Proyecto de Investigación de Excelencia de la Junta de Andalucía P10-TEP-6101-R

"Navegación autónoma de un robot móvil 4x4 en entornos naturales mediante GPS diferencial y telémetro láser tridimensional"

Navegación autónoma de Andábata

Manuel Zafra Granados
Jorge L. Martínez Rodríguez

Departamento de Ingeniería de Sistemas y Automática
Universidad de Málaga

Marzo de 2016

Índice

1. Objetivos

Para la navegación de vehículos autónomos es necesario que estos sean capaces de decidir qué camino tomar atendiendo al entorno que los rodea. Además, dichos robots, deben ser capaces de alcanzar determinados lugares esquivando los obstáculos que puedan poner en peligro su integridad.

El objetivo principal de este informe es describir la navegación autónoma de Andábata haciendo uso de mapas de elevación borrosos y los datos de los sensores de su teléfono móvil. Para poder alcanzar este objetivo se plantean una serie de objetivos segundarios como son la remodelación del sistema de control del movimiento del vehículo, el cálculo de la guiñada del mismo mediante el procesamiento de las lecturas de los magnetómetros del teléfono, el cálculo de la dirección del siguiente punto objetivo procesando los datos del GPS y el procesado de los mapas borrosos para elegir una dirección de avance esquivando obstáculos.

2. Punto de partida

Actualmente Andábata no es capaz de realizar ningún tipo de navegación autónoma. Se puede teleoperar desde varios tipos de controles remotos, seleccionando uno en cada momento. El control del movimiento del vehículo se hace indicando las velocidades de los pares de ruedas derechas e izquierdas. Las medidas de los magnetómetros y del GPS son accesibles para el robot, pero no se realiza ningún cálculo con ellas [1]. El robot también es capaz de crear mapas de elevación borrosos del terreno que lo rodea mientras se mueve, para ello utiliza la nube de puntos capturada con el telémetro láser 3D [2].

3. Implementación en ROS

3.1. Control del movimiento del robot y teleoperación

Como se ha comentado en el apartado 2, la teleoperación se realiza eligiendo la velocidad de las ruedas derechas e izquierdas. Para la navegación autónoma es necesario que el vehículo se controle con comandos de velocidad lineal y angular. Por ello, se ha realizado una remodelación completa del sistema que controla el movimiento de las ruedas y la teleoperación.

Se requiere que, en algún punto del control de las ruedas, se puedan incorporar órdenes en forma de velocidades lineales y angulares del vehículo sin perder el control por mandos remotos. Además, se ha de incorporar un tipo de mando nuevo no teleoperado por el que fluirán las órdenes de velocidades provenientes del sistema de navegación.

Lo primero que se ha hecho es separar la parte del control de las ruedas de la parte de teleoperación en dos paquetes: *andábata_teleoperación*, que contiene los nodos que leen los *joysticks* y publica las ordenes de velocidades; y *andábata_move*, que contiene lo necesario para el cálculo de las velocidades de las ruedas de cada lado del robot y de la odometría.

3.1.1. Teleoperación

En la Figura 1a) se muestra el esquema de funcionamiento del paquete *andabata_teleoperation*. Los cambios más significativas respecto a la versión anterior son que los nodos encargados de transmitir las ordenes del teléfono móvil y la tableta

