UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster del
MÁSTER UNIVERSITARIO EN I.A. AVANZADA: FUNDAMENTOS, MÉTODOS Y APLICACIONES

NAVEGACIÓN POR CENTRO DE ÁREAS UTILIZANDO CÁMARA DE MAPA DE PROFUNDIDAD

Francisco Antonio Marín García

Codirigido por:

Dr. D. Félix de la Paz López, Dr. D. José Manuel Cuadra Troncoso

Curso: 2020-2021: Convocatoria septiembre

Agradecimientos

Alguien publicó en Twitter que la UNED es la mejor universidad del mundo. No puedo estar más de acuerdo. Como alumno he podido apreciar la calidad humana y profesional de sus profesores, pero también la labor social que están realizando. Hay muchos que por circunstancias del lugar de nacimiento o por escasos recursos económicos familiares no pueden optar por estudios superiores. Luego, cuando son otras las circunstancias ahí está esta universidad para rescatar a los que en su momento no pudieron acceder a esos estudios.

Quiero agradecer a todos y cada uno de los que han sido mis profesores durante mi vida de estudiante en la UNED. A los profesores de este Máster, en particular al Profesor D. Mariano Rincón Zamorano por admitirme en estos estudios, a la Profesora Dña. Ángeles Manjarrés Riesco por su interés por mi admisión y finalmente a los Profesores D. Félix de la Paz López y D. José Manuel Cuadra Troncoso que me han guiado en este trabajo.

A mi esposa Elisa, mi hija Lorena y mi hijo Pablo que me perdonen por mis ausencias debidas a mi carga de trabajo.

Un recuerdo a los que no están, mi abuelo Antonio por sus nociones de mecánica y electricidad y mi tío Antonio por generar mi pasión por la electrónica y las radiocomunicaciones.

Resumen

Partiendo de la Tesis del Doctor Don José Manuel Cuadra Troncoso, MODELADO ADAPTATIVO DEL MEDIO PARA LA NAVEGACIÓN DE ROBOTS AUTÓNOMOS UTILIZANDO ALGORITMOS BASADOS EN EL CENTRO DE ÁREAS, este proyecto trata de desmostrar cómo la utilización de cámaras de mapa de profundidad tiene unas considerables ventajas frente a la medida de rango láser que se utilizó en la referida Tesis. Como se sabe, el láser 2D sólo puede ver objetos en un sólo plano, su plano de barrido. Pues bien, al utilizar cámaras de mapa de profundidad se tiene tantos planos como resolución vertical tenga la cámara. Además, se puede derivar, con esta forma de medir el entorno, la altura de los objetos sobre el suelo dotando al sistema la característica 3D. Como se demostrará, para hacer navegar un robot por debajo de un objeto (un puente, por ejemplo) no es necesario esta característica 3D. Para un "mundo" sobre el suelo no se tiene la necesidad de pasar a un 3D real, es suficiente con lo que se ha denominado acotado-3D. Ni si quiera hará falta calcular la altura. Con descartar toda la información del semiplano superior de visión (menos algunas filas por seguridad, unos centímetros por encima de la cámara) es suficiente. También se detecta objetos sobre el suelo (una piedra, por ejemplo) como se demuestra en el experimento del circuito cerrado

Se han realizado todos los experimentos de la Tesis (ver capítulo 9). Se ha modificado el experimento de circuito cerrado para demostrar lo que se ha dicho más arriba. Se ha introducido dos puentes, uno de ellos de dos niveles, y una "piedra" sobre el suelo.

Las medidas de las cámaras de profundidad sirven de entrada a la librería *libareacenter* que se encarga del proceso de construir los polígonos global y de avance y genera internamente la medida del centro de áreas que se utiliza para la navegación del robot. La salida de librería son las velocidades de las ruedas izquierda y derecha del robot del sistema diferencial.

