ISSN 1870-4069

Localización de un robot móvil mediante el filtro de Kalman extendido y el simulador Gazebo

Rolan Bacilio Anota, Eduardo Sánchez-Watanabe,
Alberto Petrilli-Barceló, Fermín Ramírez-Leyva

Universidad Tecnológica de la Mixteca,
México

{adairbacilio,e.akio.watanabe}@gmail.com,
{petrilli,hugo}@mixteco.utm.mx

Resumen. La localización de un robot móvil permite realizar tareas de planeación de trayectorias. Aunque existen modelos dinámicos para distintos tipos de robots móviles, los efectos dinámicos no lineales en el ambiente, modelo, sensores y mecánica impiden un conocimiento exacto de la posición del robot. Por lo anterior es necesario estimar la posición considerando efectos probabilísticos. El Filtro de Kalman Extendido (EKF) permite la estimación de la pose de un robot móvil considerando desviaciones respecto al modelo dinámico no lineal. Se presenta la implementación del EKF para la localización de un robot móvil en la plataforma de simulación Gazebo. La posición del robot es obtenida mediante segmentación por color, la imagen se adquiere a través de una cámara colocada en el entorno de simulación. Se compara la estimación de la posición con respecto a la posición adquirida mediante segmentación por color.

Palabras clave: Filtro de Kalman extendido, localización, ROS, Gazebo.

Mobile Robot Localization Using the Extended Kalman Filter and Gazebo Simulator

Abstract. Locating a mobile robot allows performing path planning tasks. Although there are dynamic models for different types of mobile robots, the non-linear dynamic effects on the environment, model, sensors, and mechanics prevent an exact knowledge of the robot's position. Therefore, it is necessary to estimate the position considering probabilistic effects. The Extended Kalman Filter (EKF) allows the estimation of the pose of a mobile robot considering deviations from the non-linear dynamic model. The implementation of the EKF for the location of a mobile robot on the Gazebo simulation platform is presented. The position of the robot is obtained through color segmentation, the image is acquired through a camera placed in the simulation environment. The position estimate is compared to the position acquired by color segmentation.

Keywords: Extended Kalman filter, localization, ROS, Gazebo.

1. Introducción

La evolución actual de la robótica se enfoca a la aplicación en ambientes no estructurados. Este tipo de ambientes se caracteriza por la presencia de incertidumbre que puede provenir de los siguientes factores: ambiente, sensores, robots, modelos y cómputo [10]. El manejo de la incertidumbre en robótica se realiza a través de la consideración de procesos estocásticos, utilizando los principios de la robótica probabilística.

La robótica probabilística se enfoca en la cuantificación de una distribución de probabilidad condicional conocida como creencia $Bel(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{1:k}, \mathbf{u}_{1:k})$, que mide la probabilidad de obtener el estado $\mathbf{x_k}$ en el tiempo k cuando se tiene el conocimiento de las variables medidas ($\mathbf{z}_{1:k}$) y las entradas aplicadas ($\mathbf{u}_{1:k}$), la notacion $1 : k$ toma encuenta todas las variables desde la muestra 1 hasta k. La estimación del estado se realiza al obtener la esperanza de $Bel(\mathbf{x}_k)$. La forma general para el cálculo de la creencia se realiza mediante el filtro recursivo de Bayes o filtro de Bayes [2]. El procedimiento para el cálculo de la creencia se basa en dos etapas: predicción y corrección. La etapa de predicción obtiene una estimación del estado a partir del comando de control utilizado y los datos de medición anteriores, la predicción de la estimación se denota como $\overline{Bel}(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k})$. Una vez realizada la predicción de estado, se realiza la corrección a partir de la medida del sensor en el tiempo t y la predicción anteriormente realizada.

La implementación del filtro de Bayes depende de la forma matemática que tengan las distribuciones de probabilidad utilizadas. Si se utiliza la distribución normal en su forma unimodal, entonces se obtiene el filtro de Kalman [10]. El filtro de Kalman toma al modelo en espacio de estados para estimar el estado y la salida, realizando un filtrado óptimo a la señal de salida [9], fue propuesto por Rudolf E. Kalman en 1960 [5] para sistemas discretos. Debido a que el filtro de Kalman tiene la restricción de ser utilizado únicamente para sistemas lineales, se ha propuesto el EKF como una extensión a sistemas no lineales.