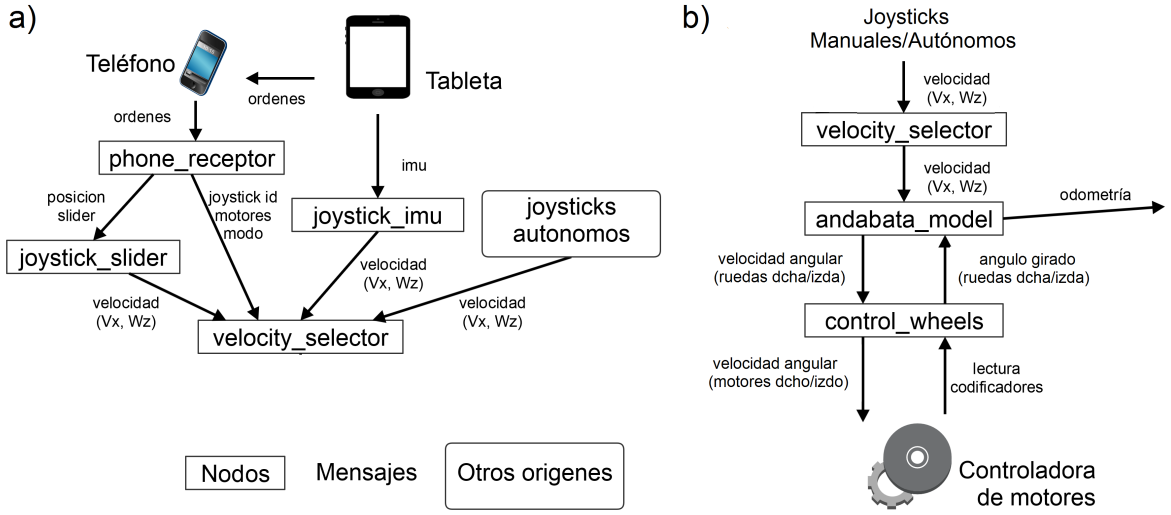

Figura 1: Esquema de nodos del sistema de teleoperación a), y de control del movimiento del robot b).

envían velocidades lineales y angulares, en vez de velocidades de las ruedas de los lados izquierdo y derecho.

3.1.2. Control del movimiento del robot

En la Figura 1b) se puede ver un esquema de los nodos que componen el paquete *andabata_move*. Concretamente, los nodos que contiene son los siguientes:

- Nodo *velocity_selector*, que proviene de los nodos *joystick_multiplexor* y *motors_control* [1], al que se le ha añadido, en la parte que corresponde al modo automático, las instrucciones para manejar las órdenes del sistema de navegación y de calibrado del compás magnético.
- Se ha interpuesto entre este nodo y el de control de las ruedas un nodo, *andabata_model* que contiene el modelo cinemático aproximado del robot (ver Apartado 3.1.3), y que convierte las ordenes de velocidad lineal y angular en ordenes de velocidad angular de las ruedas izquierda y derecha (modelo cinemático inverso). También transforma los giros de las ruedas en desplazamiento y giro del robot (modelo cinemático directo), publicando el correspondiente mensaje de odometría.

  Además, en el caso de que alguna de las ruedas supere su velocidad máxima V_{max}, adecúa la velocidad de las ruedas para mantener la curvatura de la

trayectoria. Para ello, se divide la velocidad angular ω y lineal v deseadas por un mismo divisor e. El divisor consiste en el tanto por uno de la velocidad maxima que se ha superado:

$$e = \frac{|v| + x_{CIR} \ |\omega|}{V_{max}} \tag{1}$$

- Por último, *control_wheels* es el nodo que se encarga de la comunicación con las controladoras de los motores. Recibe velocidades angulares de las ruedas que transforma en la consigna correspondiente a las controladoras. Además, lee los codificadores y publica los radianes girados por las ruedas derechas e izquierdas.

3.1.3. Modelo cinemático aproximado

El modelo cinemático 2D apróximado de Andábata se define mediante dos parámetros [3]. Debido a los cambios en la estructura de Andábata, cambio de reductora y motores, ha sido necesario un nuevo cálculo de los mismos. En las Tablas 1 y 2 aparecen los experimentos y los resultados obtenidos.

Velocidad		**Pulsos**		**Distancia lineal (mm)**		$\boldsymbol{\phi}$		$\boldsymbol{x_{CIR}}$
Der	**Izq**	**Der**	**Izq**	$\int \boldsymbol{V_d dt}$	$\int \boldsymbol{V_i dt}$	**º**	**Rad**	**(mm)**
-400	400	-340337	341186	-2889,74	2896,95	-355	-6,2055	466,25
400	-400	337618	-338858	2866,65	-2877,18	374	6,5414	439,03
-800	800	-328814	329224	-2791,90	2795,38	-360	-6,2832	444,62
800	-800	332681	-332858	2824,73	-2826,23	362	6,3260	446,64
								449,14

Tabla 1: Resultados del experimento realizado para determinar x_{CIR} para interiores.

