DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Robotica Movil

Trabajo práctico Final

Resumen

Segimiento de trayectorias para robot odometrico

Integrante	LU	Correo electrónico
Negri, Franco	893/13	franconegri2004@hotmail.com
Schmit, Matías	714/11	matias.schmit@gmail.com

Palabras claves:

Odometria, Seguimiento Trayectorias, Cimenatica, Kalman

Índice

1. Introduccion

En los últimos años se ha visto un gran avance en el campo de la robótica y en tareas automatizables. En particular, los robots omnidireccionales se han empezado a utilizar tanto en la industria, por su capacidad de maniobrar en espacios reducidos, como en la domótica por su comodidad.

En este trabajo practico implementaremos un sistema que permita la realización de trayectorias con alto grado de seguridad y precisión utilizando un robot omnidireccional. Para ello utilizaremos el entorno de desarrollo ROS y el entorno de simulación V-Rep.

El robot en particular sobre el cual trabajaremos contará con cuatro ruedas fijas de tipo Mecanum de 50mm de radio y cuenta con un sensor láser frontal.

El estudio del movimiento esta dado por el modelado de las fuerzas que afectan el sistema (dinámica) y la matemática presente en la mecánica de movimiento que define las poses del robot en el tiempo (cinemática). En este trabajo nos centramos en lo segundo.

Una vez definido el modelo cinemático se valido su implementación considerando la estimación odometrica del sistema. Se procedió a estudiar el error en la diferencia entre pose obtenida y pose deseada a partir del manejo de la orientación con control a lazo cerrado (feedback). Teniendo este primer modelo de control del error se evaluó la efectividad del seguimiento de una trayectoria sencilla.

Por ultimo se incorporo al sistema un filtro bayesiano, filtro de kalman extendido, para intentar reducir aun mas el error cometido entre pose resultante y esperada. Se muestran experimentos que muestran el impacto del filtro de Kalman sobre la orientación y el seguimiento de la trayectoria previamente usada para el caso anterior.

2. Desarrollo

2.1. Adaptación del modelo cinemático

Como ya anticipamos, para este trabajo practico utilizaremos un robot omidireccional. Este cuenta con cuatro ruedas *Mecanum* dispuestas a los costados del robot, las cuáles cuentan con rodillos especiales que permiten transmitir parte de la fuerza en dirección de un ángulo definido.

El primer desafío que se nos presenta ante este nuevo sistema es poder plantear el modelo cinemático del mismo, recordemos que en el contexto de este trabajo la cinemática es el estudio de cómo se comporta el robot en movimiento y el aporte de cada rueda en este sistema basándose en los inputs de control, el movimiento esperado y sin considerar las fuerzas que se aplican.

Considerando la relación entre inputs y movimiento esperado queda definida la cinemática directa como la obtención de la velocidad lineal y angular (expresada en m/s y radianes respectivamente) del robot a partir de la velocidad de cada actuador.

Por su parte la cinemática inversa determina la velocidad de los actuadores en base a la velocidades deseadas.

Para lograr esto, adaptaremos el modelo cinemático visto durante la materia utilizando el paper provisto por la cátedra, las formulas de cinemática directa pasarán a ser.

$$v_x(t) = (w_1 + w_2 + w_3 + w_4).r/4$$
$$v_y(t) = (-w_1 + w_2 + w_3 - w_4).r/4$$
$$w_z(t) = (-w_1 + w_2 - w_3 + w_4).r/(4(l_x + l_y))$$

Y las de cinemática inversa:

$$w_1 = 1/r(v_x - v_y - (l_x + l_y)w_z$$
$$w_2 = 1/r(v_x + v_y + (l_x + l_y)w_z$$
$$w_3 = 1/r(v_x + v_y - (l_x + l_y)w_z$$
$$w_4 = 1/r(v_x - v_y + (l_x + l_y)w_z$$

Donde el Vector V = (v_x, v_y y w_z) refiere a las velocidades lineares y angular del robot, r el radio de las ruedas (todas las ruedas tienen exactamente el mismo radio), l_x la mitad de la distancia entre las dos ruedas delanteras y l_y la mitad de la distancia entre una rueda delantera y una rueda trasera.En particular para nuestro robot sabemos que $r = 50$ mm, $l_x = l_y = 175$ mm. Por último w_1, w_2, w_3, w_4 representan las velocidades angulares ejercidas por los actuadores de las ruedas.

Finalmente con las velocidades obtenidas en v_x, v_y y w_z podremos estimar la pose del robot en cada instante calculando el desplazamiento en el tiempo que las velocidades nos proveen.

