Robótica Móvil
Estimación Experimental del Error del LIDAR

Iñaki Rañó

Notas

El objetivo de esta práctica es familiarizarse con el modelado de lecturas de sensores como variables aleatorias Gaussianas, específicamente lecturas proporcionadas por el sensor LIDAR de los robots Turtlebot. Para ello debemos entender el funcionamiento de dicho sensor lo que nos ayudará a entender los resultados experimentales.

1 Introducción al sensor LIDAR del Turtlebot

Aunque existen sensores láser de tiempo de vuelo, la gran precisión temporal que requiere el hardware asociado para medir el intervalo de tiempo que la luz tarda en viajar los hace bastante costosos. Los sensores LIDAR de los robot Turtlebot, aunque utilizan luz de laser, no se basan en el principio de tiempo de vuelo sino en el de triangulación (similar al funcionamiento de la cámara MS Kinect pero en 1D), lo que los hace bastante más asequibles.

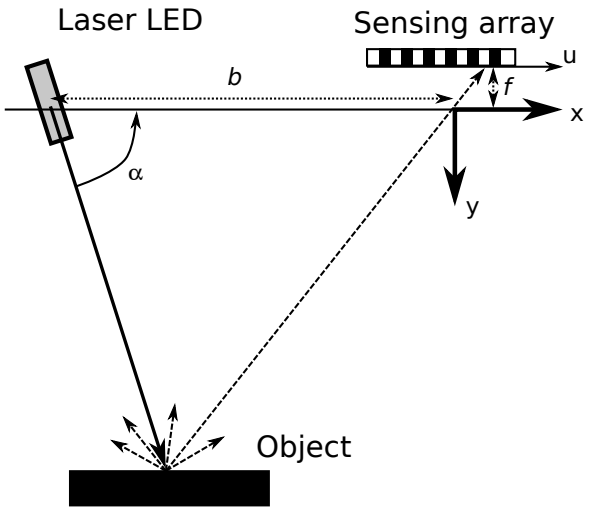

Figure 1: Principio de triangulacón

La figura 1 representa el funcionamiento de un sensor de triangulación. Un emisor de luz láser emite un rayo que incide sobre un objeto (flecha sólida en la figura), la luz es reflejada de forma difusa (flechas discontínuas) y llega a un *array* unidimensional de sensores que detecta la posición (i.e. pixel unidimensional) de la luz reflejada u. Normalmente el emisor y/o el receptor tienen lentes para focalizar la luz emitida y recibida. El emisor y el centro óptico de la lente del receptor están a una distancia b del otro (*baseline*), y la lente del

receptor tiene una distancia focal f. Aunque hay distintas configuraciones para el emisor y el receptor, la figura muestra el caso en que el eje óptico de la cámara del receptor es perpendicular a la línea que une el emisor y el centro óptico del receptor, mientras que la dirección de emisión forma un ángulo α con dicha línea. Se puede comprobar que para esta configuración la distancia medida al objeto (a lo largo del eje óptico del sensor) es:

$$y = \frac{bf}{f \cot \alpha + u} \tag{1}$$

donde u es el índice del sensor en el *array*. Suponiendo que el error en u sigue una distribución Gaussiana[1] $u \sim \mathcal{N}(0, \sigma_u^2)$, la distancia medida y seguirá aproximadamente una Gaussiana[2] $y \sim \mathcal{N}(\hat{z}, \left[\frac{dz}{du}\right]^2 \sigma_u^2)$, donde $\frac{dy}{du}$ puede calcularse usando la ecuación (1), lo que resulta en:

$$\frac{dy}{du} = -\frac{y^2}{bf} \tag{2}$$

y por tanto la varianza de y será $\sigma_y^2 = \frac{y^4}{b^2 f^2} \sigma_u^2$. Esta ecuación nos da información interesante sobre las lecturas del sensor y sobre como diseñar un sensor como este:

- La desviación estándar/varianza de y disminuye al aumentar b, i.e. interesa que el *baseline* sea lo más grande posible, pero por otro lado eso aumenta el tamaño del sensor.
- Si el emisor y el receptor están en el mismo punto $b = 0$, no se puede medir la distancia[3] (cf. ecuación (1)) y la varianza es infinita.
- **La incertidumbre en la distancia medida aumenta con la distancia medida**. Es más, la desviación estándar aumenta con la distancia al cuadrado, i.e. la varianza crece con la distancia a la cuarta potencia de la distancia medida. Esto significa que medidas más cercanas son más repetibles (¿fiables?).

