Práctica 1

Víctor Fernández Torres
César Munuera Pérez

1. Resumen

Esta primera práctica tiene el objetivo de poner al alumno en contacto con el robot físico, el simulador y los sistemas de subscripciones a los topics.

Se pide al alumno ir probando diferentes suscripciones para ir obteniendo los datos que nos arroja el robot, tanto para el robot físico como para el simulador.

De esta forma, el alumno irá tomando conciencia de los diferentes valores que el robot puede ofrecer, como las distancias a objetos, distancias recorridas, … Así mismo, al comparar el mismo programa en el simulador y en el robot, el alumno se dará cuenta de las diferencias entre una simulación en un escenario ideal y un escenario real.

2. Introducción

Esta práctica está compuesta por partes que se realizan en simulador y otras en el propio robot, y ejercicios similares al anterior. El código difiere a la hora de hacer referencia a los topics, por ejemplo. Como se deben entregar todos, no podemos pisarlo.

Por ello, y porque facilita la programación en equipo, hemos decidido hacer uso de la herramienta Git. De esta forma, cada vez que se acaba un ejercicio, lo actualizamos poniendo una correcta descripción en los commit. De esta forma, al terminar la práctica, se podrá acceder al listado de commits

3. Caracterización de los sensores de odometría

3.1 Indique y describa la información que nos ofrece el mensaje disponible en el topic odom.Muestre algún ejemplo de captura.

Esto es el contenido del topic “Odom”:

```
odom =

  ROS Odometry message with properties:

    MessageType: 'nav_msgs/Odometry'
         Header: [1×1 Header]
           Pose: [1×1 PoseWithCovariance]
          Twist: [1×1 TwistWithCovariance]
    ChildFrameId: 'robot0'
```

Si entramos para ver los detalles mediante showdetails(odom), obtenemos los siguientes datos:

```
Header
  Stamp
    Sec  :  1677137393
    Nsec :  553168752
  Seq    :  11282
  FrameId :  map_static
Pose
  Pose
    Position
      X :  3.682701653
      Y :  2
      Z :  0
    Orientation
      X :  0
      Y :  0
      Z :  0
      W :  1
  Covariance :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
Twist
  Twist
    Linear
      X :  0.2
      Y :  0
      Z :  0
    Angular
      X :  0
      Y :  0
      Z :  0
  Covariance :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
ChildFrameId :  robot0
```

Podemos ver valores de odometría como la posición del robot en el instante en el que se ha recibido el mensaje, velocidad del robot (tanto angular como lineal), etc.

3.2 Mida la resolución máxima (q) de odometría lineal y angular máxima con las diferentes combinaciones de velocidades propuestas en el simulador STDR. Construya una tabla como la que se muestra a continuación con las 8 combinaciones propuestas.

V (ms-1)	**W (rads-1)**	**q_lineal (m)**	**q_angular (r)**
0.1	0	0.0123	0
0.3	0	0.0600	0
0.5	0	0.0531	0
0.7	0	0.1001	0
0.9	0	0.1029	0

En la medición V = 0.1 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 3 m.

En la medición V = 0.3 m/s, se llegan a realizar aproximadamente 220 tomas de la posición. Se le ha puesto a correr 8 m.

En la medición V = 0.5 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 11 m.

En la medición V = 0.7 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 15 m.

En la medición V = 0.9 m/s, se llegan a realizar aproximadamente 160 tomas de la posición. Se le ha puesto a correr 15 m.

V (ms-1)	**W (rads-1)**	**q_lineal (m)**	**q_angular (r)**
0	0.3	0	0.0287
0	0.7	0	0.0316
0	0.9	0	0.0365

En la medición W = 0.3 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 3 m.

En la medición W = 0.7 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 3 m.

En la medición W = 0.9 m/s, se llegan a realizar aproximadamente 200 tomas de la posición. Se le ha puesto a correr 3 m.

3.3 Mida la resolución máxima (q) de odometría lineal y angular con las diferentes combinaciones de velocidades propuestas en el robot real (en este caso, debe tener en cuenta las aceleraciones y deceleraciones del robot). Construya una tabla como la del apartado anterior con las 8 combinaciones propuestas.