Para el problema de localización de un robot móvil, el estado del sistema corresponde a la pose del robot. La entrada depende de la configuración del robot móvil, en este caso se tiene como comandos de entrada a la velocidad lineal y a la velocidad angular del robot. El modelo de movimiento propuesto es el modelo de movimiento de velocidad [10]. Como variable sensada se utiliza una cámara virtual, que mediante segmentación por color estimará la posición del robot. Se elige como simulador a Gazebo [7] debido a su compatibilidad directa con ROS, el cual es un framework popular para controlar robots.

2. Modelo de movimiento de velocidad

El modelo dinámico de un sistema describe el comportamiento de su salida con respecto a un tiempo t utilizando modelos matemáticos descritos como ecuaciones diferenciales.

Los sistemas dinámicos se caracterizan porque las entradas anteriores al tiempo t afectan a su salida, cuando la salida de un sistema sólo depende de

la entrada en el momento t se le conocen como sistemas estáticos. En ocasiones el tiempo t es sustituido por una variable de instante k ($t = k\Delta t$), esto sucede cuando el sistema se actualiza en periodos no continuos de instantes de tiempo Δt, a estos sistemas se les conoce como sistemas discretos. Un sistema dinámico no lineal en tiempo discreto se representa por la siguiente ecuación [10]:

$$\mathbf{x}_k = g_k(\mathbf{x}_{k-1}, \mathbf{u}_k) + \epsilon_k, \tag{1}$$

donde $\mathbf{x}_k$ representa el estado del sistema, $g_k(\mathbf{x}_{k-1}, \mathbf{u}_k)$ es el vector de transición del estado. Para la ecuación de medición del sistema se tiene:

$$z_k = h_k(\mathbf{x}_k) + \delta_k. \tag{2}$$

La variable $h_k(\mathbf{x}_k)$ representa la relación entre el vector de estado y las variables a medir, cuando las variables de medición coinciden con el estado del sistema se tiene $h_k(\mathbf{x}_k) = \mathbf{x}_k$. Los vectores ϵ_k, δ_k representan el ruido del sistema que depende de variables aleatorias. La matriz de covarianza de ϵ_k es R y la matriz de covarianza de δ_k es Q.

El modelo dinámico para el movimiento de un robot móvil asume como entrada de control al vector $\mathbf{u}_k = \begin{bmatrix} v_k & \omega_k \end{bmatrix}^T$ donde v_k representa la velocidad lineal y ω_k la velocidad angular en el instante k, así mismo, $g_k(\mathbf{x}_{k-1}, \mathbf{u}_k)$ es una ecuación de movimiento no lineal dependiente de la entrada $\mathbf{u}_k$ y del estado $\mathbf{x}_{k-1}$ que está conformado por la posición (x, y) y la orientación θ en el instante $k-1$.

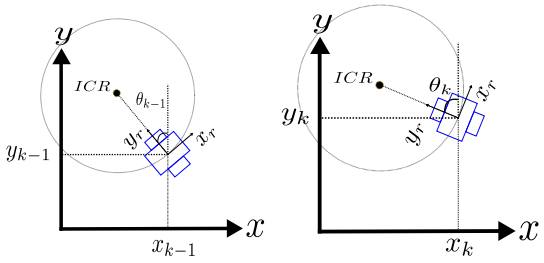
Se considera el movimiento del robot en el plano, por ello se utiliza el centro instantáneo de rotación (ICR) para obtener las ecuaciones de movimiento [8]. Se asume que la entrada $\mathbf{u}_k$ es ejecutada en el estado $\mathbf{x}_{k-1}$. Antes de ejecutar el movimiento (Figura 1(a)) el robot tiene como velocidad lineal y angular $\mathbf{u}_k = \begin{bmatrix} v_k & \omega_k \end{bmatrix}^T$, con los valores de velocidad se puede obtener el radio de curvatura mediante:

$$r = \frac{v_k}{\omega_k}. \tag{3}$$

Entonces se obtienen las coordenadas ICR como:

$$x_{icr} = x_{k-1} - \frac{v_k}{\omega_k} \sin(\theta_{k-1}), \tag{4}$$

$$y_{icr} = y_{k-1} + \frac{v_k}{\omega_k} \cos(\theta_{k-1}). \tag{5}$$



(a) Antes de moverse (b) Después de moverse

Fig. 1. Deducción de las ecuaciones de movimiento mediante el centro instantáneo de rotación.