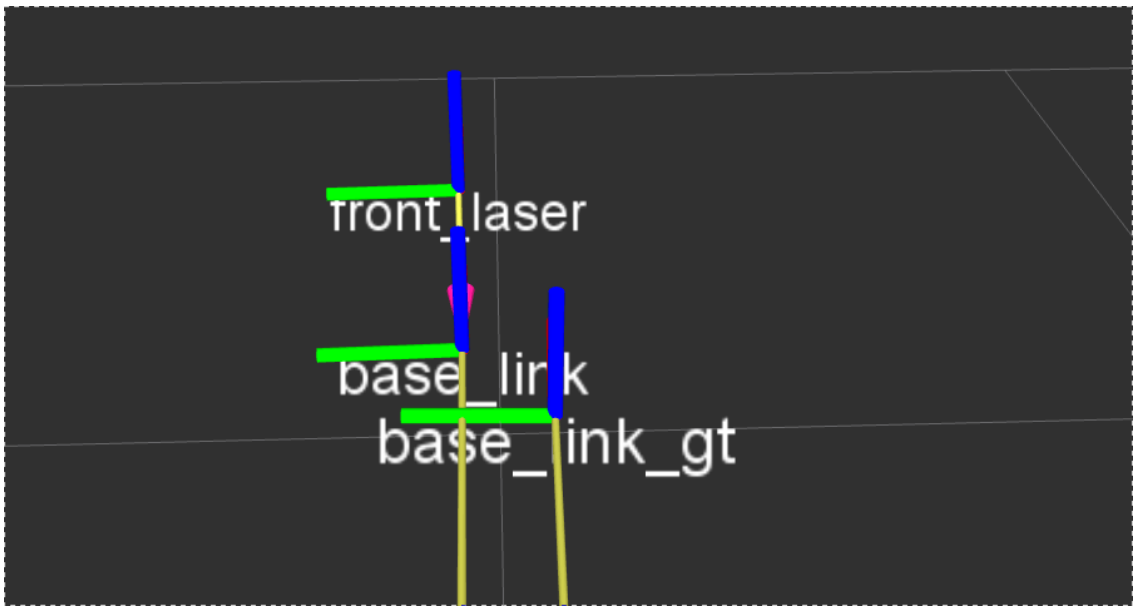
Si bien esto nos permitirá estimar la pose del robot, a causa de los errores sistemáticos del modelo utilizado, el error entre la pose real y la pose estimada divergirá con el tiempo.

En el siguiente apartado procederemos a analizar cuan bueno resulta nuestro modelo cinemático y los cálculos odométricos.

2.1.1. Experimentación

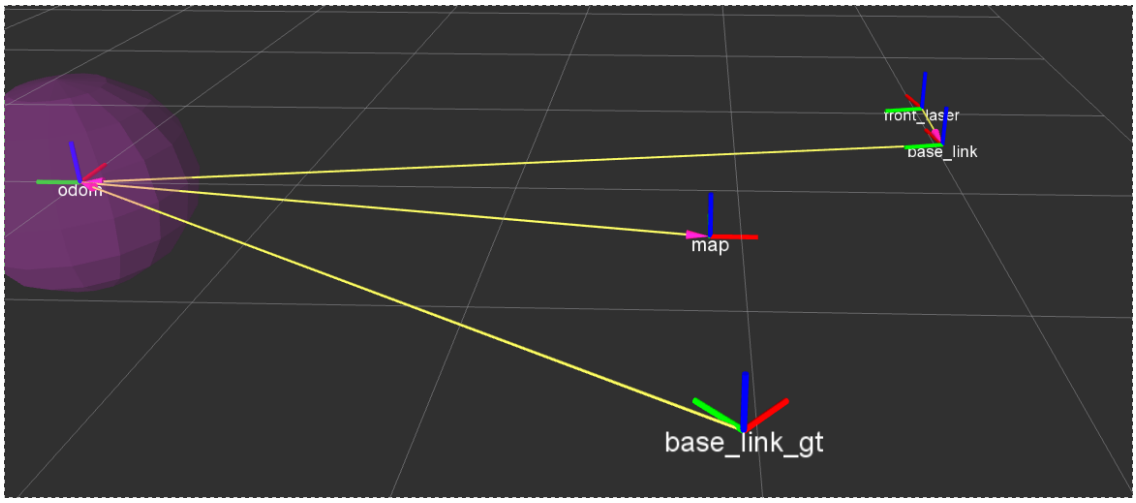
Como ya adelantamos, en este apartado buscaremos ver que tan exacto resulta nuestro modelo cinemático y la odometría calculada de manera experimental. Para ello, enviaremos mensajes a robotcmd_vel a través de rostopic con distintas consignas de velocidad y veremos por RViz cual es la posición calculada por odometría (base_link) y cual es la posición real del robot (base_link_gt).

Como primer experimento enviamos el comando de velocidad ($v_x = 2, v_y = 0, w_z = 0$) durante 8 segundos:



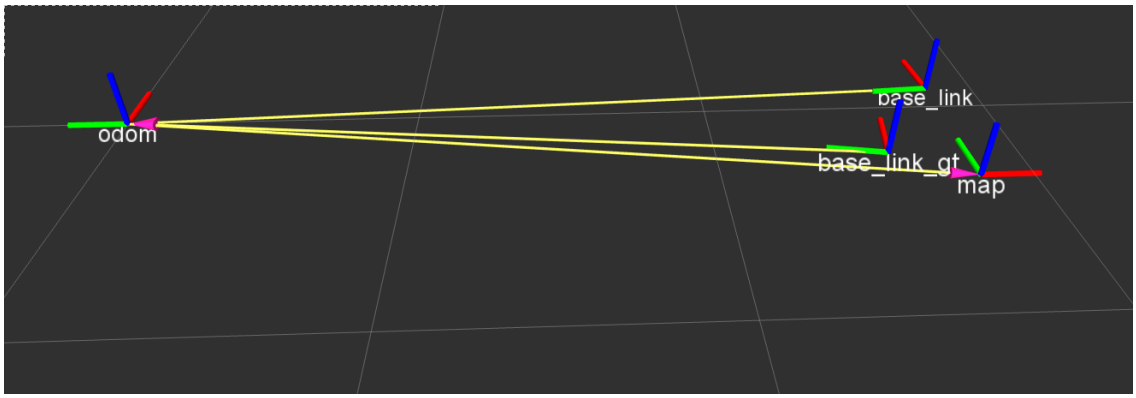
Podemos observar que incluso a altas velocidades el modelo es capaz de predecir con bastante exactitud su posición.

Para el siguiente experimento, enviamos la consigna de velocidad $(v_x = 0, v_y = 2, w_z = 0)$ durante 8 segundos.