V (ms-1)	**W (rads-1)**	**q_lineal (m)**	**q_angular (r)**
0.1	0	0.1000	0
0.3	0	0.6890	0
0.5	0	0.6990	0
0.7	0	0.9490	0
0.9	0	1.3480	0

V (ms-1)	**W (rads-1)**	**q_lineal (m)**	**q_angular (r)**
0	0.3	0	0.0256
0	0.7	0	0.0348

0	0.9	0	0.0427

4. Caracterización de los sensores de distancia ultrasónicos

4.1 Indique y describa la información que nos ofrece el mensaje disponible en el topic sonar. Muestre algún ejemplo de captura.

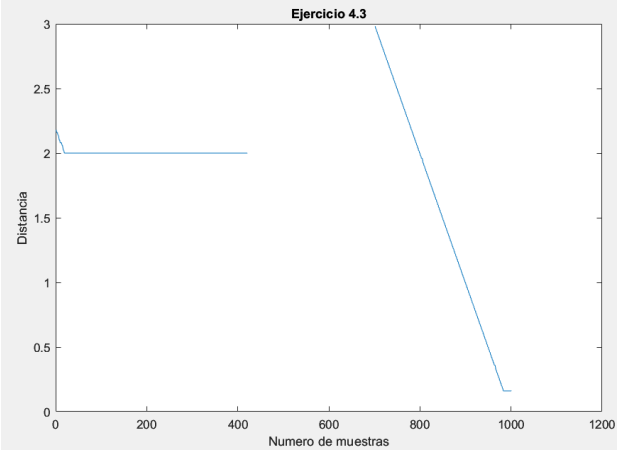
```
ROS Range message with properties:

    MessageType: 'sensor_msgs/Range'
     ULTRASOUND: 0
       INFRARED: 1
         Header: [1×1 Header]
  RadiationType: 0
    FieldOfView: 0.8700
       MinRange: 0.1500
       MaxRange: 3
         Range_: 2

Use showdetails to show the contents of the message


Header
  Stamp
    Sec  :  1677257323
    Nsec :  629451157
  Seq     :  1906
  FrameId :  robot0_sonar_0
RadiationType :  0
FieldOfView   :  0.8700000047683716
MinRange      :  0.1500000059604645
MaxRange      :  3
Range_        :  2
```

El resultado obtenido muestra la estructura del topic sonar. Tiene distintos valores como el rango (Range_), que indica la distancia del sonar0 del robot a algún objeto. También nos dice los rangos en los que funciona, el mínimo y el máximo.

4.2 En el simulador STDR, posicione el robot de tal forma que exista una distancia de 2m de uno de los sensores sónar concretos del robot. ¿Qué posición y sónar se ha elegido?

Hemos elegido el sensor sonar S0, que es el delantero. Para ello, hemos situado al robot en la posición x = 5.8701, y = 5.

4.3 Obtenga 1.000 medidas de distancia del sensor sónar elegido y dibuje en una gráfica (comando plot en Matlab) la distancia medida. ¿Son estables las medidas? ¿Hay ruido en la medida? En caso afirmativo, calcule el valor máximo, medio y la varianza del ruido.



Hemos realizado estas medidas un par de veces, obteniendo los mismos resultados siempre. Esto se debe a que el simulador muestra un "ambiente ideal".

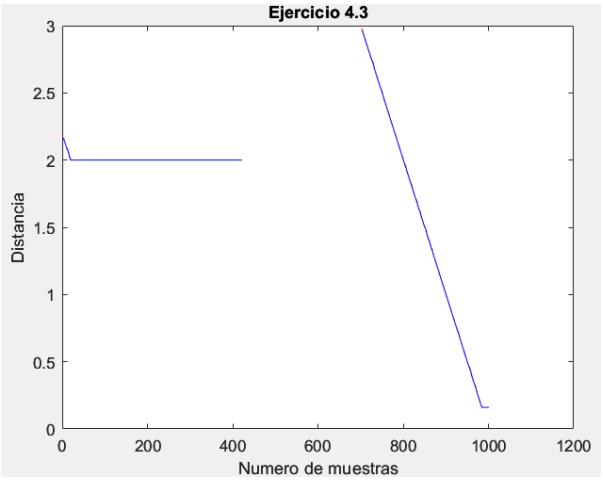
Por otro lado, sobre el ruido, podemos observar que no tenemos.