Abstract

Starting from the Thesis of PhD Don José Manuel Cuadra Troncoso, ADAPTIVE MODELING OF THE MEDIUM FOR THE NAVIGATION OF AUTONOMOUS ROBOTS USING ALGORITHMS BASED IN THE CENTER OF AREAS, this project tries to demonstrate how the use of depth map cameras has considerable advantages compared to laser range measurements that was used in the referred Thesis. As is known, the laser can only see objects in a single plane, its scanning plane. Well, when using depth map cameras we have as many planes as the vertical resolution the camera has. In addition, it is possible to derive, with this way of measuring the environment, the height of the objects above the ground, giving the system the 3D characteristic. As will be demonstrated, this 3D feature is not required to navigate a robot under an object (a bridge, for example). For a "world" on the ground there is no need to go to a real 3D, what has been called acotado-3D, is enough. You won't even need to calculate the height. With discarding all the information of the upper half plane of vision (minus some rows for security, a few centimeters above the camera) it is enough. Objects on the ground (a stone, for example) are also detected as demonstrated in the closed-loop experiment

All the experiments of the Thesis have been carried out (see chapter 9). The closed circuit experiment has been modified to demonstrate what has been said above. Two bridges have been introduced, one of them of two levels, and a "stone" on the ground.

The measurements of the depth cameras serve as input to the *libareacenter* library that is in charge of the process of building the global and advance polygons and internally generates the measure of the center of areas that is used for robot navigation. The library output is the velocity of the left and right wheels of the differential system robot.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general y objetivos

Este trabajo se enmarca en el área de la Inteligencia Artificial de la Robótica Perceptual y Autónoma y más precisamente en el control de la navegación de un robot basado en el centro de áreas.

1.1. Motivación y objetivos

La Tesis Doctoral del profesor D. José Manuel Cuadra Troncoso (Cuadra Troncoso (2011)) tuvo como objetivo la investigación de nuevos métodos de navegación reactiva para robots autónomos, considerada como base de cualquier otra forma de navegación a alto nivel. En la citada Tesis se utilizó sensores de rango 2D para obtener un polígono estrellado en base a las mediciones de las distancias a los objetos que rodean al robot. El objetivo de este trabajo es, mejorar la obtención del polígono estrellado que configura los haces del láser al incidir en los obstáculos utilizado en la citada Tesis, mediante una cámara de mapa de profundidad. Los puntos o elementos de imagen del mapa (matriz) de profundidad o distancias generados por la cámara se han de proyectar sobre el plano horizontal. Se ha de pasar de la matriz a un vector que contenga los "puntos" relevantes de dicha matriz. Estos puntos del vector conformarán los vértices del polígono estrellado que será la base del cálculo de su centro de áreas.

El presente trabajo mejora la utilización de medida de rango láser al usar la técnica denominada 2D ½. El láser sólo detecta objetos en su plano de barrido. Incluso los actuales sólo llegan a 128 planos de barrido. Con la citada técnica los objetos pueden estar en tantos planos como resolución vertical tenga la cámara. La resolución horizontal equivale al número de haces del láser pudiendo ser un número superior a 180.

Así se propone alcanzar los siguientes objetivos:

1. Mejorar la visión del "mundo" que genera la medida de rango láser mediante el empleo de cámaras de mapa de profundidad.

2. Aprovechar el uso de cámaras para evolucionar la medida a 3D.

3. Anular el defecto de la variación de la medida láser en función de ángulo de incidencia del haz láser/superficie del objeto para ciertos tipos de materiales y superficies transparentes mediante la utilización de dichas cámaras.

1.2. Estructura de la memoria

La memoria de este proyecto se estructura en los siguientes capítulos:

1. Introducción general y objetivos
2. Controles continuos
3. Cámara de profundidad
4. De la cámara de profundidad al vector de distancias
5. Generación del polígono estrellado
6. Control del robot
7. Aplicación Centroid navigation
8. Simulador CoppeliaSim
9. Experimentos realizados
10. Conclusiones y Trabajos futuros: 3D

Sobre la estructura de esta memoria se ha tratado que sea progresiva a medida que se han ido consiguiendo los objetivos.

