.

1. Filtro de Kalman Extendido

En este ejercicio se implementará un filtro de Kalman Extendido (EKF) basándose en una estructura de código provista por la cátedra.

1.1. Notas preliminares

La estructura provista contiene los distintos componentes del filtro EKF, para que el esfuerzo del desarrollo sea en los detales de la implementación del filtro propiamente dicho. El archivo comprimido que se provee incluye las carpetas:

- **data** contiene la descripción del mundo y las lecturas del sensor
- **matlab** contiene la estructura del filtro EKF con secciones para ser completadas
- **plots** guarda los gráficos generados como resultado

Para ejecutar el filtro, desde la carpeta *matlab* correr el archivo `extended_kalman_filter.m`. Mientras que corre el filtro, los gráficos se van guardando en la carpeta *plots*. El algoritmo inicialmente está incompleto y no correrá correctamente. La estructura hace uso de la librería *librobotics*, escrita por Kai Arras (ASL-EPFL) para la visualización. Todas las funciones están definidas en la estructura de código.
Algunos consejos adicionales:

- Desactivar la visualización comentando la línea `plot state(...)` en el script `extended_kalman_filter.m` para acelerar la ejecución.
- Para probar el filtro sólo con algunos pasos, cambiar el tiempo final en el bucle *for* principal con, por ejemplo, `for t = 1:50`.

1.2. Paso de predicción EKF

Supongamos que tenemos un robot diferencial operando en el plano, por lo que su estado esta definido por $\langle x, y, \theta \rangle$. Su modelo de movimiento esta definido como el Modelo de Odometría visto en clase en el capítulo de *modelos de movimiento*.

1. Calcular la matriz jacobiana G_t de la función de movimiento g sin ruido.
2. Implementar el paso de predicción el filtro EKF en el archivo `prediction_step.m` usando el jacobiano G_t calculado en el punto anterior. Asumir que el ruido del modelo de movimiento esta definido por:

$$Q_t = \begin{pmatrix} 0{,}2 & 0 & 0 \\ 0 & 0{,}2 & 0 \\ 0 & 0 & 0{,}02 \end{pmatrix}.$$

1.3. Paso de corrección EKF

1. Calcular la matriz jacobiana H_t de la función de medición h sin ruido de un sensor que sólo mide distancia (range).
2. Implementar el paso de corrección del filtro EKF en el archivo `correction_step.m` usando el jacobiano H_t. Asumir que el ruido de medición está caracterizado por la la matriz diagonal cuadrada R_t:

$$R_t = \begin{pmatrix} 0{,}5 & 0 & 0 & \dots \\ 0 & 0{,}5 & 0 & \dots \\ 0 & 0 & 0{,}5 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{size(z_t) \times size(z_t)}.$$

Luego de implementar los pasos de predicción y corrección, puede generarse una animación con las figuras guardadas en la carpeta *plots*.
Nota: Con el comando *ffmpeg* (si está instalado en tu sistema operativo) se puede generar una animación de los gráficos de la carpeta *plots* de la siguiente manera:
`ffmpeg -r 10 -i kf_%03d.png ekf.mp4`

2. Mapeo con poses conocidas

Un robot debe construir un mapa de grilla de ocupación (celdas $c_0, \ldots, c_n$) de un entorno unidimensional usando una secuencia de mediciones de un sensor de distancia.
Asumir el siguiente modelo de sensor: cada celda de la grilla con una distancia menor que la distancia medida se asume ocupada con una probabilidad de $p = 0{,}3$. Cada celda más allá de la distancia medida se asume ocupada con una probabilidad de $p = 0{,}6$. Las celdas ubicadas a más de $20cm$ por detrás de la distancia medida no cambian su probabilidad de ocupación.

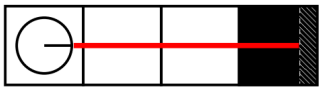

Figura 2.1: representación de grilla unidimensional

Calcular el mapa de grilla de ocupación resultante usando el modelo inverso del sensor con MATLAB/Octave. Usar un vector `m=0.5*ones(1,21)` para inicializar los valores de los *beliefs* (probabilidad de que las celdas estén ocupadas) y un vector `c=[0:10:200]` como coordenadas de las celdas. Visualizar el *belief* resultante con `plot(c,m)`.

Resolución de la grilla	$10cm$
Logitud del mapa (1-D)	$2m$
posición del robot	c_0
orientación del sensor	mirando hacia c_n (ver figura)
mediciones (en cm)	101, 82, 91, 112, 99, 151, 96, 85, 99, 105
prob. *a priori*	0.5