2 Estimación de la desviación estándard del LIDAR

En esta primera parte de la práctica vamos a estimar experimentalmente la desviación estándar del LIDAR, ya que aunque la ecuación (2) nos permite calcularla a partir de los parámetros del sensor, siempre hay imprecisiones en el proceso de fabricación. Llamaremos $\mathbf{r} = [r_1, r_2, \cdots, r_d]$ a un vector de medidas proporcionado por el LIDAR, que está configurado para proporcionar 1440 lecturas de distancia en los 360° alrededor del robot, i.e. $d = 1440$. Como el sensor envía periodicamente lecturas $\mathbf{r}$, denotaremos $\mathbf{r}(k)$ al barrido obtenido en el instante k, i.e. $\mathbf{r}(k) = [r_1(k), r_2(k), \cdots, r_d(k)]$.

Un método sencillo para poder estimar la varianza es poner el robot inmóvil a cierta distancia de un objeto (p.e. una pared) y guardar varias lecturas $k = 1, 2, \cdots, N$ en una dirección i (p.e. la dirección perpendicular a la pared) para calcular su desviación estándar,

[1] Esto no es cierto ya que u toma valores discretos y la Gaussiana se usa en valores contínuos, pero es una suposición típica.

[2] Esto es más falso que un euro de madera ya que la relación entre y y u no es lineal.

[3] No hay visión en profundidad con un solo ojo/cámara.

que sería:

$$\bar{r}_i = \frac{1}{N}\sum_{k=1}^{N} r_i(k) \tag{3}$$

$$\sigma_r^2 = \frac{1}{N-1}\sum_{k=1}^{N}(r_i(k) - \bar{r}_i)^2 \tag{4}$$

Como en nuestro caso σ_r depende de la distancia habría que repetir el proceso para varias distancias lo que daría pares $(\bar{r}_i, \sigma_z(\bar{r}_i))$, i.e. tendriamos distintos valores de σ_r para distintas distancias promedio $\bar{r}_i$.

Una forma más rápida (y más imprecisa) sería situar al robot (estacionario) en un entorno estático y realizar un cierto número de medidas N. Puesto que nada se mueve en el entorno las distancias en cada dirección i serán constantes y diferentes (a no ser que el robot esté en el centro de un entorno circular), así, a través de una sola recogida de datos podemos obtener $d = 1440$ valores de σ_r para distintos valores de $\bar{r}$. El directorio `scan-files` contiene varios ficheros `scan-0XX.dat` en formato texto plano con secuencias de lecturas del LIDAR $\mathbf{r}(k)$ en diferentes puntos del campus (además del script `pyhton` usado para generarlos). Usaremos estos ficheros para calcular la desviación estándar de las lecturas del LIDAR del robot[4]. Cada línea de un fichero contiene una lectura completa en un instante de tiempo $\mathbf{r}(k)$, pero las dos primeras columnas de cada línea son el tiempo de lectura en segundos (relativo a la primera lectura) y el número de distancias proporcionadas por el barrido ($d = 1440$), es decir el fichero contiene:

$$\begin{array}{cccccc} t_1 & 1440 & r_1(1) & r_2(1) & \cdots & r_{1440}(1) \\ t_2 & 1440 & r_1(2) & r_2(2) & \cdots & r_{1440}(2) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{array}$$

El script `analyse-lidar.py` contiene una función `read_scan_file()` que importa todos los barridos del LIDAR como un *array* de `numpy` (descartando las dos primeras columnas). En el *array* devuelto por la función (lo llamaremos D) cada fila i corresponde con una lectura $\mathbf{r}(i)$ y cada columna j con una dirección del LIDAR r_j, que en el caso de que el sensor no tuviera ruido debería ser siempre la misma distancia.