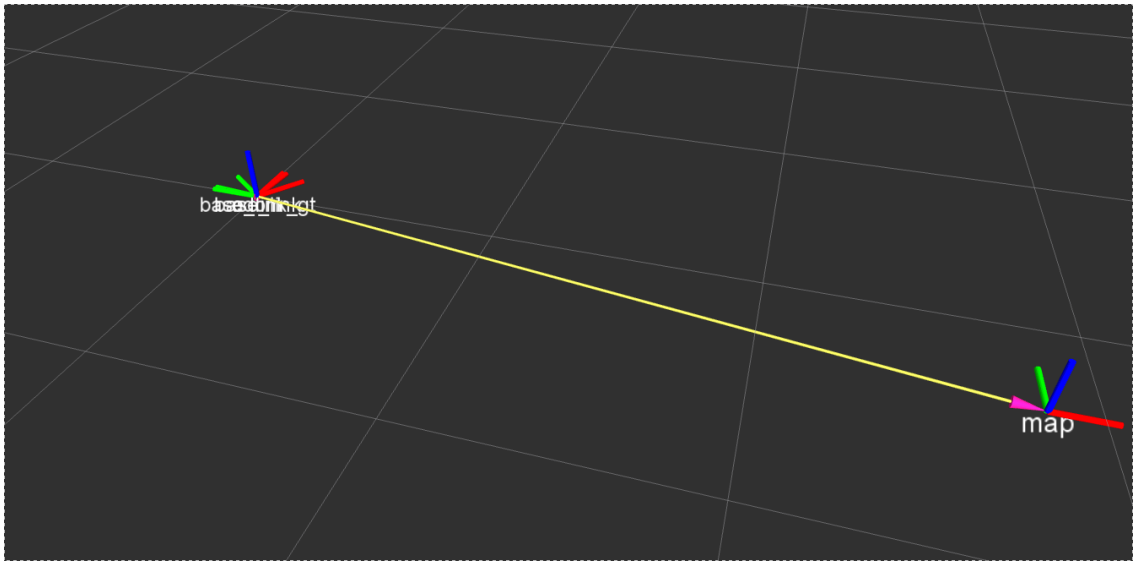
En este caso, podemos ver que la estimación odometrica diverge ampliamente con la provista por el simulador, incluso en vez de realizar una trayectoria recta hacia uno de los costados realiza una elipse. Esto puede deberse principalmente a que los rodillos especiales que le permiten trasladarse en el eje y resbalan o que los actuadores no logran cumplir la consigna deseada. Para el siguiente experimento volvemos a realizar el mismo experimento pero reduciendo la velocidad deseada.

Enviando la consigna $(v_x = 0, v_y = 0{,}5, w_z = 0)$ durante 8 segundos:



Podemos ver que para velocidades menores la calidad de las predicciones odométricas son mucho mejores, aún así puede verse que en este eje el error crece mucho mas rápido que en el eje x.

Para finalizar enviamos como consigna ($v_x = 0, v_y = 0, w_z = 2$), luego de realizar 3 vueltas sobre su eje, detenemos al robot y visualizamos en RViz:



Puede verse que el error en este caso tampoco crece de manera desproporcionada incluso exigiéndole altas velocidades.

2.2. Adaptación del control a lazo cerrado

Así como encontramos errores sistemáticos intrínsecos al diseño del robot, vale la pena destacar que por fuera del entorno simulado en el que estamos probando el modelo, también nos encontraríamos con errores no sistemáticos del entorno que harían aun menos confiable a nuestra estimación odometrica. Sumando a consideración la experimentación y esto último es necesario seguir agregando funcionalidades a nuestro modelo que permita mejorar la precisión en el desplazamiento.

Al igual que para el robot con modelo diferencial queremos que este se pueda trasladar de una pose actual (x_i, y_i, w_i) en un tiempo t_i a una pose objetivo (x_f, y_f, w_f) en un tiempo diferente t_f. Para eso utilizaremos un control a lazo cerrado basado en el controlador Proporcional Integrativo Derivativo (PID) que nos permitirá observar si el robot se encuentra en la pose deseada en cada instante, retroalimentando el error a los controladores de velocidad.

En este caso, al tener un modelo cinemático holonómico las cuentas se simplifican con respecto al modelo diferencial, el control independiente sobre cada velocidad permite plantear cada dimensión por separado:

$$\Delta_x = x_f - x_i/(t_f - t_i)$$
$$\Delta_y = y_f - y_i/(t_f - t_i)$$
$$\Delta_w = w_f - w_i/(t_f - t_i)$$

Donde Δ_x, Δ_y y Δ_w es el .$^{\text{er}}$ror.$^{\text{en}}$tre la pose actual y la objetivo. Minimizar este error significará haber alcanzado la pose objetivo, por lo que definimos las velocidades del robot como:

$$V_x = K\Delta_x$$
$$V_y = K\Delta_y$$
$$\theta = K\Delta_w$$

Donde K es una constante proporcional.

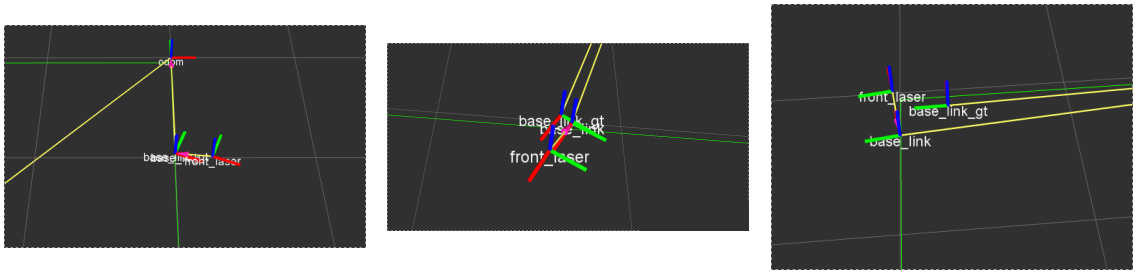
Nos queda solucionar un caso borde que se da si la pose actual y la objetivo se encuentran a una distancia infinita (o cercana a infinita), en este caso será necesaria una velocidad infinita para alcanzarla. Dado que esto no es físicamente posible, definimos una cota máxima de velocidad.