En la gráfica, el primer leve descenso hasta 2m se debe a que se encuentra muy cerca de una pared, y se queda a la misma distancia hasta las 400 mediciones.

Como sigue moviéndose, llega a un momento que ese sonar no detecta ningún obstáculo, desde las 400 hasta las 700 mediciones aproximadamente. Hasta que detecta una pared y se estampa con ella.

Al finalizar la práctica supimos que el robot debía estar parado. Sin embargo, esto nos ha servido más para conocer el funcionamiento real, y ver si tenía ruidos o inestabilidades.

4.4 Implemente un filtro media móvil con los últimos 5 valores de distancia y dibuje en una gráfica. ¿Son más estables las medidas que en el caso anterior? ¿Sería útil este método si el robot está en movimiento en lugar de permanecer estático?



Este es el resultado de aplicar un filtro de media móvil al código.

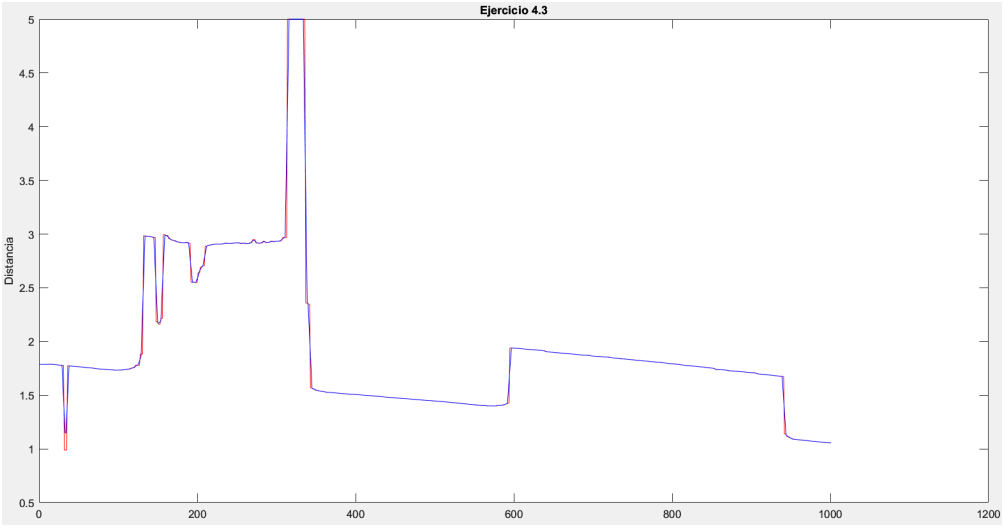
Como podemos observar, el resultado es el mismo que el anterior, ya que los resultados obtenidos de la ejecución del programa en el sobot del simulador no tienen ruido alguno.

Se ha empleado la función *movmean*, y se ha impreso en la gráfica primero en rojo sin filtro, luego en azul con filtro, por eso vemos que no hay cambios.

4.5 Repita pasos 2-4 con el robot real.

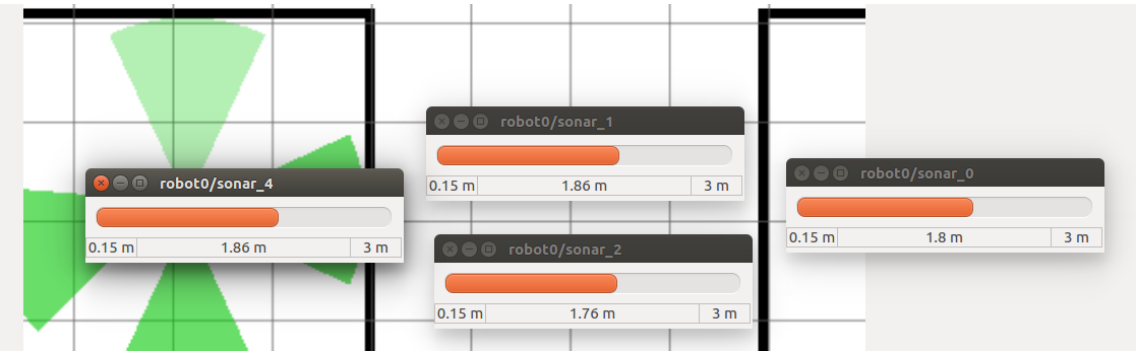
Volvemos a ejecutar el mismo código, con el filtro media móvil en el robot real.