Después de realizar el movimiento con velocidad $\mathbf{u}_k$ y un tiempo constante Δ_t la orientación del robot habrá cambiado a:

$$\theta_k = \theta_{k-1} + \omega_k \Delta_t. \tag{6}$$

Entonces el robot cambiará a su nueva posición espacial como muestra la Figura 1(b). Se puede determinar la nueva posición a partir de la posición del ICR debido a que este se conserva si $\mathbf{u}_k$ es constante. Bajo la consideración anterior se tiene:

$$x_k = x_{icr} + \frac{v_k}{\omega_k} \sin(\theta_k), \tag{7}$$

$$y_k = y_{icr} - \frac{v_k}{\omega_k} \cos(\theta_k). \tag{8}$$

Combinando las ecuaciónes (4) y (5) con (7),(8) y (6), se obtienen las ecuaciones de movimiento para un robot móvil con entradas (v_k, ω_k) mediante la ecuación (9) [10]:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \frac{v_k}{w_k} \left(\sin(\theta_{k-1} + \omega_k \Delta_t) - \sin(\theta_{k-1})\right) \\ y_{k-1} - \frac{v_k}{w_k} \left(\cos(\theta_{k-1} + \omega_k \Delta_t) - \cos(\theta_{k-1})\right) \\ \theta_{k-1} + \omega_k \Delta_t. \end{bmatrix} \tag{9}$$

La ecuación (9) representa las ecuaciones cinemáticas del robot móvil en tiempo discreto. Las variables sensadas del robot móvil. corresponden directamente a la pose del robot, esto es:

$$\mathbf{z}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}. \tag{10}$$

3. Filtro de kalman extendido

El EKF puede verse como una extensión al filtro de Kalman clásico que abarca sistemas dinámicos de tipo no lineal. La única diferencia entre el Filtro

de Kalman clásico y el filtro de Kalman extendido se basa en la linealización del sistema dinámico utilizando la matriz Jacobiana para obtener la matriz de transición de estado, entonces se puede utilizar el algoritmo del filtro de Kalman [6]. Para ello se calcula el Jacobiano de las funciones g_k y z_k utilizando las ecuaciones (9) y (10) respectivamente:

$$G_k = \frac{\partial g_k(\mathbf{x}_{k-1}, \mathbf{u}_k)}{\partial \mathbf{x}_{k-1}} = \begin{bmatrix} 1 & 0 & \frac{v_k}{w_k}\left(\cos(\theta_{k-1} + \omega_k \Delta_t) - \cos(\theta_{k-1})\right) \\ 0 & 1 & \frac{v_k}{w_k}\left(\sin(\theta_{k-1} + \omega_k \Delta_t) - \sin(\theta_{k-1})\right) \\ 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

$$H_k = \frac{\partial h_k(\mathbf{x}_k)}{\partial \mathbf{x}_k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{12}$$

Por simplicidad, se proponen a las matrices de covarianza como:

$$R = Q = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{13}$$

Con las ecuaciones (9), (10), (11), (12) y (13) se implementa el algoritmo del filtro de Kalman extendido. Para la orientación, se extraen los datos de la unidad de medición inercial (IMU) del robot. Los algoritmos 1 y 2 muestran el procedimiento para el cálculo de la corrección y predicción respectivamente. En la fase de predicción las variables $\hat{x}_k^-$ y P_k^- corresponden a los valores de la media y matriz de covarianza de $Bel(\mathbf{x}_k)$. La etapa de predicción toma a consideración al sistema no lineal y actualiza la covarianza a través de el valor de covarianza del ruido inducido.

Algoritmo 1 Filtro de Kalman Extendido: Fase de Predicción.

Entrada: $\hat{x}_{k-1}$,u_k, P_{k-1}, R
Salida: $\hat{x}_k^-$, P_k^-
1: $\hat{x}_k^- = g(x_{k-1}, u_k)$
2: $P_k^- = G_k P_{k-1} G_k^T + R$

La fase de corrección toma las salidas de la fase de predicción, en conjunto con la medición y covarianza del ruido proveniente del sensor para calcular una ganancia de corrección K_k. Como elementos de salida se tiene a la media $\hat{x}_k^-$ y a la covarianza del sistema P_k.

Algoritmo 2 Filtro de Kalman Extendido: Fase de Corrección.

Entrada: $\hat{x}_k^-$, P_k^-, z_k, Q
Salida: $\hat{x}_k$, P_k
1: $K_k = P_k^- H_k^T (H_k P_k^- H_k^T + Q)^{-1}$
2: $\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$
3: $P_k = (I - K_k H_k) P_k^-$

El Algoritmo 2 muestra el procedimiento de cálculo para la obtención de la corrección, es necesario realizar la operación de matriz inversa. La implementación de los algoritmos se realiza en C++ utilizando la biblioteca de matrices Eigen [4].