Velocidad		**Pulsos**		**Distancia lineal (mm)**		**d**	$\boldsymbol{\mu}$
Der	**Izq**	**Der**	**Izq**	$\int \boldsymbol{V_d dt}$	$\int \boldsymbol{V_i dt}$	**(mm)**	
400	400	325461	325700	2763,43	2765,46	2620	0,9477
-400	-400	-287533	-287769	-2441,39	-2443,39	-2280	0,9335
800	800	294461	294919	2500,21	2504,10	2330	0,9312
-800	-800	-498069	-497924	-4229,01	-4227,78	-4000	0,9460
							0,9396

Tabla 2: Resultados del experimento realizado para determinar μ para interiores.

3.2. Creación de mapas multiproceso

Como se describió en [2], el nodo *map* del paquete *andábata_map* crea y publica un mapa de elevación borroso por cada nube de puntos que recibe. El tiempo de creación del mapa depende del número de puntos que contenga la nube de puntos, pero, en general, toma más tiempo del deseable para la navegación autónoma. Actualmente, este proceso se divide en dos partes: cálculo de las alturas de las celdas y de la máscara de confianza, que se ejecutan en dos núcleos de los 8 con los que cuenta el procesador de Andábata.

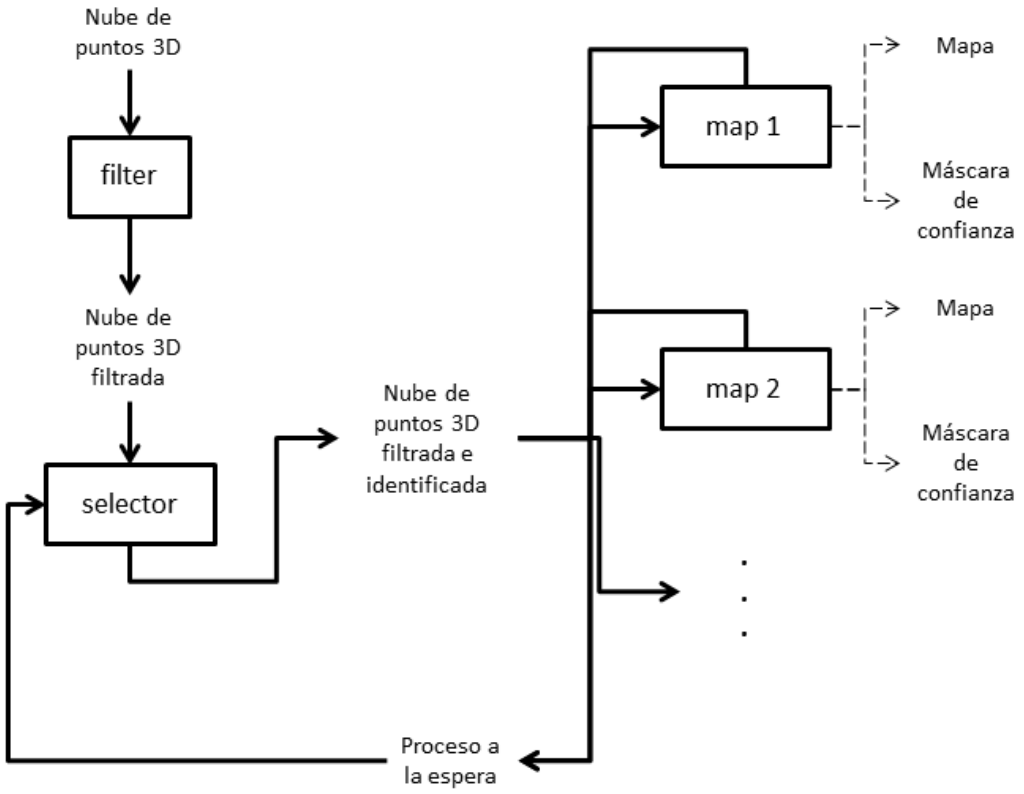

Figura 2: Diagrama del flujo de datos.

Se ha decidido lanzar dos veces el proceso y crear un gestor, llamado *selector*, que decida qué nodo procesa cada nube de puntos. El esquema de la Figura 2 muestra el flujo de datos a través de los nodos. El gestor se encarga de que la frecuencia de publicación de mapas sea lo más constante posible alternando el envío de las nubes de puntos a uno u otro nodo *map*, y haciendo que algunas nubes de puntos no sean procesadas cuando hay una nube de puntos 3D disponible más reciente.