En esta gráfica podemos observar la señal original (de color rojo), y la señal filtrada (azul). Es cierto que ahora, a diferencia del ejemplo anterior en el robot del simulador, se puede llegar a apreciar algo más la diferencia entre las señales. Sin embargo, esta diferencia no es muy grande ya que la original no tiene mucho ruido.



Nosotros hemos implementado el código pensando que el robot DEBÍA estar en movimiento. La diferencia, como se puede apreciar, es mínima. Sin embargo, cuando hemos probado estando el robot parado, la diferencia es nula. Por tanto, esto sólo resultaría útil si el robot está en movimiento.

4.6 En el simulador, posicione al robot en la casilla X. Estando el robot perfectamente paralelo a las paredes de la celda, seleccione las medidas de los sensores sonar del robot que podrían resultar útiles para obtener las cuatro rectas que definen las paredes que lo rodean. Compruebe que las orientaciones de las rectas son paralelas dos a dos y perpendiculares entre ellas. Defina una función de calidad para obtener el grado de confianza de dichas paredes empleando, por ejemplo, la relación entre las diferentes pendientes.



En esta captura mostramos las distancias de los sensores a las 4 paredes. Como se puede ver, la diferencia es mínima, el robot está casi perfectamente centrado.

Ahora, ¿cómo hemos realizado la función de fiabilidad? Pues estando el robot perfectamente centrado en esas posiciones, hemos decidido realizar 100 tomas de cada sensor, cada uno apuntando a su correspondiente pared. El sónar 0 a la de la derecha, el sónar 1 a la de arriba, … Este planteamiento ha sido el mismo para todas las funciones de fiabilidad de detección de una pared.

Cada vez que el robot detectase algo a menos de 3 metros (su rango máximo), aumentamos un contador. Si de las 100 tomas, hemos detectado 70, es que es fiable a un 70%, ya que nosotros sabemos que hay una pared a 2 metros.

Teniendo en cuenta estos datos, estos son los resultados obtenidos:

```
La fiabilidad del sonar 0 es del 100 %.
La fiabilidad del sonar 1 es del 100 %.
La fiabilidad del sonar 2 es del 100 %.
La fiabilidad del sonar trasero es del 100 %.
```

4.7 Diseñe una función que indique, mediante un código, el número de paredes que se encuentra el robot en sus laterales. La codificación que se puede emplear para proporcionar la salida es la que se muestra en la Figura 4. Además, indique el grado de confianza basado en la función de calidad definida en el apartado anterior.

Lo que hemos implementado es una cadena de *"if",* que chequea en cada uno la distancia que detecta cada sensor. Sabiendo las posiciones de cada sensor, podemos conocer que si cierto sensor está detectando algo a menos de 3m y otro no, es que detecta ciertas paredes (las de que se nos pide codificar).

Por ejemplo, si el sónar 0 (delantero) es el único que detecta algo a menos de 3m, quiere decir que se debe imprimir "1".

Sin embargo, para el robot real hay que hacer un par de modificaciones. No solo para saber cual es cada sensor (ya que no son los mismos), este da 5 siempre que no detecte nada. Es decir, que mientras que el simulador da infinito, el real da 5 cuando no detectan nada.

Ver código para mejor comprensión.

4.8 Comprueba los resultados de la función diseñada en el apartado anterior tanto con el simulado STDR como con el robot real y con las 16 posibles combinaciones. Complete la tabla siguiente indicando los resultados.

