Introducción a la Robótica Móvil

Primer cuatrimestre de 2018

Departamento de Computación - FCEyN - UBA

Percepción (Parte II) - clase 10

Sensado con LASER e IMU

Sensado

¿Qué sensores vimos hasta ahora?

- Sensor de contacto (*bumper*).
- Telémetro infrarrojo.
- Sonar de ultrasonido.
- Odómetro (*encoder*).

Para poder resolver los problemas que vamos a estudiar en la segunda parte de la materia (localización, planificación de trayectorias, construcción de mapas) no alcanzan. Hoy vamos a ver:

- IMU (Inertial Measurement Unit) compuesta por giróscopo y acelerómetro.
- LASER (Light Amplification by Stimulated Emission of Radiation).

Giróscopo: principio de funcionamiento

El giróscopo funciona gracias al efecto Coriolis:

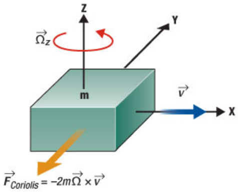


Figura: Efecto Coriolis.

- Cuando una masa m se mueve en una dirección $\mathbf{v}$ respecto de un sistema de referencia con una velocidad angular $\mathbf{\Omega_z}$, la masa experimenta una fuerza en la dirección perpendicular al eje de rotación del sistema y a la velocidad del cuerpo (efecto Coriolis).
- El desplazamiento físico de la masa causado por el efecto Coriolis puede ser leído por un sensor capacitvo.

Giróscopo: principio de funcionamiento

En general, los giróscopos MEMS (Microelectromechanical Systems) usan una configuración de diapasón:

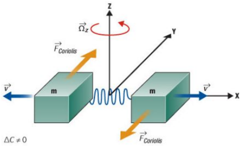


Figura: Efecto Coriolis en una configuración de diapasón.

- Dos masas oscilan constantemente en direcciones opuestas. Al aplicar una velocidad angular al sistema, las fuerzas de Coriolis actúan en sentidos contrarios para cada masa, resultanto en un diferencia capacitiva.
- Si se aplicase una aceleración lineal sobre el sistema, las masas se desplazarían en la misma dirección, lo cuál no genera tal diferencia.
- Esta diferencia capacitiva es proporcional a la velocidad angular, y puede ser convertida en una señal analógica o digital.

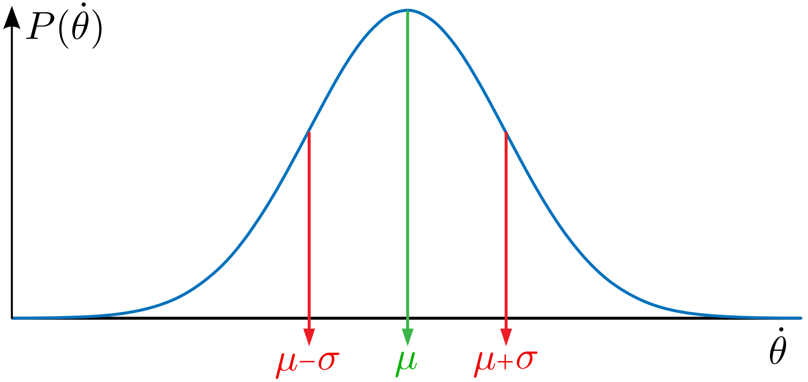
Giróscopo: modelo

- El giróscopo está sujeto a un sesgo (*bias*) y a un ruido.
- Utilizamos un modelo para describir la velocidad angular ${}^{IMU}\boldsymbol{\Omega} = [\omega_x,\ \omega_y,\ \omega_z]$ medida por el sensor en un instante de tiempo:

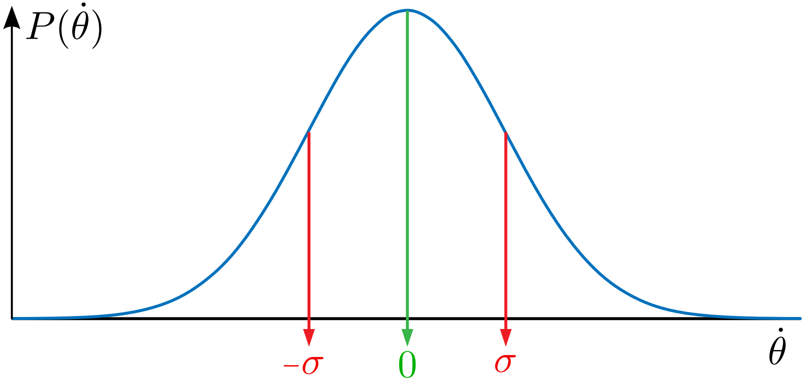
$$ {}^{IMU}\boldsymbol{\Omega} = {}^{IMU}\boldsymbol{\Omega}^{*} + \mathbf{b}_{\boldsymbol{\Omega}} + \boldsymbol{\mu}_{\boldsymbol{\Omega}} $$