2.2.1. Experimentación

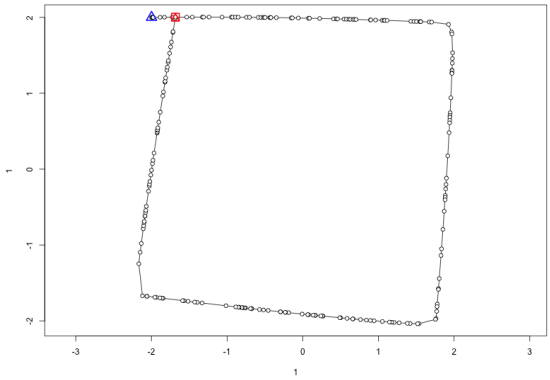
A continuación se pone a prueba el control a lazo cerrado:

Para ello pondremos al robot a realizar el seguimiento de una trayectoria cuadrada de 2 metros de lado y veremos cual es el comportamiento del robot tomando $K = 0{,}4, 1$ y 4 y visualizandolo en r-viz.

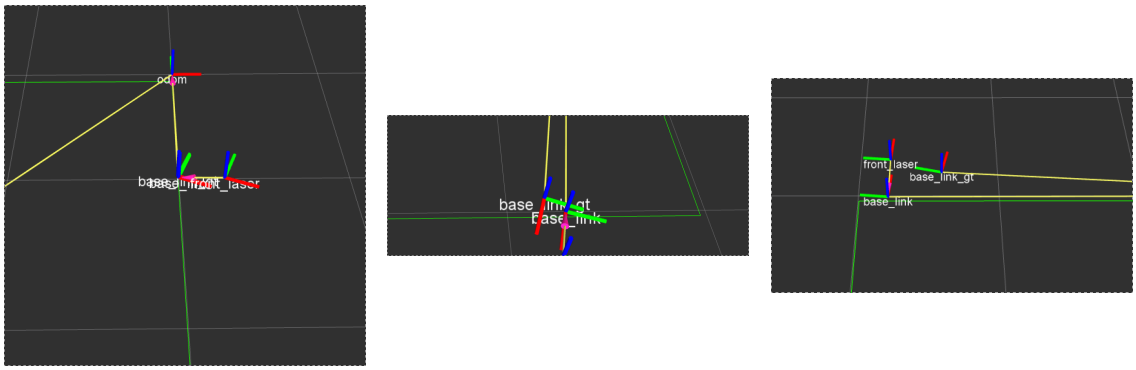
$K = 0{,}4$

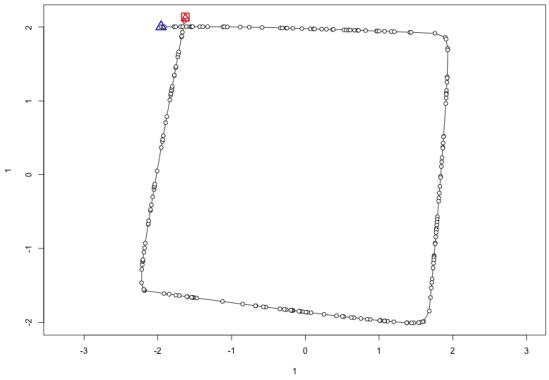


Como otra metodología de visualización, logeamos las poses provenientes del graund truth provistas por roscore y los graficamos de la siguiente manera:

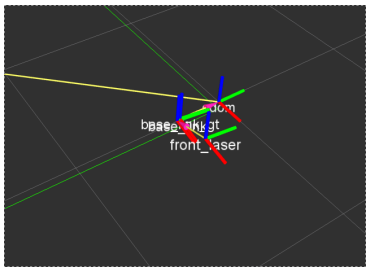


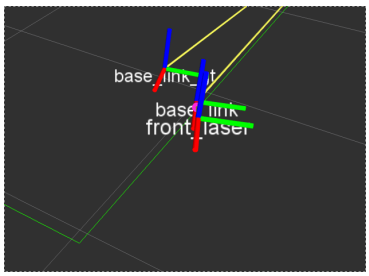
$K = 1$

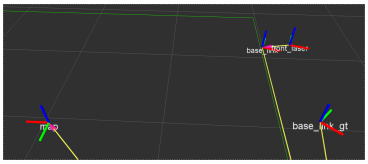


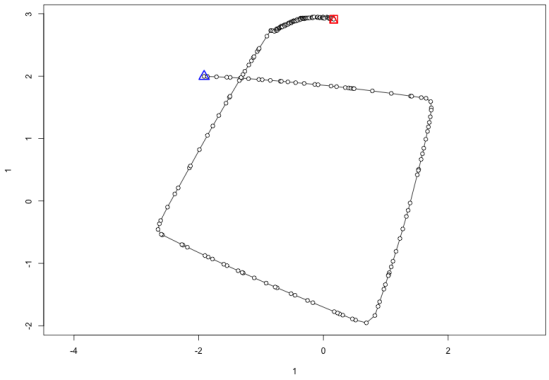


$K = 4$









En estas imágenes podemos notar que si bien el robot sigue la trayectoria respecto a su posición estimada de manera precisa, la estimación de la posición pierde precisión a medida que aumentamos el valor de K y observamos que para cualquier valor mayor a 1 diverge de la trayectoria esperada muy rápido. Consideramos que esto se debe a que a grandes velocidades, el modelo cinemático acumula error más rápidamente y por eso cada posición va empeorando respecto a la esperada.