Se ha decidido hacerlo de esta forma, ya que, si solo se aumenta el número de procesos, la frecuencia de salida de mapas aumentaría, pero el retraso entre el momento de captura de una nube de puntos y la disponibilidad de su correspondiente mapa no disminuye. Con este procedimiento se consigue disminuir el retardo y aumentar la frecuencia de salida de mapas. La configuración adoptada es de 2 nodos *map*, que hace que, en total, se utilicen 4 nucleos del procesador para la creación de mapas borrosos. La Figura 3 muestra el uso interno del procesador cuando Andábata navega de forma autónoma.

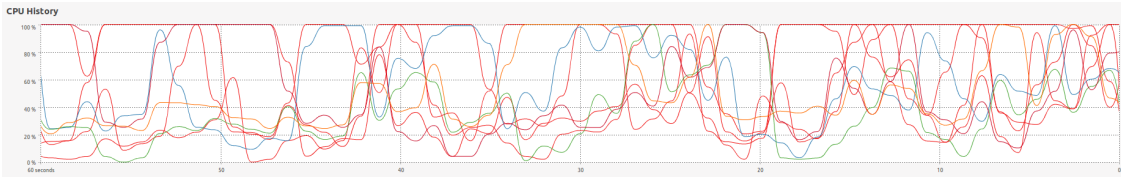

Figura 3: Uso de los núcleos del procesador de Andábata.

3.3. Navegación local punto a punto

Para la navegación local punto a punto se ha seguirdo el diagrama de flujo que se muestra en la Figura 4. Si en algún momento el robot se inclina más de unos límites predefinidos (20º en *roll* o en *pitch*) se envía la orden de parar.

La trayectoria que seguirá el robot entre punto y punto depende de lo lejos que esté el punto objetivo y del error de orientación(ver Figura 5) del robot respecto del mismo [4]. Concretamente la velocidad angular ω tiene un valor de:

$$\omega = \frac{error\_orientacion}{G} \tag{2}$$

donde la ganancia G vale:

$$G = \frac{constante\_proporcional}{distancia\_objetivo} \tag{3}$$

donde *constante_proporcional* es un parametro que determina la amplitud de la curvatura en la trayectoria, se ha usado una constante de valor 1.

Como se puede observar en la Figura 4, los objetivos se pueden cancelar o cambiar. Cuando el robot está lo suficientemente cerca de un punto objetivo (20 m) pasa al siguiente, y cuando llega al punto final se para. Como se recibe un vector de puntos es posible seguir una trayectoria determinada.