  a partir de la velocidad angular real del sistema ${}^{IMU}\boldsymbol{\Omega}^{*}$, un término de bias constante $\mathbf{b}_{\boldsymbol{\Omega}}$ y un ruido (blanco) instantáneo de medición $\boldsymbol{\mu}_{\boldsymbol{\Omega}}$.

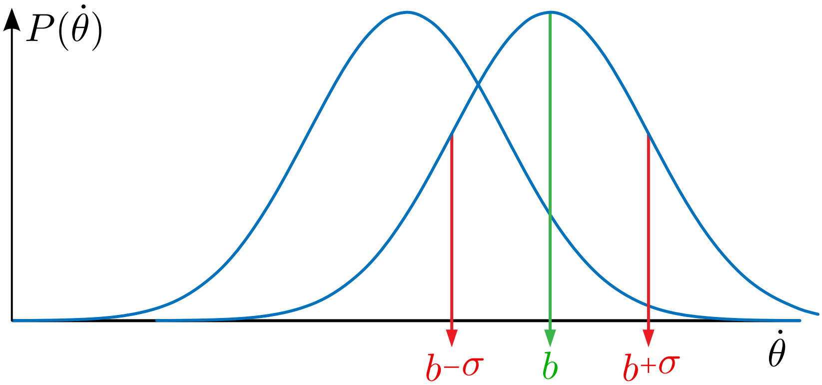
Ruido blanco y bias



Ruido blanco y bias



Ruido blanco y bias



¿Como se podría corregir esto?
¿Cuantos bias existen en realidad?

Giróscopo: calibración

¿Por qué el **giróscopo** necesita calibrarse?

- Errores de calibración de fábrica.
- Cambios de temperatura (*warm-up effect*).
- Cambios en la fuerza de gravedad (dependiendo de la aplicación).
- Si queremos obtener la velocidad angular real ${}^{IMU}\boldsymbol{\Omega}^*$, hace falta poder estimar los otros términos.
- $\boldsymbol{\mu}_{\boldsymbol{\Omega}}$ lo modelamos como un ruido blanco gaussiano, y por lo tanto con media $\mathbf{0}$. Un promedio del mismo tenderá a $\mathbf{0}$.
- Entonces, suponiendo que podemos dejar el giróscopo completamente quieto en un intervalo de tiempo (de calibración), si tomamos el promedio de n mediciones:

$$\frac{\sum {}^{IMU}\boldsymbol{\Omega}}{n} = \frac{\sum {}^{IMU}\boldsymbol{\Omega}^*}{n} + \frac{\sum \mathbf{b}_{\boldsymbol{\Omega}}}{n} + \frac{\sum \boldsymbol{\mu}_{\boldsymbol{\Omega}}}{n} = \frac{\mathbf{0}}{n} + \frac{n \cdot \mathbf{b}_{\boldsymbol{\Omega}}}{n} + \frac{\mathbf{0}}{n} = \mathbf{b}_{\boldsymbol{\Omega}}$$

obtenemos una estimación del sesgo o bias del sensor.

Giróscopo: integración

- El girósocopo nos da una estimación de la velocidad angular para cada instante de tiempo.
- Sin embargo, nos interesa saber la orientación del robot, no su velocidad angular.
- Podemos estimar la orientación integrando numericamente la velocidad angular en pequeños incrementos:

$$\Delta\theta = \Delta t \left({}^{IMU}\boldsymbol{\Omega} - \mathbf{b}_{\boldsymbol{\Omega}}\right)$$
$$\theta_{t_{k+1}} = \theta_{t_k} + \Delta\theta$$

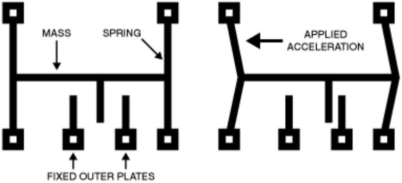
donde ${}^{IMU}\boldsymbol{\Omega}$ es la última velocidad angular medida, $\mathbf{b}_{\boldsymbol{\Omega}}$ es el bias previamente estimado y $\Delta t = t_{k+1} - t_k$ es el tiempo transcurrido entre la última medición y la actual.