2.3. Modelado de EKF

Como solución al problema presentado en el apartado anterior, intentaremos mejorar la corrección de la pose del robot agregando un fase de predicción del estado en el cual se encuentra.

Para eso utilizaremos un filtro bayesiano implementando el Filtro de Kalman Extendido similar al visto en la materia para el modelo diferencial.

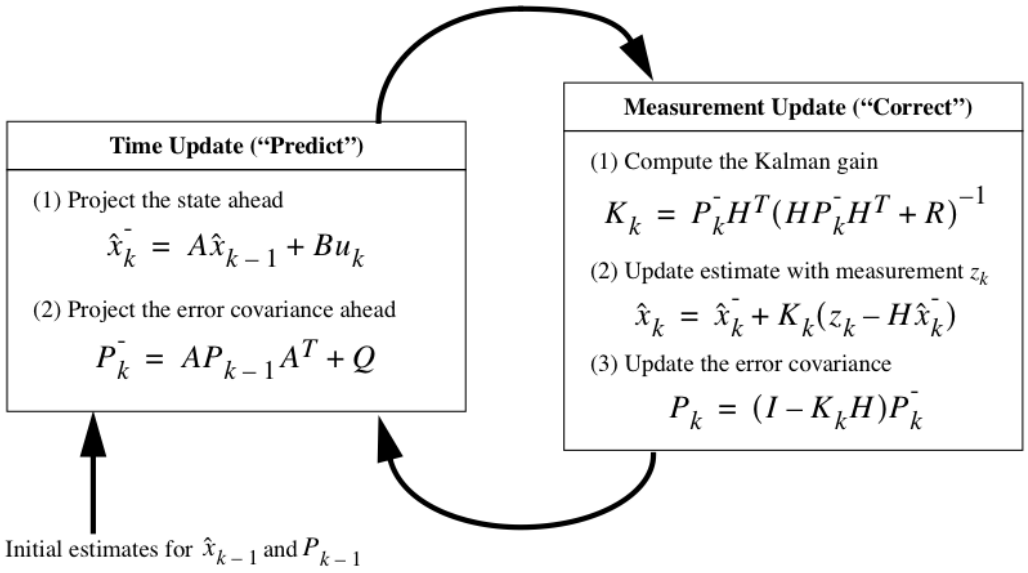
Lo que está ocurriendo, en resumen, es que el robot usará información del censado de un serie de postes ubicados en el entorno para ir actualizando el nivel de certidumbre respecto a la ubicación en la que se encuentra. Partiendo de un *prior* inicial el modelo queda definido por un fase de predicción (*time update*) donde se obtienen estimaciones del estado y la covarianza del error y otra fase de corrección (*measurement update*) donde se actualizan los valores de la fase anterior con las mediciones obtenidas, el *posterior* obtenido pasa a ser nuestro nuevo *prior* y el algoritmo continua iterando de forma análoga.

Partiendo de las funciones que definen ubicación y censado tenemos:

$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) + w_k$ (Proyección de la pose actual en función de una f de transición sobre la iteración anterior y la consideración del ruido w)

$z_k = h(x_{k-1}, v_{k-1}) + v_k$ (Estado del censado a partir de una función h sobre la iteración anterior considerando ruido del sensor v)

El siguiente diagrama expresa el funcionamiento del EKF:



Las variables presentes en el modelo son:

Vector x (estado en el que se encuentra el sistema) Matriz P (Error de la covarianza) Matriz Q (covarianza del ruido en la transición (process)) Matriz R (covarianza del ruido en el censado (measurement)) Matriz W (Jacobiano de las derivadas parciales de f respecto a w (error en el calculo del estado)) Matriz A (Jacobiano de las derivadas parciales de f respecto a x (estado)) Matriz V (Jacobiano de las derivadas parciales de h respecto a v (error de medición)) Matriz H (Jacobiano de las derivadas parciales de h respecto a x (estado))

Por ultimo K se denomina *ganancia de Kalman* y representa el nivel de ponderación que se le da a la información del censado por sobre la predicción estimada.

En este caso de lo que teníamos en el robot diferencial en el modelo de predicción ahora contamos con más grados de libertad posibles en la translación. Ahora es posible que el robot se mueva tanto en x como en y y θ.

Por este motivo, el vector $\overrightarrow{u}$ que representa las entradas de control cambia de la siguiente manera:

$$\overrightarrow{u} = \begin{bmatrix} Vx \\ Vy \\ w \end{bmatrix}$$

El modelo de estado $\overrightarrow{x}$ se mantiene, ya que el robot permanece igual a la del modelo anterior:

$$\overrightarrow{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Dada la modificación en $\overrightarrow{x}$, ahora el modelo de movimiento $f(\overrightarrow{x}, \overrightarrow{u}, \overrightarrow{w})$ ahora pasará a ser:

$$f(\overrightarrow{x}, \overrightarrow{u}, \overrightarrow{w}) = \begin{bmatrix} x + Vx\Delta t cos(\theta) + Vy\Delta t sen(\theta) + w_1 \\ y + Vx\Delta t sen(\theta) + Vy\Delta t cos(\theta) + w_2 \\ norm_{[-\pi,\pi]}(\theta + w\Delta t) + w_3 \end{bmatrix}$$

Calculando los jacobianos de la función f respecto a $\overrightarrow{x}$ y a $\overrightarrow{w}$ respectivamente, tenemos:

$$A = \begin{bmatrix} 1 & 0 & -sen(\theta)\Delta x + cos(\theta)\Delta t Vy \\ 1 & 0 & -cos(\theta)\Delta x - sen(\theta)\Delta t Vy \\ 0 & 0 & 1 \end{bmatrix}$$