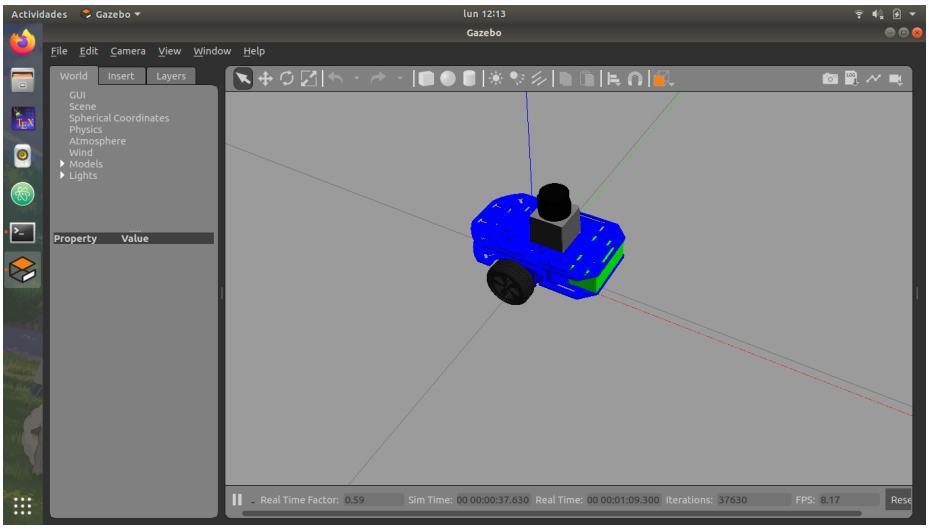


Fig. 2. Simulación de robot móvil.

4. Gazebo

Gazebo es un simulador multirobot para ambientes externos diseñado por Open-Source Robotics Foundation. En [3] se citan las principales características del simulador Gazebo obtenidas de una encuesta a diferentes personas que trabajan en el área de la robótica, de las cuales se citan algunas a continuación:

- Soporta múltiples motores de física. (*Open Dynamics Engine*, Bullet).
- Sistema operativo: 100 % GNU/Linux.
- Interfaz de programación de aplicaciones (API) Principal: 80 % C++.

- Razones de uso: mejor herramienta de evaluación, software listo para usar en el laboratorio y es de código abierto.
- Uso principal: en robótica móvil, en robótica de servicios, en robots humanoides.
- Middleware utilizado: 93 % ROS.

Las razones anteriores influyeron en la decisión de utilizar el simulador Gazebo. Para realizar la simulación se optó por un modelo de robot móvil de tipo diferencial. Este robot se puede controlar mediante los comandos de velocidad lineal y angular, además cuenta con una IMU y con cámaras para poder visualizar [1]. Para identificar la posición del robot se utiliza segmentación por color y se extrae la posición a través del centroide de la imagen. La compatibilidad de Gazebo con ROS permite implementar algoritmos que pueden usarse en un robot con alteraciones mínimas en el código.

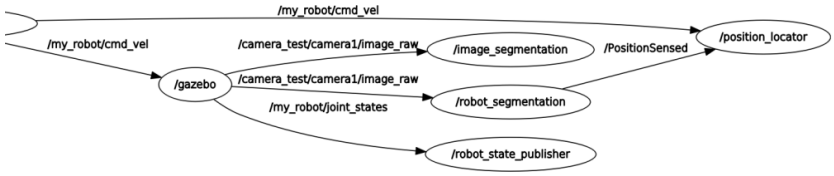


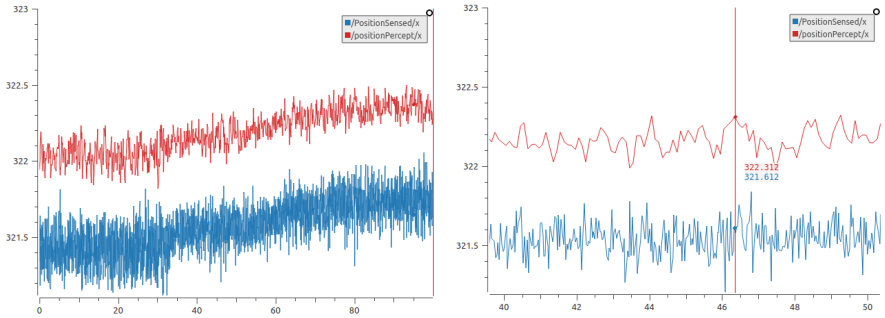
Fig. 3. Nodos y tópicos utilizados en la simulación.