¿Que sucede cuando trabajamos con cuaterniones?

$$Q_{t_{k+1}} = Q_{t_k} \cdot \Delta Q$$

Acelerómetro: principio de funcionamiento

El acelerómetro nos permite medir las aceleraciones.



- Un acelerómetro MEMS típicamente consiste de una masa con la libertad de moverse en una dimensión, y dos conjuntos de placas capacitivas: uno de ellos sujeto al sustrato, y otro sujeto a la masa.
- Una aceleración del sistema, produce un desplazamiento de la masa, generando una diferencia en la capacitancia entre las placas fijas y móviles.
- A partir de esa diferencia en la capacitancia se estima la aceleración.

Acelerómetro: modelo

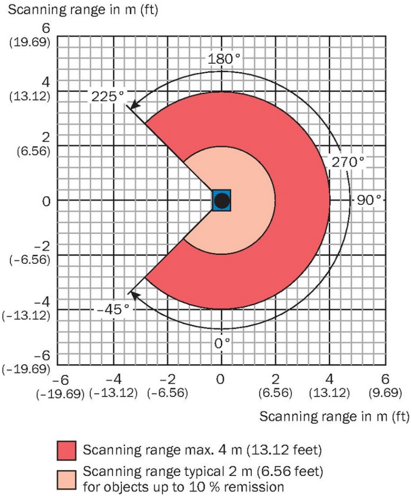
Al igual que con el giróscopo, utilizamos un modelo para describir la aceleración ${}^{IMU}\mathbf{a} = [a_x,\ a_y,\ a_z]$ experimentada por el sensor en un instante de tiempo:

$$
{}^{IMU}\mathbf{a} = {}^{IMU}\mathbf{a}^* - {}^{IMU}\mathbf{g}^* + \mathbf{b}_a + \boldsymbol{\mu_a}
$$

donde $\boldsymbol{\mu_a}$ es el ruido (blanco) instantáneo de la medición y $\mathbf{b}_a$ un desvío constante (bias) de los valores.

En general se puede considerar que $\mathbf{b}_a = 0$ ya que suele ser despreciable.

Telémetro láser (o láser)

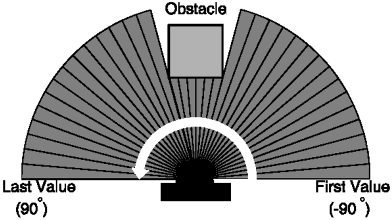


Datos importantes:

- Apertura angular (min, máx).
- Resolución o incremento.
- Rango.

Telémetro láser: mediciones

- Una medición en un instante de tiempo determinado, se suele representar con un arreglo de n posiciones.
- Cada posición representa la medición en una dirección angular φ distinta, y cada valor indica la distancia r que midió el láser en esa dirección particular.
- Luego, para cada instante de medición obtenemos una colección de coordenadas polares (r_i, φ_i).



El ángulo φ_i correspondiente a la posición i del arreglo se puede encontrar como:

$$\varphi_i = \varphi_{min} + i\frac{\varphi_{max} - \varphi_{min}}{n}$$

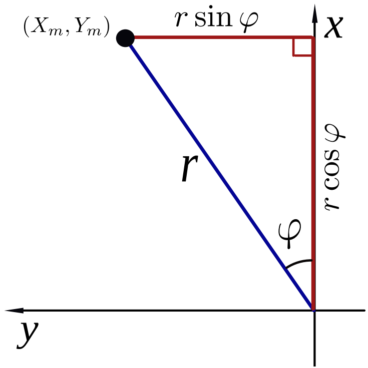
- Cuando detectamos obstáculos, nos es útil saber su posición en coordenadas cartesianas (x_i, y_i), no polares.
- Para pasar un punto representado en coordenada polares (r, φ) a una representación cartesiana (x, y) hacemos:

