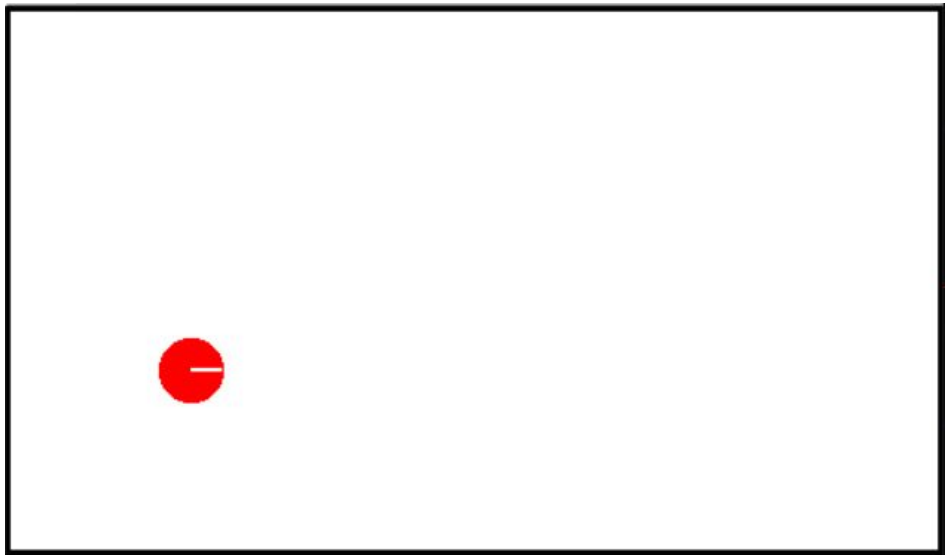
LAB 1: OPERACIONES BASICAS DE MOVIMIENTO Y PERCEPCION EN TURTLEBOT

grupo 5



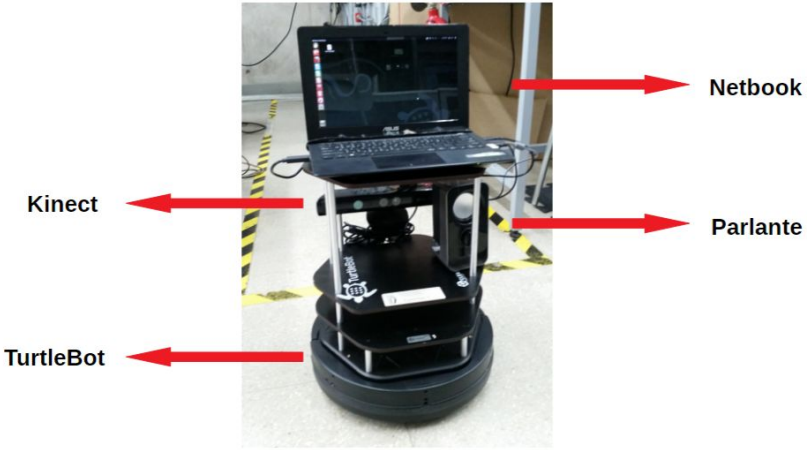
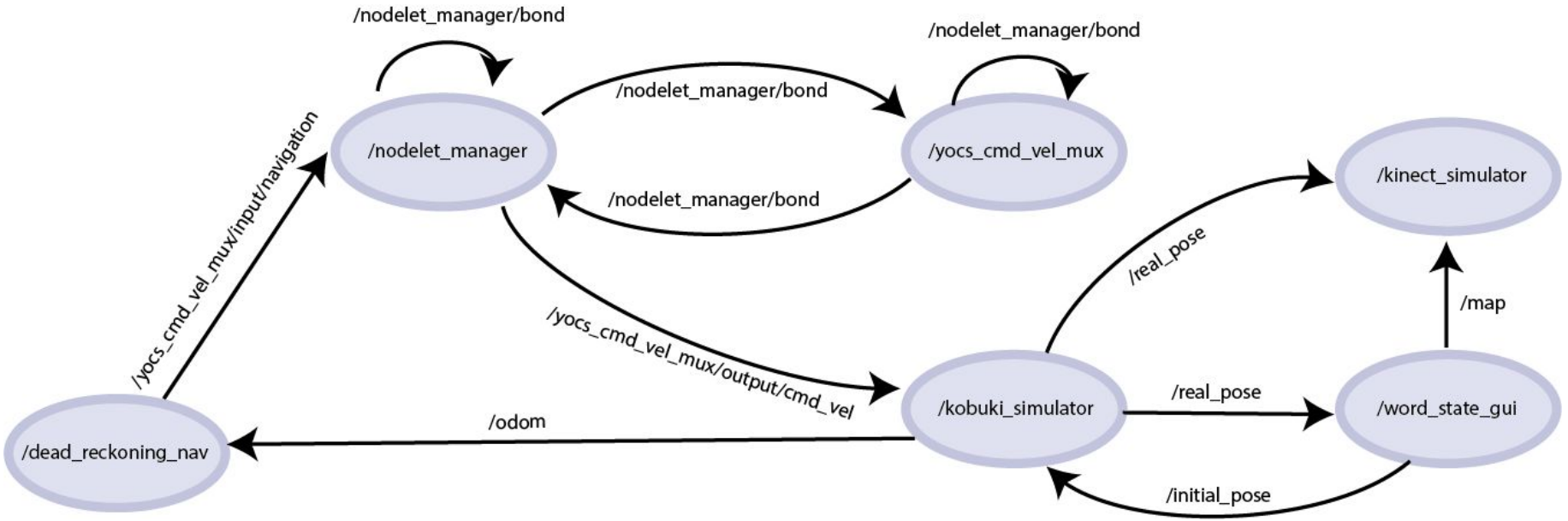