El primer capítulo, como su nombre indica, es de introducción y objetivos.

En el capítulo *Controles continuos* se hace una sucinta enumeración de algunos los métodos de navegación que se incluyen en esta categoría para terminar exponiendo el método del centro de áreas como base general de este trabajo.

En el capítulo *Cámara de profundidad* se expone los principios de este componente que se ha de utilizar como medida del entorno.

En el capítulo *De la cámara de profundidad al vector de distancias* se describe como se obtiene el vector de distancias y cuál ha sido el proceso de investigación para llegar a ello.

En el capítulo *Generación del polígono estrellado* y en una primera sección se argumenta la *Necesidad de dos cámaras* y en la siguiente sección *Del vector de distancias a los puntos del polígono estrellado* se introduce la forma de construir el polígono estrellado y se justifica por qué el polígono generado que en principio es un triángulo isósceles hay que convertirlo en un sector circular. Por último, en la sección *Desde 2D ½ a acotado-3D* se introduce el concepto de altura de los objetos sobre el suelo que sería la base para el tratamiento en 3D de estos datos. Sin embargo, se matiza que no hay necesidad de ese concepto sino es suficiente con eliminar parte de la información para los propósitos de este trabajo. Ya se utilizará este concepto en trabajos futuros.

En el capítulo *Control del robot* se hace alusión a la librería *libareacenter* y su utilización en este trabajo.

En el capítulo *Aplicación Centroid navigation* se exponen las partes que constituyen la aplicación que permite comunicarse con el simulador y realizar pruebas y experimentos

En el capítulo *Simulador CoppeliaSim* se exponen dos problemas que se ha encontrado a la hora de su utilización en los experimentos.

En el capítulo *Experimentos realizados* se detallan todos los experimentos realizados y sus resultados.

En el capítulo *Conclusiones y Trabajos futuros: 3D* se alude a la mejora que introduce la utilización de las cámaras de mapa de profundidad con respecto a la medida de rango láser y se apunta como trabajo futuro la utilización de la altura de los objetos que se consideró en el capítulo 5 para conseguir un auténtico 3D.

En resumen, de lo que trata este trabajo es demostrar que la utilización de cámaras de mapa de profundidad frente a la medida de rango láser en la navegación por centro de áreas tiene dos ventajas:

1. Los haces del láser no ven objetos como la cristalera que pudiera existir en un entorno, las cámaras de mapa de profundidad sí. En la Figura 1.1 se puede observar como la trayectoria del robot pasa por detrás del panel transparente en la parte superior derecha.

2. Los haces del láser ven los objetos que estén en el plano de barrido, al utilizar cámaras de mapa de profundidad existen tantos planos como resolución vertical tengan las cámaras. La consecuencia de esto es que se puede medir la altura de los objetos en el campo de visión y se puede navegar por debajo de objetos como, por ejemplo, un puente (ver Figura 1.2) y se puede detectar objetos sobre el suelo (ver Figura 1.3).