Para estimar la desviación estándar como función de la distancia solo hay que calcular la media de cada columna de la matriz D y su desviación estándar correspondiente. Sin embargo hay una serie de problemas prácticos:

- Algunas de las lecturas del sensor, entradas en la matriz D contienen `inf`, que ocurre cuando la luz reflejada no ha llega al *array* sensorial (p.e. espejos, superficies pulidas, cristales...). Estas lecturas pueden identificarse usando la función `isfinite()` de `numpy`.

- El rango del sensor es de 0.5 a 25 metros, por lo que es recomendable considerar solo lecturas con medias que entren en ese rango (incluso el valor máximo puede reducirse a $r_M < 25$ porque sabemos que las lecturas lejanas son más imprecisas).

- En ocasiones las distancias proporcionadas por el sensor son siempre la misma, lo cual genera una varianza nula. Aunque esto parece una situación ideal, no ayuda a la estimación de σ_r y, por tanto, esos datos pueden eliminarse[5].

[4] Los ficheros `scan-01X.dat` han sido generados con un robot y los `scan-00X.dat` con otro, i.e. los LIDAR son distintos.

[5] También pueden eliminarse varianzas que sean demasiado grandes debido a distintos errores del LIDAR, p.e. lecturas en direcciones en la que hay una discontinuidad.

Implementa la función `get_stds(D)` que calcule las medias y las desviaciones estándar de las columnas de la matriz D y las devuelva teniendo en cuenta los problemas prácticos anteriores. **Q1. Representa gráficamente los valores de desviaciones estándar frente a las medias para distintos ficheros y comprueba si se cumple la relación dada por la ecuación obtenida para σ_r. ¿Siguen los puntos obtenidos una relación cuadrática? ¿Hay algún rango en que la relación se cumpla 'mejor'?**

En este momento tenemos un conjunto de valores de medias y desviaciones estándar (posiblemente en un rango menor que $[0.5, 25]$), con lo que podemos obtener la relación funcional entre las dos variables (media, variable independiente, y desviación estándar, variable dependiente). Para ello podemos implementar una simple regresión lineal, ya que el modelo teórico nos dice que $\sigma_r = a\bar{r}^2$, donde (según el modelo) $a = \frac{\sigma_u}{bf}$. Dadas las muestras $(\bar{r}_i, \sigma_i)$ para $i = 1, 2, \cdots, d$ ($d \approx 1440$) queremos encontrar a que minimize el error

$$E(a) = \frac{1}{2}\sum_{i=1}^{d}(\sigma_i - a\bar{r}_i^2)^2 \tag{5}$$

Imponiendo la condición $\frac{dE}{da} = 0$ y resolviendo para a se obtiene que:

$$a = \frac{\sum_{i=1}^{d}\sigma_i\bar{r}_i^2}{\sum_{i=1}^{d}\bar{r}_i^4} \tag{6}$$

Implementa la función `linear_regression()` que obtiene el valor de a a partir de las medias y las desviaciones estándar obtenidas. El programa python dibujará los puntos usados y la función obtenida de la regresión con el valor de a calculado. **Q2. ¿Se ajusta la función de regresión a los puntos? ¿Qué valor de a obtienes? Incluye al menos una gráfica generada a partir de uno de los ficheros de datos y comentala. Q3. ¿Cuál es la desviación estándar del sensor al medir 20m según tu modelo? ¿Cuál es la varianza si el sensor mide 15m? Explica cómo las calculas.**

3 Posiciones Cartesianas con incertidumbre

Ahora que tenemos una estimación de la varianza de las lecturas podemos calcular un intervalo de confianza para las distancias medidas por el sensor. Dada una distancia medida r, la distancia real[6] estará entre $r \pm 3\sigma$ con un 99.7% de probabilidad. El código `python` representa las lecturas de distancia (en azul) con intervalo de confianza $\pm 3\sigma$ (en rojo) para una serie de barridos del LIDAR. Lo que se muestra es directamente el vector $\mathbf{r}$, como una función del índice del vector, i.e. r_i frente a i. Las lecturas de por sí son difíciles de relacionar con el entorno del robot, por ejemplo, las paredes rectas del entorno aparecen como una especie de función secante. Resulta más facil interpretar el entorno convirtiendo las distancias a coordenadas Cartesianas relativas al sistema de referencia del robot (en realidad del sensor LIDAR).