Netbook
Kinect
Parlante
TurtleBot

Diagrama de diseño de software



Programando movimientos del TurtleBot

Nivel 1 : aplicar_velocidad (speed_comand_list)

- argumento de la función: lista de tripletas (v,w,t)
- su objetivo es llegar de un punto A otro B con esos comandos

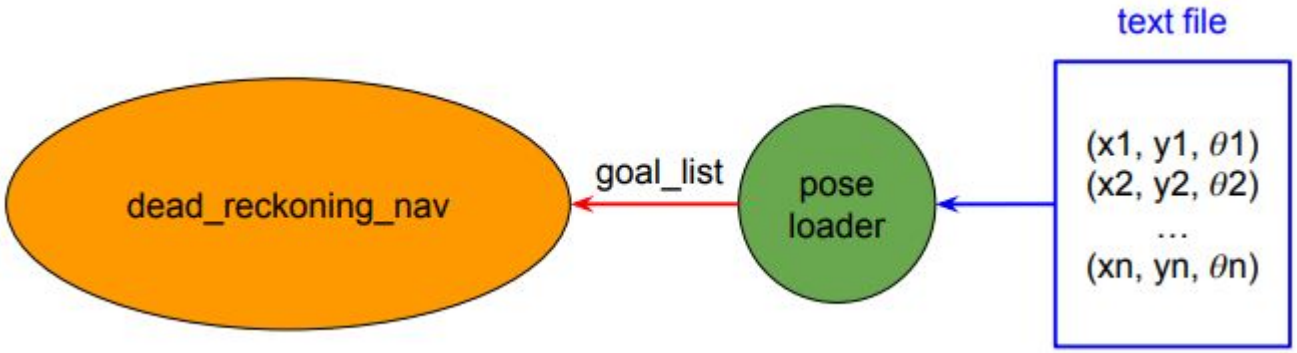
Programando movimientos del TurtleBot

Nivel 2 : mover_robot_a_destino (goal_pose)

- Mueve el robot de una posición actual a un destino goal_pose de tipo (x, y, thetha)
- Calcula la pose final y la envía a aplicar_velocidad en el formato (v,w,t)