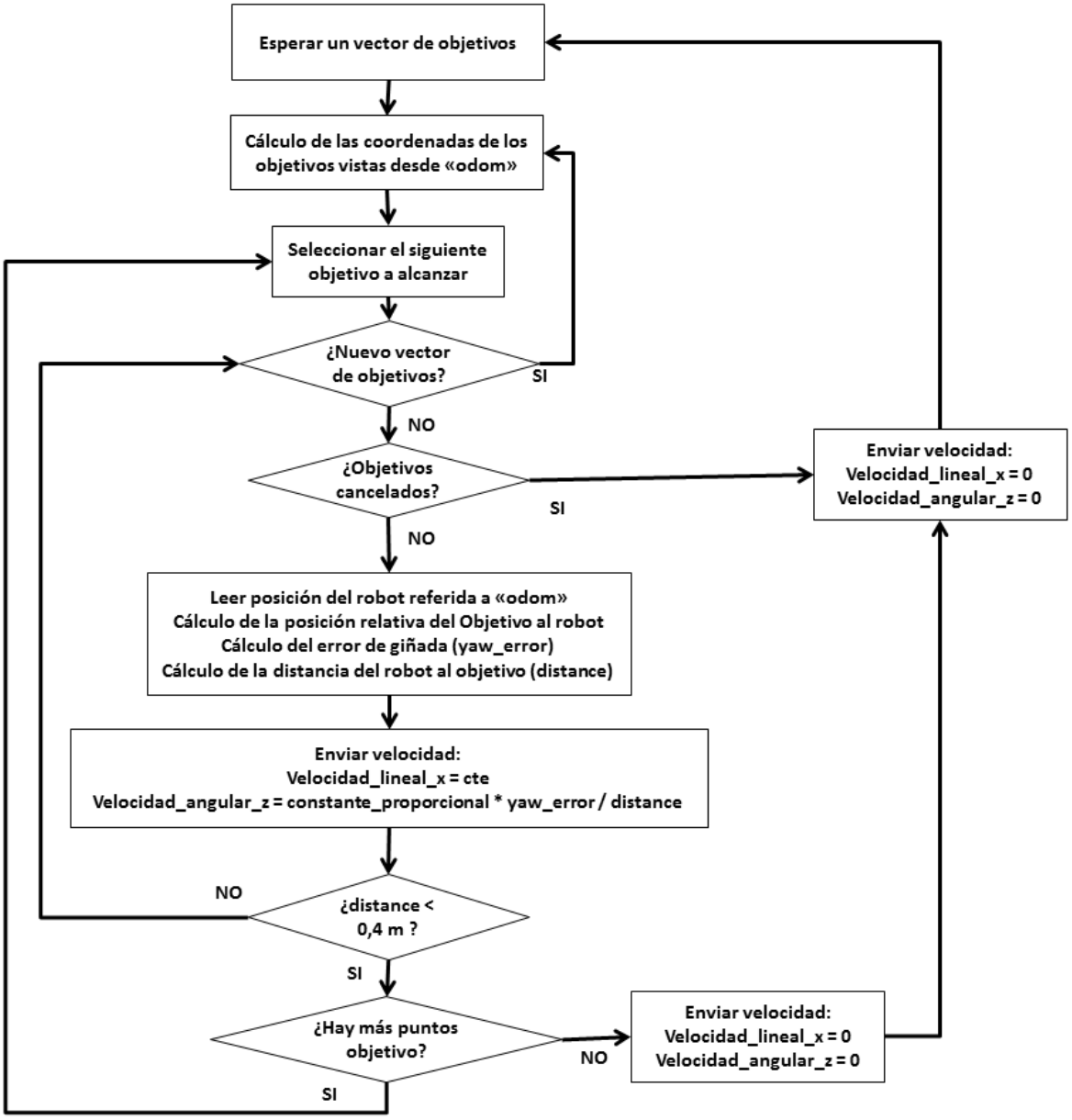

Figura 4: Diagrama de flujo principal.

Se ha utilizado *actionlib*, que es un paquete ROS que implementa un método eficiente de plantear procesos que tomarán un tiempo largo y que deben poder ser interrumpidos o cancelados [5]. Este paquete se comentará a continuación.

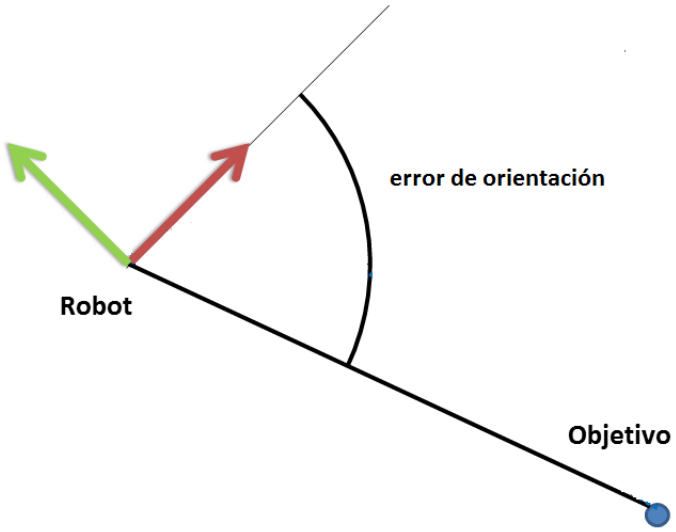

Figura 5: Error de orientación respecto a un punto objetivo.

3.3.1. Paquete *actionlib*

En cualquier gran sistema basado en ROS, hay casos en que puede ser necesario enviar una petición a un nodo para realizar alguna tarea, y también recibir una respuesta a la solicitud. Ésto se puede lograr a través de los servicios de ROS, si el servicio tarda mucho tiempo en ejecutarse, el usuario debe tener la posibilidad de cancelar la solicitud durante la ejecución o conseguir información periódica acerca del progreso de la solicitud. El paquete *actionlib* proporciona herramientas para crear servidores que ejecutan objetivos de larga duración que se pueden interrumpir. También proporciona una interfaz de cliente para enviar peticiones al servidor.