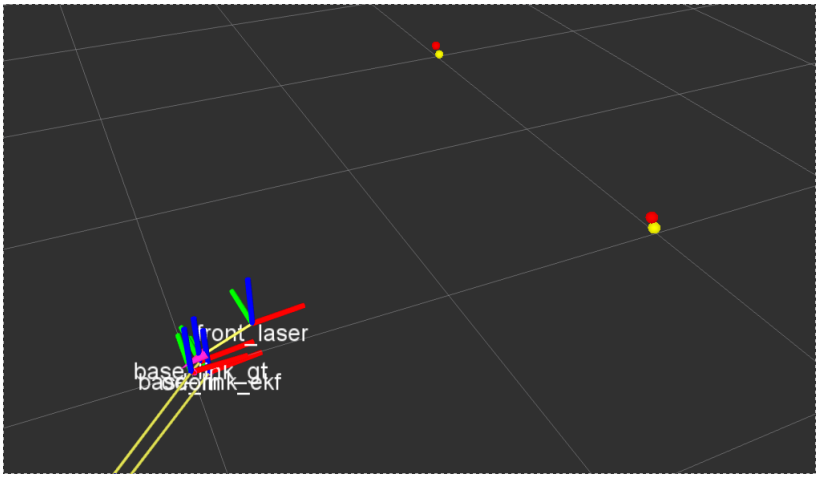
$$W$$

El modelo de censado $h(\overrightarrow{x}, \overrightarrow{v})$ y las mediciones z al no depender del tipo de movimiento que el robot omnidireccional ejerce no cambian, así que permanecen con los valores definidos por defecto iguales a los del modelo diferencial. De la misma forma la matriz Jacobiana H tampoco presenta cambios.

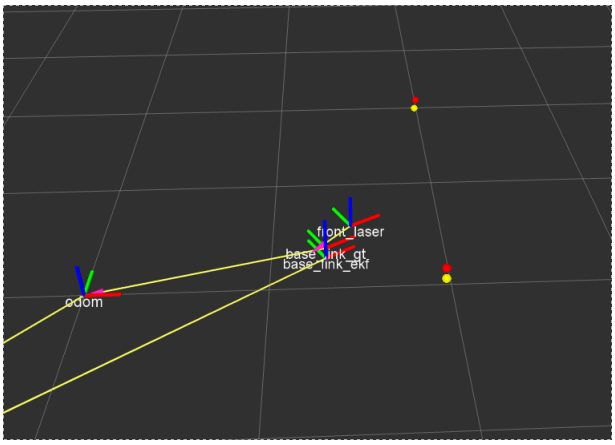
2.3.1. Experimentación

En esta sección buscamos experimentar que tan bueno resulta el modelo EKF con las modificaciones descriptas previamente.

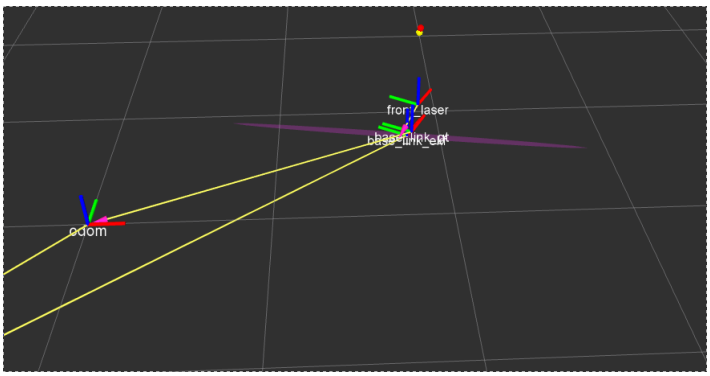
Partiendo de un punto en el que el robot se encuentra frente a tres postes, podemos observar que logra estimar su posición con una matriz de covarianza (en azul) casi nula.



Reduciendo la distancia entre los postes y el robot, si este solo percibe dos al mismo tiempo su covarianza continúa siendo casi imperceptible en la imagen.

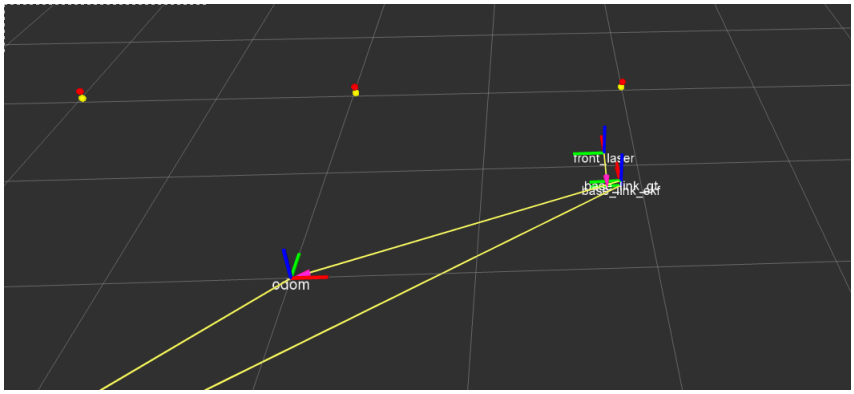


Ahora posicionando el robot para que solo cense un poste, el grado de certeza decae considerablemente. Dado que ahora solo tenemos la referencia al eje x la covarianza en el eje y empieza a aumentar haciendo perceptible el elipsoide violeta ensanchándose sobre ese eje.