Para realizar el experimento, el robot móvil recibe una velocidad lineal y angular constante, esto para generar una trayectoria de tipo circular. Entonces, mediante el comando *rqt_plot* se grafican los valores obtenidos del simulador y del filtro de Kalman extendido. La Figura 3 muestra los nodos y tópicos implementados para implementar el EKF, el comando de velocidad es enviado a la simulación de Gazebo (mediante el tópico *my_robot/cmd_vel*), donde se utiliza la cámara ubicada arriba del escenario y se publican el tópico de la imagen *camera_test/camera1/image_raw*. El nodo */my_robot/segmentation* obtiene la posición del robot que es enviada a */position_locator* donde se utiliza el algoritmo de EKF.

5. Resultados

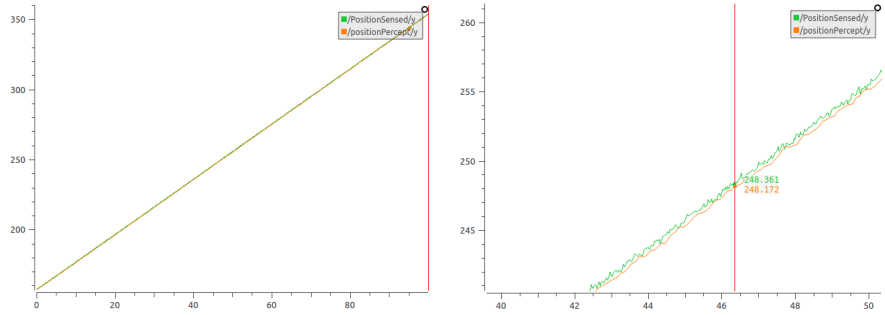
Se realizaron dos pruebas en el simulador Gazebo: movimiento en linea recta, movimiento alrededor de un círculo. Para el movimiento en línea recta, el robot esta orientado al eje y. Las líneas azul y verde corresponden al valor obtenido a partir de la segmentación. La Figura 4(a) muestra los valores adquiridos, mientras que 4(b) es un acercamiento de la imagen, se observa un desfase máximo de un milímetro, y menores oscilaciones de los valores. Se obtiene un resultado similar para la coordenada y (Figura 5), donde se observa un comportamiento

lineal. Los valores introducidos para el movimiento lineal son $\mathbf{u}_k = \begin{bmatrix} 10 & 0 \end{bmatrix}^T$, con v_k en mm/s.



(a) x_k (b) x_k

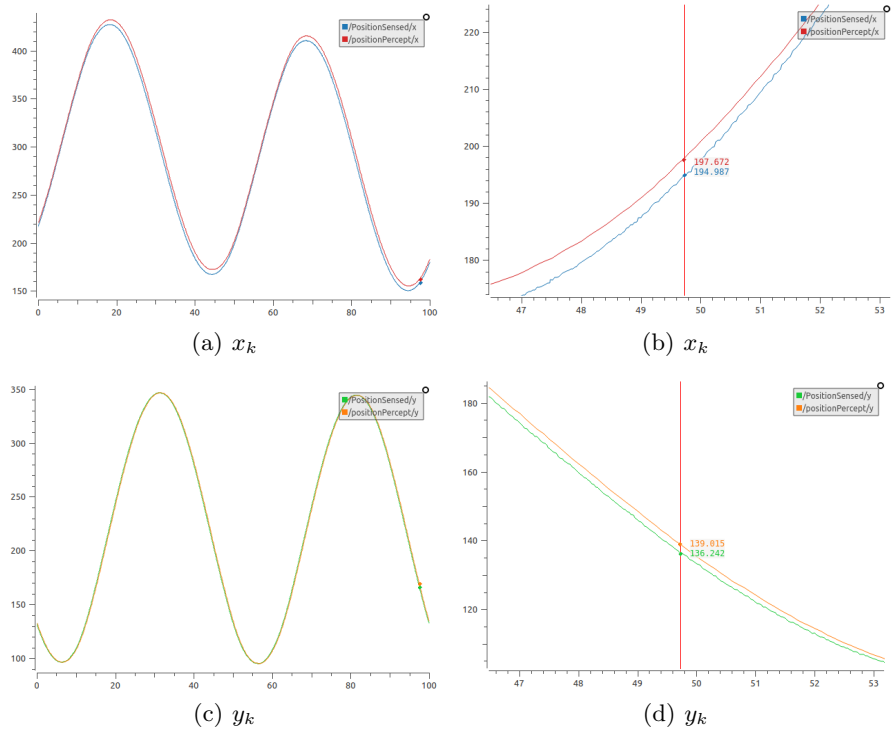
Fig. 4. Resultados del movimiento lineal, posición en el eje x.