El *ActionClient* y *ActionServer* se comunican a través de un "Protocolo de Acciones de ROS", que se construye sobre los mensajes de ROS. El cliente y el servidor proporcionan una API simple para solicitar objetivos (en el lado del cliente) o para ejecutar los objetivos (en el lado del servidor) a través de llamadas a funciones.

Para que se comuniquen el cliente y el servidor es necesario definir una serie de mensajes, lo cual se hace al especificar la acción. La especificación de la acción se realiza utilizando un archivo ".action". El archivo ".action" tiene la definición del objetivo, del resultado y de la realimentación. Estos archivos se colocan en el directorio "./action" de un paquete y son muy similares a los archivos ".srv" de un servicio.

El objetivo es enviado desde el cliente al servidor donde se especifica el trabajo concreto que este debe realizar. El mensaje de realimentación contiene información

sobre el estado actual de la tarea. El mensaje de resultado contiene la información obtenida al realizar el cálculo o proceso.

En el caso específico del que se ocupa este informe, el nodo cliente envía un vector de puntos objetivo a los que debe dirigirse. Estos puntos pueden estár referenciados a cualquier eje de coordenadas existente en el entorno ROS. El nodo servidor ejecuta el diagrama de la Figura 4, envía mensajes de realimentación con la distancia del robot al punto objetivo en cada momento y el mensaje de resultado es el tiempo que ha tardado en alcanzar el objetivo.

3.4. Desacoplo de las medidas de los sensores globales

Para poder fijar un objetivo mediante su latitud y su longitud y realizar una navegación local sin que intervengan directamente los datos obtenidos del sensor GPS es necesario desacoplar las medidas del GPS del cálculo de la trayectoria del robot y realizar la navegación de forma local, ya que las medidas del GPS del teléfono móvil tienen un error de unos 10 m, por lo que no se puede hacer un control preciso de la posición del robot.

De esta tarea se encarga un nodo llamado *obj_fixer* perteneciente al paquete *andábata_navigation*. A este nodo llegan los datos de las medidas de los sensores GPS y magnetómetros del teléfono móvil, además de una matriz que contiene las coordenadas de latitud y longitud de los objetivos a alcanzar. La función es la de, sabiendo la posición global del objetivo y del robot, y la guiñada de este último, calcular la posición del objetivo referido a los ejes de coordenadas del origen de la navegación local (*odom*). De esta forma, cada vez que recibe una medida del sensor GPS, publica un punto con las coordenadas del objetivo referenciados a *odom*.

Para poder realizar los cálculos es necesario minimizar los errores en las medidas de los sensores que intervienen (GPS y magnetómetros). Para las medidas del GPS se consideran como válidas sólo las medidas tomadas con suficientes satélites fijados. En el caso de los magnetómetros del teléfono móvil se decidió realizar una calibración inicial.

3.4.1. Calibración del compás magnético

Los magnetómetros del teléfono móvil son muy sensibles al entorno pudiendo variar su medida ampliamente, siendo el propio robot, una gran fuente de interferencias. Se ha decidido implementar un sistema que calibre los sensores magnéticos cada vez que el robot vaya a realizar uso de ellos, es decir, cuando entra en modo automático por primera vez después de arrancarse.

Se realiza la calibración de los magnetómetros dirigidos a los ejes X e Y haciendo girar el robot sobre su eje Z por lo menos una vuelta completa y ajustando las medidas de dichos ejes a una circunferencia centrada en el origen (ver Figura 6). Este ajuste es guardado mientras que el nodo que se encarga de esta tarea, llamado *compass*, se encuentre activo.