Robot_real / Simulador	**Combinación real**	**Combinación identificada**	**Grado de confianza**
Simulador	1	1	100 %
Simulador	2	2	100 %

Simulador	3	3	100 %
Simulador	4	4	100 %
Simulador	5	5	100 %
Simulador	6	6	100 %
Simulador	7	7	100 %
Simulador	8	8	100 %
Simulador	9	9	100 %
Simulador	10	10	100 %
Simulador	11	11	100 %
Simulador	12	12	100 %
Simulador	13	13	100 %
Simulador	14	14	100 %
Simulador	15	15	100 %
Robot_real	1	1	89 %
Robot_real	2	2	91 %
Robot_real	3	3	97 %
Robot_real	4	4	91 %
Robot_real	5	5	90 %
Robot_real	6	6	91 %
Robot_real	7	7	96 %
Robot_real	8	8	91 %
Robot_real	9	9	92 %
Robot_real	10	10	91 %
Robot_real	11	11	98 %
Robot_real	12	12	87 %
Robot_real	13	13	94 %
Robot_real	14	14	98 %
Robot_real	15	15	93 %

5. Sensores de distancia láser

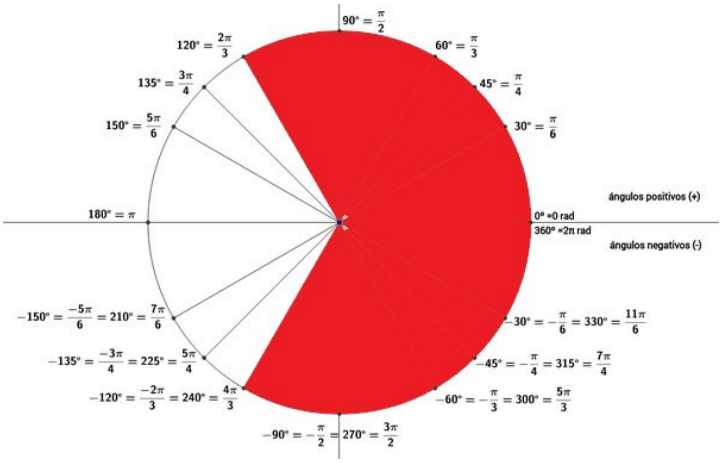
5.1 Indique y describa la información que nos ofrece el mensaje disponible en el topic laser. Muestre algún ejemplo de captura.

```
ROS LaserScan message with properties:

     MessageType: 'sensor_msgs/LaserScan'
          Header: [1×1 Header]
        AngleMin: -2.0944
        AngleMax: 2.0944
  AngleIncrement: 0.0063
   TimeIncrement: 0
        ScanTime: 0
        RangeMin: 0.0600
        RangeMax: 4.0900
          Ranges: [667×1 single]
     Intensities: [0×1 single]
```

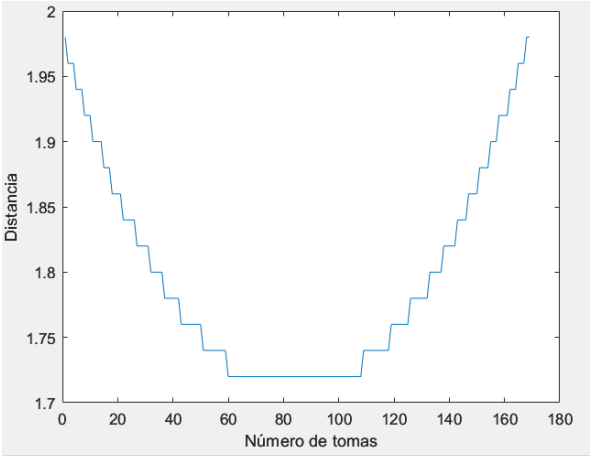
```
Header
  Stamp
    Sec  :  1678990085
    Nsec :  443889589
  Seq    :  48114
  FrameId :  robot0_laser_0
AngleMin       :  -2.094395160675049
AngleMax       :  2.094395160675049
AngleIncrement :  0.006289475131779909
TimeIncrement  :  0
ScanTime       :  0
RangeMin       :  0.05999999865889549
RangeMax       :  4.090000152587891
Ranges         :  [1.979999899864197, 1.959999918937683, 1.959999918937683, 1.959999918937683, 1.939999938011169, 1.
Intensities    :  []
```