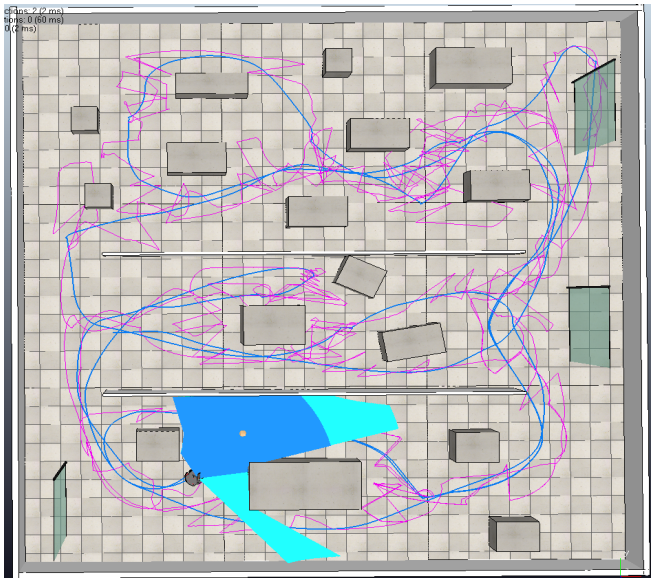

Figura 1.1: Detección de obstáculos transparentes

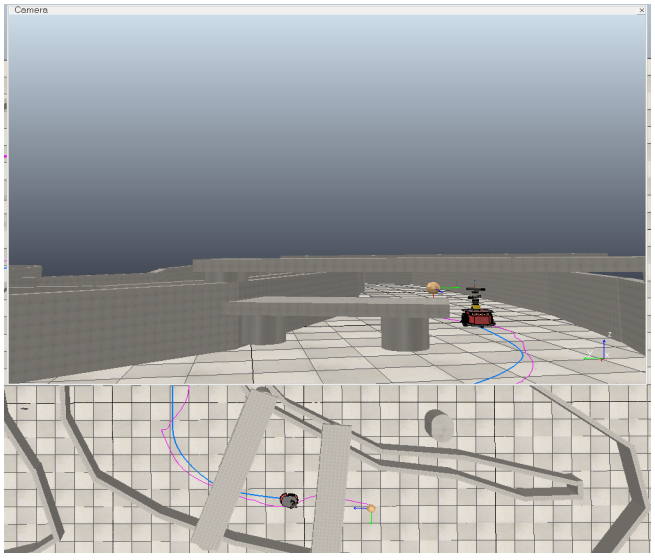

Figura 1.2: Paso por debajo de un puente

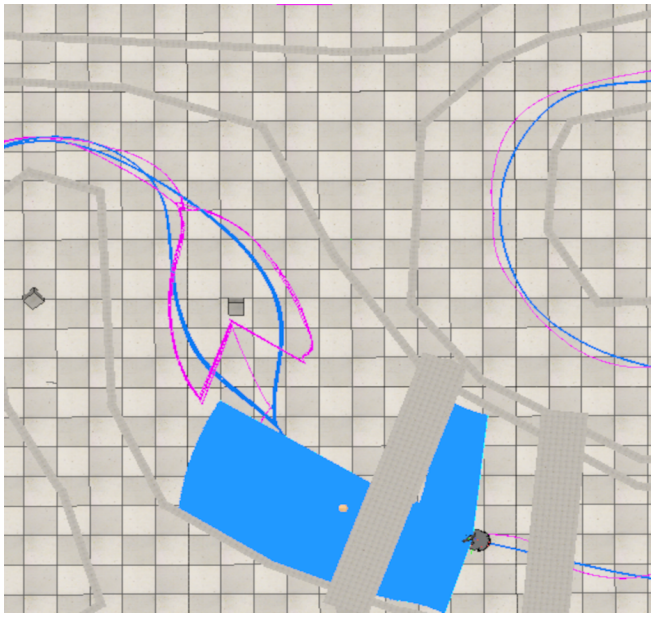

Figura 1.3: Detección de obstáculos sobre el suelo

Capítulo 2

Controles continuos

Se comienza este capítulo con una introducción donde se postula la diferencia entre las estrategias de control locales y las estrategias de control globales. Se realiza un breve recordatorio de los tres paradigmas en el campo de la robótica autónoma y sus primitivas y se enfatiza lo que significa un control reactivo. A continuación, se realiza un repaso a los controles que interesan en este trabajo, es decir, los controles de tipo continuo y las estrategias de control locales de la navegación reactiva. Dentro de estas estrategias, se expone primeramente los métodos de *Campos de potencial*. Debido a las críticas que recibió el método, se propusieron diferentes modificaciones al método como los *Campos de potencial armónicos*, los *Campos de potencial variantes en el tiempo* y las llamadas *Fuerzas virtuales ficticias*. A continuación, se realiza una breve síntesis de lo que significa los métodos de *Histograma de campo vectorial*, se comienza por el VFH o *Vectorial Field Histogram* aludiendo sus deficiencias y se termina con las mejoras introducidas en los métodos VHF+ y VHF*. Tras estos métodos se cita el enfoque de *Ventana dinámica*, DWA, y se comenta sus ventajas y su principal contribución, que es una solución para incorporar a la dinámica en los métodos reactivos para evitar colisiones. Luego, se pasa a describir el método del *Centro de áreas total* y se compara con los métodos de *Campos de potencial*. Se estudian distintos casos de campo de direcciones y se llega a la conclusión de que las fuerzas que mueven el robot no siempre derivan del potencial. Finalmente, se incide en el método del *Centro de áreas básico o parcial*. Se expone todo el proceso que se sigue tras la medida del área libre, es decir, el polígono global y el cálculo de su centro de áreas, así como, dependiendo de la entrada del centro de áreas en una zona no libre, se subdivide el polígono global en dos subpolígonos. Más tarde se seleccionará uno de ellos dependiendo de ciertos criterios para formar lo que se ha de llamar polígono de avance y que mediante el cálculo del centro de áreas de este polígono se fijara el nuevo punto donde ha de navegar el robot por el área libre sorteando los obstáculos. Rebasados estos obstáculos, se generará de nuevo el polígono global y así sucesivamente. Es la base teórica que sostiene el presente trabajo.

2.1. Introducción