$$X_m = r \cdot \cos(\varphi)$$

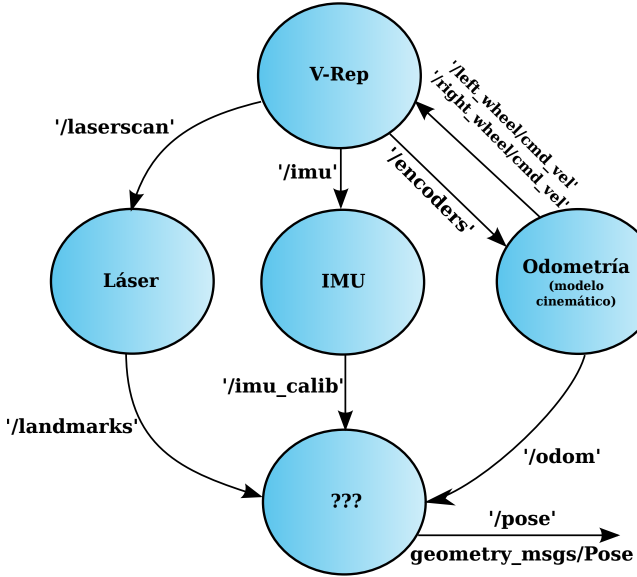
$$Y_m = r \cdot \sin(\varphi)$$

$$r = \sqrt{X_m^2 + Y_m^2}$$

$$\varphi = atan2(Y_m, X_m)$$



Para el Taller, ¿qué vamos a ver ahora?



Mensajes de la unidad de movimiento inercial

sensor_msgs/Imu

std_msgs/Header header
geometry_msgs/Quaternion orientation
geometry_msgs/Vector3 angular_velocity
geometry_msgs/Vector3 linear_acceleration

std_msgs/Header

uint32 seq
time stamp
string frame_id

La información de la orientación no viene dada directamente por el sensor, sino que depende de la integración que utilicemos sobre las mediciones.

Mensajes del sensor Láser

sensor_msgs/LaserScan

std_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 range_min
float32 range_max
float32[] ranges

std_msgs/Header

uint32 seq
time stamp
string frame_id

La información va a estar en relación al marco de coordenadas del láser. Se aplican transformaciones para **traducir la información al marco de coordenadas del robot**.

Mensajes de las referencias/landmarks

robmovil_msgs/LandmarkArray

std_msgs/Header header
Landmark[] landmarks

robmovil_msgs/Landmark

float32 range
float32 bearing

Un poco de la librería geométrica tf

http://docs.ros.org/indigo/api/tf/html/c++/

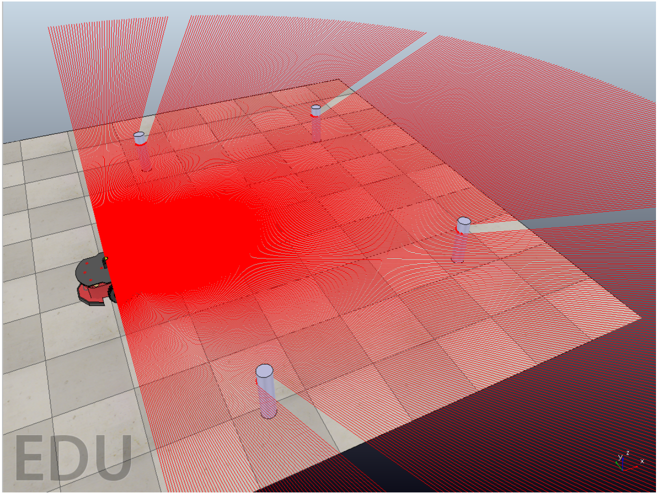
tf::Vector3:

- **vec1** $+$ **vec2** : suma de vectores
- **vec1** $\cdot$ **vec2** : multiplicación de vectores **elemento a elemento**
- **vec1.dot(vec2)** : producto escalar entre vectores
- **vec1** $\cdot$ **scalar** : multiplicación de vectores por un escalar
- **vec1** / **scalar** : división de vectores por un escalar
- **vec1.length()** : norma 2 del vector
- **vec1.normalize()** : se normaliza el vector

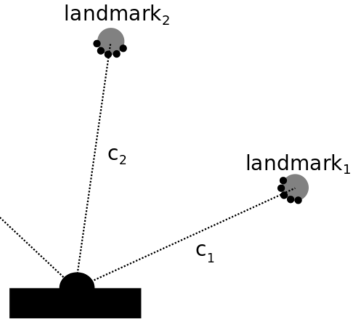
tf::createQuaternionFromYaw:

- calcula ΔQ cuando queremos resolver $Q_{t_{k+1}} = Q_{t_k} \cdot \Delta Q$

Detección de postes mediante un sensor láser



Detección de postes mediante un sensor láser



l_k^i : son las k mediciones del landmark i $\qquad l_{min} = min\_dist(l_k^i, laser)$

$$c_i = l_{min} + normalize(l_{min}) \times radio(\text{landmark}_i)$$