(a) y_k (b) y_k

Fig. 5. Resultados del movimiento lineal, posición en el eje y.

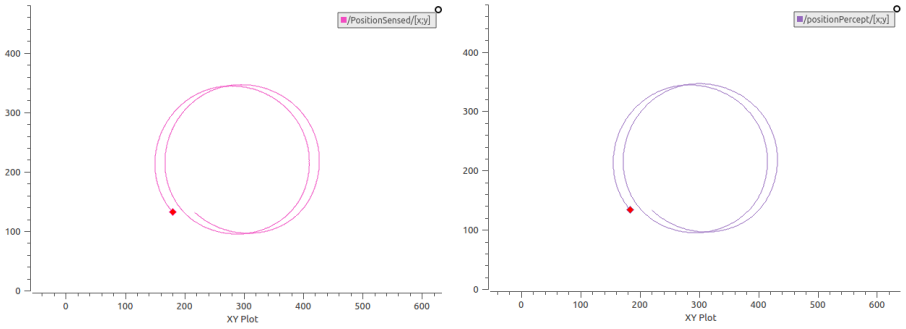
Los resultados del movimiento en una trayectoria circular se muestran en la Figura 6. Las gráficas de color azul y verde corresponden a la medida adquirida mediante los sensores, mientras que las líneas roja y naranja representan el valor al aplicar el filtro. Similar al resultado anterior, se observa una desviación no mayor a 5 milímetros y disminución del ruido al utilizar el filtro. En la Figura 7(a) se puede observar el movimiento realizado en el plano xy mientras que 7(b) se observa el resultado del filtro.



(a) x_k (b) x_k

(c) y_k (d) y_k

Fig. 6. Resultados de la aplicación del filtro de Kalman Extendido, movimiento en forma de circular.



(a) Movimiento sensado (b) Movimiento predecido

Fig. 7. Gráfica XY del movimiento en forma circular.

6. Conclusiones

La implementación del EKF mejora significativamente el sistema de localización del robot, al proveer una medida estable de la posición del robot con la

que se puede en un futuro trabajar en conjunto con algoritmos de navegación más complejos. El filtro de Kalman extendido programado en C++ y utilizando el framework de ROS obtuvo una respuesta en el tiempo de $1ms$, por lo que lo convierte en una opción eficiente para la implementación del filtro de Bayes. Utilizar Gazebo permite la migración del mismo código a una aplicación real.

Referencias

1. Coleman, D.: Gazebo ROS demos (2019), https://github.com/ros-simulation/gazebo_ros_demos
2. Dardari, D., Falletti, E., Luise, M.: Satellite and Terrestrial Radio Positioning Techniques: A Signal Processing Perspective (2012)
3. Ivaldi, S., Padois, V., Nori, F.: Tools for dynamics simulation of robots: a survey based on user feedback pp. 1–15 (2014), http://arxiv.org/abs/1402.7050
4. Jacob, B., Guennebaud, G.: Eigen (aug 2019), http://eigen.tuxfamily.org/index.php?title=Main_Page
5. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82(1), 35–45 (1960)
6. Keatmanee, C., Baber, J., Bakhtyar, M.: Simple Example of Applying Extended Kalman Filter Simple Example of Applying Extended Kalman Filter. 1st International Electrical Engineering Congress (March) (2014)
7. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 3, 2149–2154 (2004)
8. Peacock, T.: Kinematics of Rigid Bodies Instant Centers or Instantaneous Centers. Dynamics and Control pp. 1–6 (2007)
9. Superiores, E., Graduados, P.: Aplicación de los Filtros de Kalman a Sistemas de Control. Ingenieria UC (January 2001), 2403 (2001)
10. Thrun, S.: Probabilistic robotics, vol. 45. Association for Computing Machinery, New Yor, NY, USA, 1 edn. (2002)