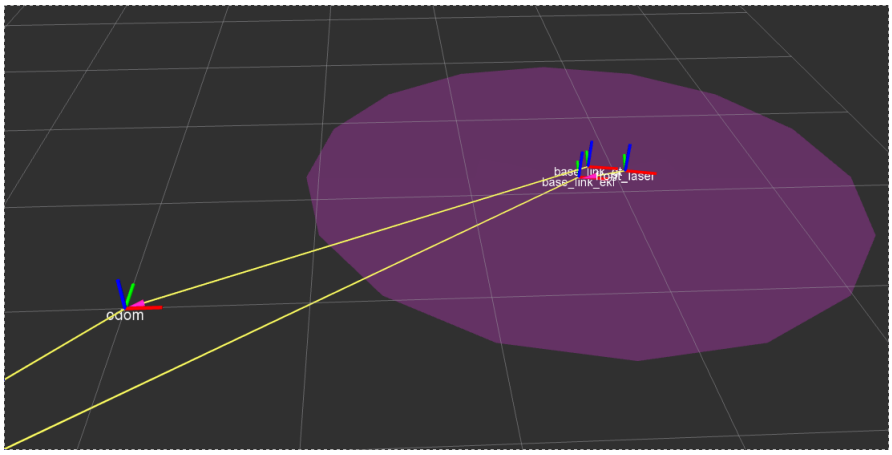


Si ahora volvemos a censar dos postes podemos observar como el robot tiene la posibilidad de

recuperarse y volver a conseguir una predicción precisa de su posición, la covarianza en el eje y se reduce a valores parecidos a los que teníamos antes.



Finalmente, si el robot no es capaz de censar ningún poste, la covarianza tanto en x como en y aumenta.

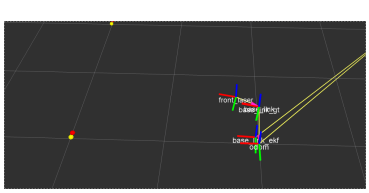


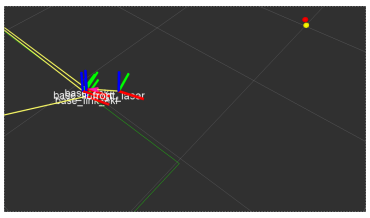
3. Seguimiento de trayectorias a lazo cerrado utilizando localización basada en EKF

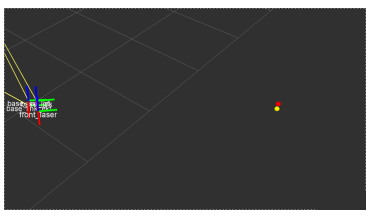
En este apartado volvemos a poner a prueba el robot en el circuito previamente realizado con la trayectoria a lazo cerrado.

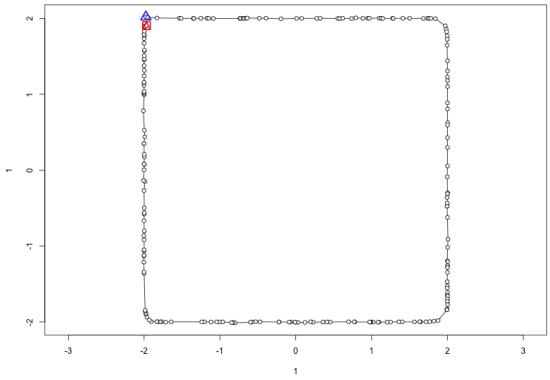
Nuevamente utilizamos RViz para visualizar el error y probamos con $K = 0{,}4, 1$ y 4:

Para $K = 0{,}4$:



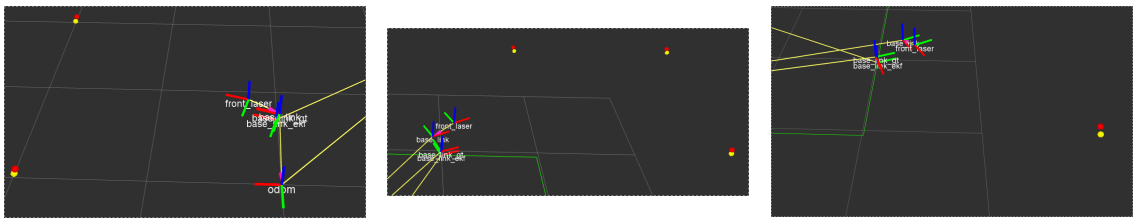


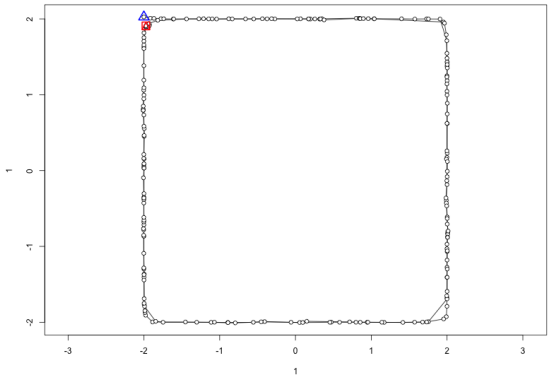




En este caso podemos ver que el base_link_ekf permanece casi pegado al ground_truth durante todo el recorrido. Además contamos con el base_link (estimación odometrica sin utilizar EKF) para analizar de manera comparativa con los otros dos valores. En este caso en particular, el base_link no se aparta demasiado de los otros dos valores.