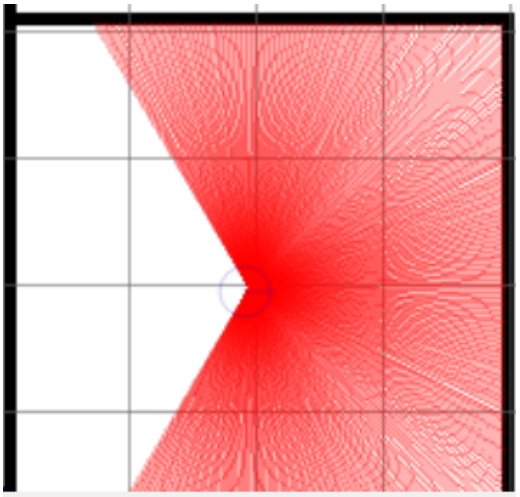
5.2 Repita los pasos 2-4 y 6-8 del apartado anterior con el sensor láser.

Como se ha podido observar en el ejercicio anterior, el ángulo máximo y mínimo que es capaz de abarcar el sónar va desde -2.0943 hasta 2.0943. O lo que es lo mismo, desde 2π/3 hasta -2π/3.



Para el **punto 2**, hemos situado en este lugar al robot, obteniendo 170 haces a menos de 2m.



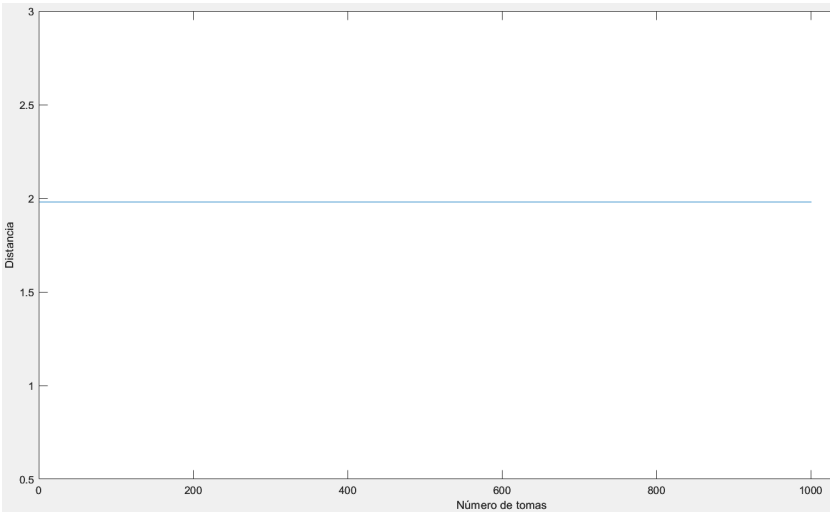


No hemos elegido un haz concreto, sino todos los que muestran una distancia menor a 2m.

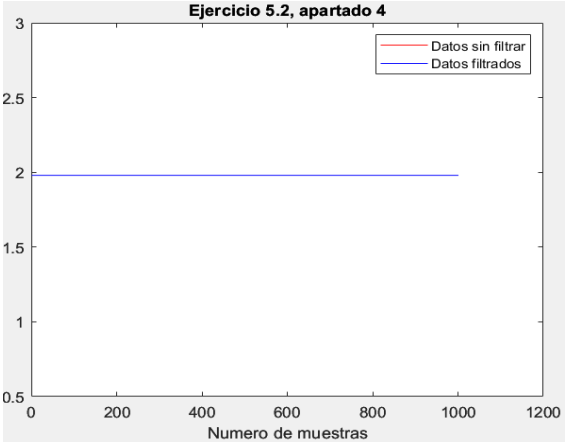
Este es el resultado del **punto 3**. En este caso, si hemos elegido un haz concreto, el primero. Como para comprobar la estabilidad y el ruido no es necesario ver uno en concreto, hemos usado simplemente el primero.

Para ello, en cada iteración, obtenemos el primer haz del array de haces, y realizamos 1000 iteraciones.