Para convertir las distancias r_i a puntos (x, y) relativos al robot es necesario conocer el ángulo (respecto al eje x) al que corresponde dicha lectura del barrido. Para ver cómo se relaciona el índice i con el ángulo debemos saber cómo realiza el barrido el sensor y a qué intervalos se muestrean las distancias. Según la documentación del LIDAR, el barrido se realiza en sentido horario, y según la información proporcionada por `rostopic` (ver fichero

[6] Suponiendo que el sensor está calibrado y no hay errores sistemáticos.

`header.txt`) el incremento entre dos lecturas consecutivas es de $\Delta\theta = 0.00435422640294$ radianes[7]. Teniendo en cuenta que la primera lectura corresponde al frente del sensor (robot), i.e. 0°, el incremento angular y la dirección de barrido, se puede calcular el ángulo θ_i correspondiente a la lectura r_i. Con esta información se pueden obtener las coordenadas Cartesianas relativas al sistema de referencia del robot como:

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \end{aligned} \tag{7}$$

Por otra parte se puede calcular/approximar la matriz de covarianza de la lectura en coordenadas Polares usando el resultado de la regresión del apartado anterior (i.e. σ_r) y suponiedo que la desviación estándar en la variable θ (i.e. σ_θ) es, por ejemplo, un tercio del incremento entre dos lecturas, de forma que la lectura siguiente estará en el límite del 99.7% de la variable angular θ. Esta suposición tiene bastante sentido porque después de un incremento angular de $3\sigma_\theta$ tendremos una nueva lectura de distancia. Típicamente se supone que las variables de distancia r y ángulo θ son independientes, y a partir de la covarianza en polares Σ_P se puede obtener la covarianza en coordenadas Cartesianas $\Sigma_C = J\Sigma_P J^T$[8], donde J es la matriz Jacobiana de la transformación de coordenadas evaluada en la média.

Con todo esto en mente se pueden implementar las funciones que faltan para representar las lecturas del robot en coordenadas Cartesianas de su sistema de referencia. La función a implementar `polar2cart()` toma como argumentos el vector correspondiente a una lectura (r, θ) y su matriz de covarianza Σ_P y debe retornar un vector en coordenadas Cartesianas y su correspondiente matriz de covarianza Σ_C. Por otra parte la función a implementar `plot_scan_cart()` toma como entrada el vector de barrido completo $\mathbf{r} = [r_1, r_2, \cdots, r_d]$ y debe representar las lecturas con su incertidumbre asociada en coordenadas Cartesianas. Esta función usará a su vez la función `plot_error_ellipse()` que dibuja la elipse de incertidumbre con un intervalo de confianza de 3σ alrededor del punto obtenido en coordenadas Cartesianas.

Completa las funciones necesarias para representar las lecturas en coordenadas Cartesianas, e incluye alguna imágen en el informe. Haz zoom en alguno de los puntos del entorno para poder ver la incertidumbre de forma más clara. **Q4. ¿Hay alguna elipse de incertidumbre que en realidad sea un círculo? ¿Podría pasar? Justifica tu respuesta. Q5. ¿Hay diferencia entre las elipses de incertidumbre cerca y lejos del robot? ¿Cual y por qué? Q6. ¿Por qué las elipses típicamente 'apuntan' hacia el origen?**

[7] Hay una pequeña discrepancia entre el incremento teórico $\frac{2\pi}{1440}$ y el dado por ROS.

[8] Esto es una approximación porque el cambio de coordenadas no es lineal.