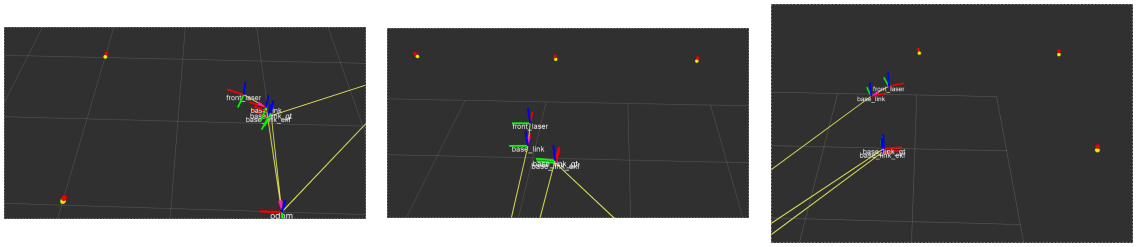
Repetimos el experimento para $K = 1$:

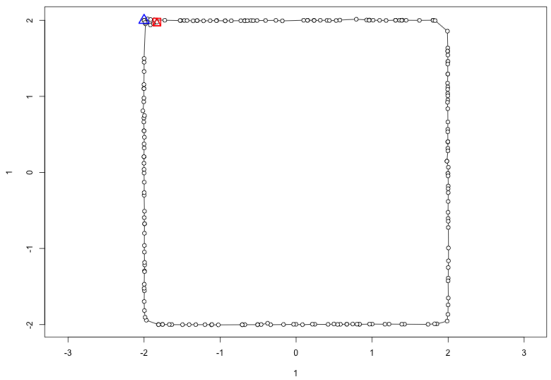




Si bien el base_link_ekf continúa muy cercano al ground_truth, podemos ver que base_link ya para este caso diverge hacia un valor incorrecto.

Finalmente para $K = 4$:



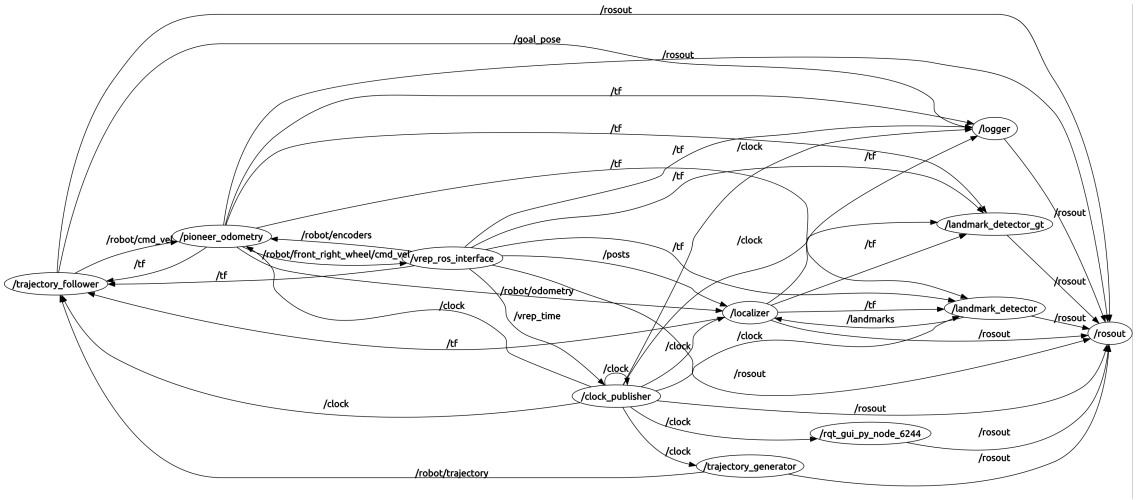


A diferencia del modelo previo de lazo cerrado Kalman permite estimar de manera muy precisa la pose real del robot aún para velocidades tan restrictivas.

En conclusión podemos apreciar que aplicando EKF es posible aplicar valores mucho mas restrictivos sobre la velocidad sin perder precisión en los valores odométricos. Si recordamos en el apartado de control a lazo cerrado esto no era posible ya que la acumulación de errores sistemáticos y del modelo cinemático en sí llevaba a estimaciones que no representaban de manera alguna la realidad.

3.1. Detalles del sistema desarrollado

Para concluir el trabajo presentamos un pantallazo general de todos los nodos trabajando en conjunto:



Sobre esta gran cantidad de mensajes siendo intercambiados en paralelo podemos destacar como lo más relevante al problema los siguientes:

El nodo encargado de la odometría recibe la información de los comandos de velocidad t encoders para enviar su estimación de la pose, esta es capturada por el "trajectory follower"que mide la distancia a la pose del goal juntando esto con lo recibido por el "trajectory generatorz publicando las consignas de velocidades necesarias para avanzar hacia el goal.

Por su parte el nodo "localizer"se suscribe a los tópicos de odometría, la información de senado y la ubicación del robot y postes en el simulado, con esto obtiene las consignas de covarianza y publica las transformaciones desde el marco EKF al del robot.

Por último el detector de landmarks de la IMU usa la información usa la información de scanning para actualizar la información de distancia respecto a los "landmarks".

4. Concluciones

Como conclusiones finales de este trabajo se logró llevar a la práctica una gran cantidad de los conceptos vistos durante la cursada, a partir del análisis hecho sobre un nuevo modelo se pudo pasar de un sistema resuelto (modelo diferencial) a este que inicialmente resultaba una incógnita.

A medida que implementamos los distintos componentes necesarios para conseguir el objetivo propuesto adquirimos conocimientos sumamente útiles sobre el sistema bajo estudio, la herramienta de desarrollo y temas anteriormente abordados (cinemática, seguimiento de trayectorias, filtro de kalman), consideramos que el aprendizaje de esto resulta muy enriquecedor

Centrándonos en los resultados observamos que nuestra implementación presenta buenos resultados y en varias trayectorias no triviales se logró la navegación autónoma dentro del entorno simulado. La experimentación nos permitió ver los atributos que, en su correcta implementación, cada una de las partes partícipes en el sistema (Controlador de posición, seguidor de trayectoria, localizador de Kalman, etc) aportan tanto por separado y funcionando en conjunto.