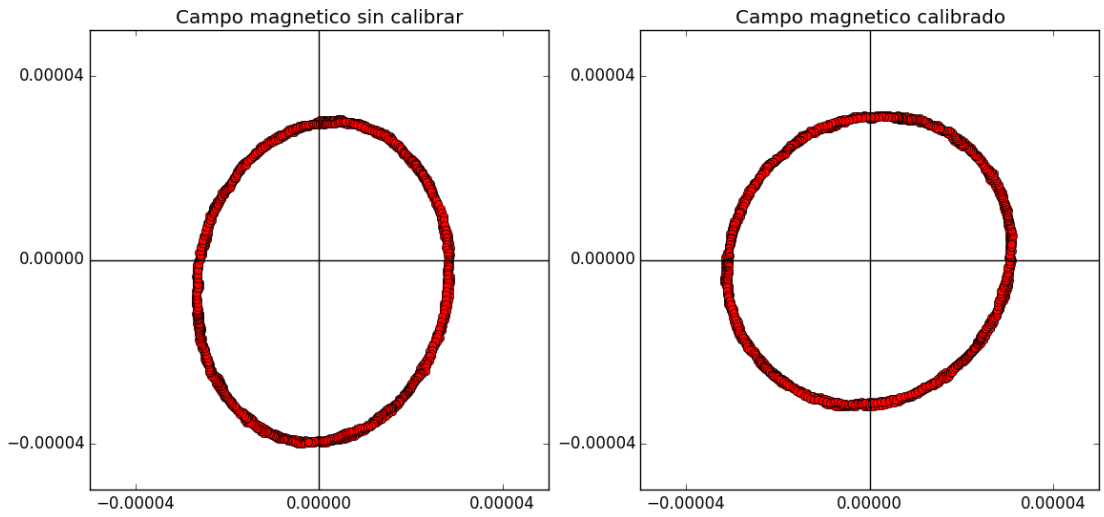

Figura 6: Ejemplo de la medida del campo magnético al dar una vuelta sobre el eje Z antes y despues de la calibración.

Una vez que las medidas de los magnetómetros están corregidas se puede obtener la guiñada directamente mediante la fórmula:

$$guiñada = arctan\left(\frac{campo\_magnetico\_en\_el\_eje\_X}{campo\_magnetico\_en\_el\_eje\_Y}\right) \tag{4}$$

Esta fórmula es válida solo para un movimiento en 2D. Lo más correcto sería utilizar un modelo 3D, pero esto requiere de la calibración del magnetómetro del eje Z, lo cual en el caso del robot Andábata, es impracticable.

3.5. Elección de la dirección

El nodo *dir_selector* del paquete *andábata_navigation* recibe el mapa borroso creado y el punto objetivo referenciada a los ejes de coordenadas locales (*odom*, donde se arrancó el robot) y decide qué dirección debe tomar el robot. La posición del punto objetivo se guarda cada vez que se recibe un mensaje de este tipo (de forma asíncrona con la recepción de mapas), mientras que al recibir un mensaje de mapa borroso se realizan los cálculos necesarios para elegir una dirección.

La dirección que minimice la expresión 5 será la que seguirá Andábata.

$$\frac{(error\_direccion + a\ cambio\_direccion)}{distancia\_recorrible} \tag{5}$$

donde *error_direccion* es el valor absoluto de la diferencia entre la dirección del robot y la dirección al objetivo, *cambio_direccion* es el valor absoluto de la diferencia entre la dirección y la última dirección escogida, *distancia_recorrible* es la distancia que se puede recorrer en dicha dirección y el parámetro a se debe escoger para dar un peso adecuado al cambio de dirección.

Para el cálculo de la distancia recorrible por el robot en una determinada dirección se evaluan puntos cada 0.1 m partiendo de la posición actual del robot. En cuanto el punto evaluado tenga poca confianza en la máscara de confianza (menos de 0.75), tenga un gradiente muy alto (mayor de 0.5) o se salga del mapa borroso, se considera alcanzada la máxima distancia.

El gradiente del mapa borroso en un punto (x, y) se calcula con las fórmulas:

$$\frac{\partial H}{\partial x} = \sum_{\forall i,j} \left(\mu_{\alpha j}\ \frac{\partial \mu_\tau}{\partial x} i + \mu_{\tau i}\ \frac{\partial \mu_\alpha}{\partial x} j \right)\ z_{ij} \tag{6}$$

$$\frac{\partial H}{\partial y} = \sum_{\forall i,j} \left(\mu_{\alpha j}\ \frac{\partial \mu_\tau}{\partial y} i + \mu_{\tau i}\ \frac{\partial \mu_\alpha}{\partial y} j \right)\ z_{ij} \tag{7}$$

$$norma\_gradiente(x, y) = \sqrt{\left(\frac{\partial H}{\partial x}\right)^2 + \left(\frac{\partial H}{\partial y}\right)^2} \tag{8}$$

donde μ_α y μ_τ son las pertenencias del punto al rango angular y de distancia respectivamente; y z_{ij} es la altura de la celda (i, j) del mapa borroso.