Como se puede observar en el plot, no tiene ruido. Es estable ya que lo hemos ejecutado un par de veces y en todas obtenemos el mismo resultado.



Para el apartado del filtro media móvil (**punto 4**), se ha usado la misma función que para los sónar. En el mismo plot, podemos apreciar la señal sin filtrar (roja), y la señal filtrada (azul).

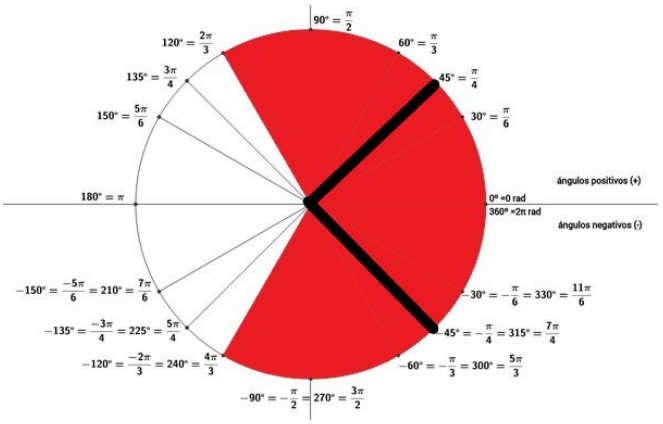


Como se puede observar, la diferencia es nula. Al estar el robot parado, no es necesario aplicar un filtro.

Además, la señal no tenía ruido y era estable.

Por tanto, esto solo sería útil para un robot en movimiento.

Para el **punto 6**, hemos decidido dividir entre 3 el total de haces del láser de la siguiente forma:



De esta manera, podemos distinguir 3 posibles paredes. Al tratar cada sección de forma independiente, podemos comprobar el número de veces que todos los haces de cada sección detectan una pared.

Si de 100 tomas, todos los haces de una misma sección nos informan de la existencia de una pared, es que la fiabilidad es del 100%.

Hay 667 haces en total, ya que el array de distancias tiene 667 elementos. De esta forma, usaremos 222 haces para las secciones laterales y 223 para la delantera. Se realizarán, por tanto, 100 tomas distintas de los 222 o 0223 haces.

Primero se separarán en 3 arrays los haces; izquierdo, derecho y central. Después, en un bucle de tamaño 100, se realizará lo siguiente:

- Un bucle para contar el número de haces cuya distancia sea menor a 4 metros. De esta forma, sabemos que tenemos pared. Esto se hace en cada array.
- Sabiendo el número de “detecciones de pared”, calculamos el porcentaje de detección de pared. Esto lo sumamos a un acumulador, hay uno por array (3).

Fuera del bucle, el acumulador de porcentajes será dividido entre las iteraciones (100), para obtener la media de los porcentajes. De esta forma, conocemos la media del porcentaje de detección, de cada toma de haces que se realiza.

Para el **punto 7**, se reutiliza el código del punto 6. Básicamente hemos asumido que la codificación de pared debía tener el mismo planteamiento que el de la precisión de detección de pared. Por tanto, una pared sería detectada en el caso de que más de la mitad de los haces diesen positivo.

Entonces, si un porcentaje de detección es mayor a 51, es que hay pared. Así podemos decir que tenemos una codificación con su respectivo porcentaje de probabilidad.

Punto 8:

Robot_real / Simulador	**Combinación real**	**Combinación identificada**	**Grado de confianza**
Simulador	1	1	100 %
Simulador	2	2	100 %
Simulador	3	IMPOSIBLE	IMPOSIBLE
Simulador	4	4	100 %
Simulador	5	5	100 %
Simulador	6	IMPOSIBLE	IMPOSIBLE
Simulador	7	7	100 %
Simulador	8	IMPOSIBLE	IMPOSIBLE
Simulador	9	9	100 %
Simulador	10	IMPOSIBLE	IMPOSIBLE
Simulador	11	IMPOSIBLE	IMPOSIBLE
Simulador	12	12	100 %
Simulador	13	IMPOSIBLE	IMPOSIBLE
Simulador	14	14	100 %
Simulador	15	IMPOSIBLE	IMPOSIBLE
Robot_real	1	1	
Robot_real	2	2	
Robot_real	3	3	
Robot_real	4	4	
Robot_real	5	5	
Robot_real	6	6	
Robot_real	7	7	
Robot_real	8	8	