Programando movimientos del TurtleBot

Nivel 3 : accion_mover_cb



DEMOSTRACIÓN
(al final de la clase)

roslaunch lab1_pkg avanzar_y_rotar.launch

Avanzar y rotar

Deafío: 3 vueltas de trayectoria cuadrada de 1m **SIN** factor de corrección sobre el tiempo de giro

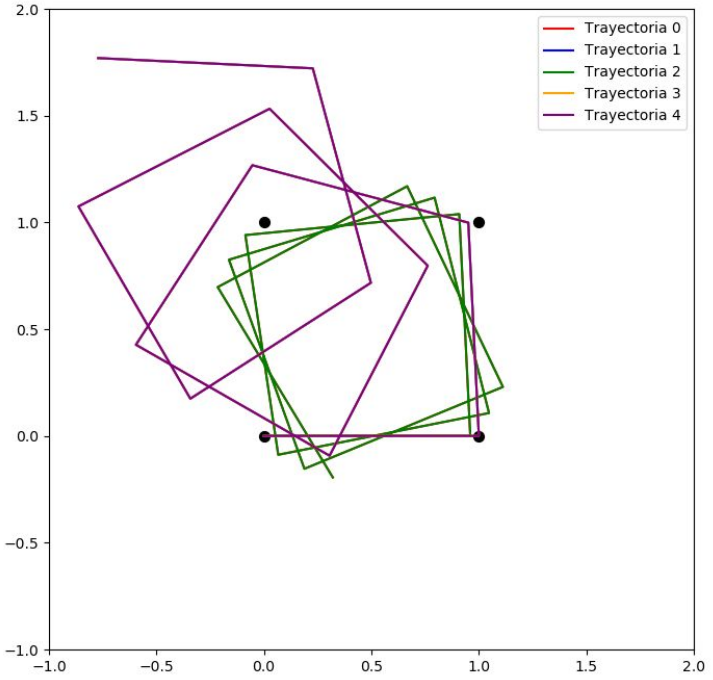
Resultado

Error promedio:

- 1.308 [m]

Desviación estándar:

- 0.761 [m]



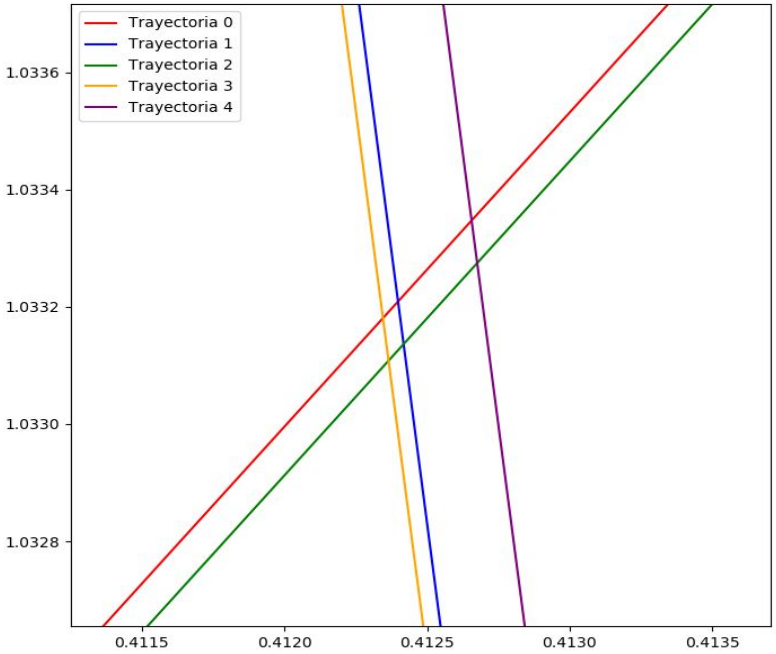
Resultado

Error promedio:

- 1.308 [m]

Desviación estándar:

- 0.761 [m]