Las derivadas parciales de la pertenencia de un punto (x, y) a una celda se pueden calcular como:

$$\frac{\partial \mu_\alpha}{\partial x} = \begin{cases} \frac{y}{\tau^2}\ \frac{1}{inc\_ang} & \text{si } dif\_ang > 0 \\ \frac{-y}{\tau^2}\ \frac{1}{inc\_ang} & \text{si } dif\_ang < 0 \end{cases} \tag{9}$$

$$\frac{\partial \mu_\alpha}{\partial y} = \begin{cases} \frac{-x}{\tau^2}\ \frac{1}{inc\_ang} & \text{si } dif\_ang > 0 \\ \frac{x}{\tau^2}\ \frac{1}{inc\_ang} & \text{si } dif\_ang < 0 \end{cases} \tag{10}$$

$$\frac{\partial \mu_\tau}{\partial x} = \begin{cases} 0 & \text{si } \tau < \tau c_1 \text{ ó } \tau > \tau c_{conj\_dist} \\ \frac{x}{\tau} \frac{1}{\tau c_i - \tau c_{i-1}} & \text{si } \tau > \tau c_1 \text{ y } \tau < \tau c_{conj\_dist} \text{ y } \tau c_i - \tau > 0 \\ \frac{-x}{\tau} \frac{1}{\tau c_{i+1} - \tau c_i} & \text{si } \tau > \tau c_1 \text{ y } \tau < \tau c_{conj\_dist} \text{ y } \tau c_i - d < 0 \end{cases} \quad (11)$$

$$\frac{\partial \mu_\tau}{\partial y} = \begin{cases} 0 & \text{si } \tau < \tau c_1 \text{ ó } \tau > \tau c_{conj\_dist} \\ \frac{y}{\tau} \frac{1}{\tau c_i - \tau c_{i-1}} & \text{si } \tau > \tau c_1 \text{ y } \tau < \tau c_{conj\_dist} \text{ y } \tau c_i - \tau > 0 \\ \frac{-y}{\tau} \frac{1}{\tau c_{i+1} - \tau c_i} & \text{si } \tau > \tau c_1 \text{ y } \tau < \tau c_{conj\_dist} \text{ y } \tau c_i - \tau < 0 \end{cases} \quad (12)$$

donde τ es la distancia de la proyección del punto al centro del mapa, *inc_ang* es la diferencia de angulo entre el centro de una celda cualquiera y la consecutiva, dif_ang es el valor absoluto de la diferencia de angulo entre el punto y el centro de la celda, τc_k es la distancia del centro del rango de distancias i al centro del mapa, y *conj_dist* es el número de rangos de distancia que tiene el mapa.

Dado que el nodo de navegación autónoma es punto a punto y aquí se selecciona una dirección, al módulo de navegación se le envía un punto en la dirección escogida que está a la máxima distancia recorrible en esta dirección y otro punto es dicha dirección pero a 100 metros, para que mantenga la dirección hasta que se procese otro mapa.

4. Referencias

[1] M. Zafra, "Construcción de Mapas de Exteriores Mediante Octrees para un Robot Móvil equipado con un Telémetro Láser 3D," Proyecto Final de Carrera (E.T.S.I. Industrial), Universidad de Málaga, 2015.

[2] M. Zafra, J. L. Martínez, and A. J. Reina, "Procesado de una Nube de Puntos para la Creación de un Mapa de Elevación Borroso.," Inf. Tec., Universidad de Málaga, Dic. 2015.

[3] M. Zafra, J. L. Martínez, and J. Morales, "Obtención de Barridos 3D Nivelados mientras Andábata se Desplaza," Inf. Tec., Universidad de Málaga, Sep. 2015.

[4] J. Morales, J. L. Martínez, A. Mandow, A. Garcia-Cerezo, and S. Pedraza, "Power Consumption Modeling of Skid-Steer Tracked Mobile Robots on Rigid Terrain," *IEEE Transactions on Robotics*, vol. 25, pag. 1098–1108, Oct. 2009.

[5] E. Marder-Eppstein and V. Pradeep, "actionlib." `https://github.com/ros/actionlib`.