Robot_real	9	9	
Robot_real	10	10	
Robot_real	11	11	
Robot_real	12	12	
Robot_real	13	13	
Robot_real	14	14	
Robot_real	15	15	

La columna de la fiabilidad del robot real no la hemos rellenado por lo siguiente:

El robot real tiene 360 haces, uno por grado. Comienza desde -pi hasta pi. Por tanto, el array de haces va desde el haz 1 en -pi hasta el 360 en pi.

Nosotros hemos hecho 4 divisiones, como el robot del simulador. De forma que se extrae:

- 45 haces para un array trasero.
- 90 haces para el derecho.
- 90 haces para el central.
- 90 haces para el izquierdo.
- 45 haces haces por la derecha para el array trasero.

De esta forma simulamos la existencia de 4 láseres que ven las 4 posibles paredes. Se extraen en ese órden justo porque es el órden del array de rangos, de -pi a pi.

Sin embargo, como en los pasillos no es posible realizar medidas reales de menos de 7m, que es lo que llega a captar el láser, tenemos dudas de si realmente funciona.

Estamos convencidos de que la lógica es correcta y está bien implementado. Sin embargo, los resultados no nos terminan de convencer del todo.

5.3 ¿Tiene más o menos ruido que el sensor sonar?

Después de haber realizado muchas pruebas correspondientes a los puntos anteriores, y haber realizado plots, podemos confirmar que no hemos apreciado ruido en el robot simulado.

Sin embargo, creemos que en un ambiente real, con el robot real, tendremos más ruido con el láser que con el sonar. Esto se debe a que el láser tiene muchas más tomas, retorna muchos más valores, que el sónar. Por tanto, la probabilidad de tener más ruido es mayor en el láser.

5.4 ¿Cómo es el grado de confianza empleando este sensor?

Tal y como se ha explicado en el 5.2 en los dos últimos apartados, el grado de confianza en el robot simulado es 100%. No hay fallo. Hay que tener en cuenta que con que 1 haz no detectase pared, baja el porcentaje. Pero es la media de 100 porcentajes acumulados.

6. Actuadores en ROS

6.1 Diseñe una función avanzar que reciba como parámetro de entrada la distancia a avanzar (2m, 4m...).

La función creada recibe como parámetros la distancia a recorrer, la subscripción (necesaria para poder recibir datos), y el publisher, para poder mandar el mensaje con las velocidades al robot.

Básicamente se comprueba la posición actual del robot, y se le pide que avance hasta que la distancia recorrida sea x metros superior a la primera medida. Esos x metros son los que se le pasa como parámetro.

Cuando se ha llegado a la distancia deseada, se sale del bucle y se le manda un mensaje con velocidad 0, para que se pare.

6.2 Diseñe una función girar que se reciba como parámetro de entrada el ángulo a girar (45o, 90o,...).

Para esta función hemos decidido calcular la diferencia de posiciones para saber cuantos grados se han movido, para ello transformamos el número de grados al sistema de representación que tiene el robot. Dependiendo de si los grados introducidos son mayores o menores que 0, el robot girará hacía una dirección o la otra.

7. Conclusiones

En esta práctica se han aprendido muchos aspectos en cuanto a la adquisición de los datos que estos robots pueden llegar a captar.

También hemos aprendido que, tal y como sospechábamos, los valores y datos que se pueden llegar a captar en el robot simulado son de un “ambiente ideal”. Los que se captan en el robot real vienen con más ruido e inestabilidades.

Sin embargo, nos ha parecido un muy buen sistema el de programar primero para el robot simulado, y luego adaptarlo para el robot real. Las adaptaciones son mínimas, y el tiempo que se pierde realizando pruebas con el real es superior. La lógica de dicho programa es la misma, aunque luego se deba cambiar las suscripciones, los elementos de donde se reciben datos, … Además, que con el simulador se puede trabajar desde cualquier lado.