Avanzar y rotar

Desafío: 3 vueltas de trayectoria cuadrada de 1m **CON** factor de corrección sobre el tiempo de giro

Resultado

Error promedio:

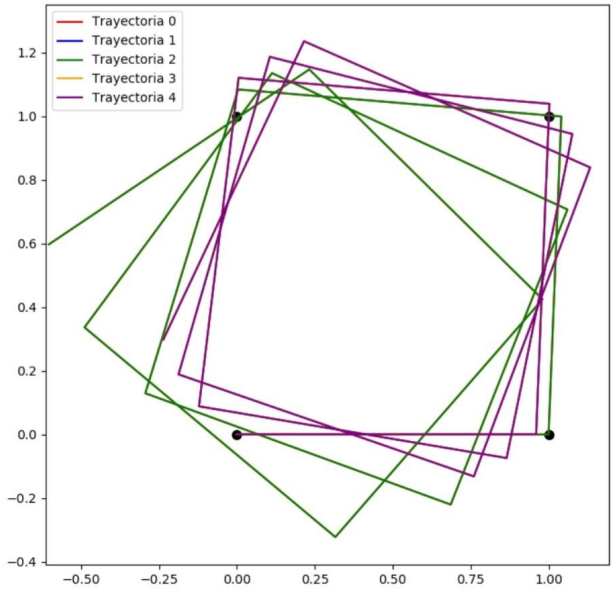
- 0.567 [m]

Desviación estándar:

- 0.229 [m]

Factores de correccion velocidad angular y giro:

- 1.1
- 1.01



¿Qué pasó?

En el primer caso hubo mucho más error que en el que tiene corrección

Hubo tanto un error de ángulo en cada giro, lo que causó un error en la posición final

Entre las 5 iteraciones, siempre nos dieron 2 trayectorias por lo que varias de estas eran colineales

¿Qué lo puede causar?

Factores de imprecisión de software y hardware:

Hardware: odometría no es precisa debido a que puede resbalar, si el terreno es irregular una rueda puede andar más que otra, puede atorarse

Limitaciones de la tecnología que no permite mayor precisión de datos del entorno.

Limitaciones de hardware que limitan la velocidad de procesamiento de la información

Software: Frecuencia de actualización no lo suficientemente rápida que genera pérdidas de información (limitaciones de tecnología)

¿Qué lo puede causar?

Corrección de software entregado para que realice comportamientos más cercanos a la realidad

DEMOSTRACIÓN

roslaunch lab1_pkg avanzar_y_rotar.launch

Percepción básica

Detector de obstáculo

criterio utilizado

El criterio fué detectar obstáculos al costado izquierdo, central y derecha.

- Si solo se activaba derecha/izquierda y el del centro gira 18° hacia el lado contrario
- Si detectaba en todas partes da media vuelta y continúa avanzando

DEMOSTRACIÓN

roslaunch lab1_pkg obstacle_detector.launch

Acción y percepción

Detector de obstáculo y aviso

criterio utilizado

- Cada vez que el robot detecta cualquier tipo de obstáculo lo reporta con sonido

DEMOSTRACIÓN

roslaunch lab1_pkg accion_y_percepcion.launch

CONCLUSIONES GENERALES

- Siempre va a ser necesario aplicar un factor de corrección para que el robot tenga una buena trayectoria

- Idealmente este factor debe estar involucrado con odometría real para descartar errores asociados a ésta y poder corregir la trayectoría

- La cámara es una buena herramienta, sin embargo, usarla por sí sola no entrega resultados fiables

- A diferencia de otros diseños de software para la programación en ROS es muy necesario hacer un diagrama de bloques al inicio para tener una coherencia con la información

CONCLUSIONES GENERALES

- La implementación no se ajusta a fallos de feedback por ejemplo, si en el camino el robot se queda enganchado y las ruedas deslizan no hay forma de tener un feedback

- Lo mejor es tener una gran cantidad de sensores para minimizar la cantidad de errores y tener una claridad más real del entorno