La navegación es el proceso de determinar o mantener una trayectoria hacia la ubicación de una meta. Se puede distinguir entre estrategias de control locales y estrategias de control globales Arkin (1998). El concepto *reactive robotics* fue introducido por Brooks (1991), a menudo conocida como robótica basada en el comportamiento *(behavior-based robotics*). La navegación reactiva se incluye dentro de las estrategias locales. Los paradigmas de la robótica son, según Murphy (2005) "Un paradigma es una filosofía o conjunto de asunciones y/o técnicas que caracterizan una aproximación a una clase de problemas". Actualmente se consideran tres paradigmas: Jerárquico, Reactivo, e Híbrido deliberativo/reactivo. También se consideran tres primitivas: SENSE, PLAN, y ACT. En De la Paz López (2003) se expone "Un control reactivo se basa en una colección de pares condición-acción, con un conjunto mínimo de estados internos, ejecutando simplemente la acción correspondiente para cada conjunto sensorial". En termino de las primitivas SENSE luego ACT, es decir, no existe PLAN o planificación. La información de los sensores -SENSE- es local entendiéndose como parcial del "mundo".

2.2. Tipos de control continuo

Los controles que aquí se exponen pertenecen a las estrategias de control locales de la navegación reactiva y no los de las estrategias de control globales Arkin (1998).

2.2.1. Métodos de campos de potencial

Los campos de potencial fueron introducidos por Krogh (1984). Lo fundamental en la estrategia de los métodos de potencial (Potential field, Virtual Forces Field, VFF) es el vector de acción que se corresponde aproximadamente a la velocidad y la orientación de un robot en movimiento. Los campos de potencial siempre utilizan vectores para representar comportamientos y la suma de vectores para combinar vectores de diferentes comportamientos para producir un comportamiento emergente.

El primer principio de una arquitectura de campos potenciales es que la acción motora de un comportamiento debe representarse como un campo potencial. El campo potencial es una matriz, o campo, de vectores. Un vector es una construcción matemática que consiste en una magnitud y una dirección. Los vectores se utilizan a menudo para representar una fuerza de algún tipo. Por lo general, se dibujan como una flecha, donde la longitud de la flecha es la magnitud de la fuerza y el ángulo de la flecha es la dirección. Por lo general, los vectores se representan con una letra mayúscula en negrita, por ejemplo, $\mathbf{V}$. Un vector también puede escribirse como una tupla, donde representa la magnitud y la dirección. Por convención, la magnitud es un número real entre 0.0 y 1.0, pero la magnitud puede ser cualquier número real. La matriz representa una región del espacio. En la mayoría de las aplicaciones robóticas, el espacio está en dos dimensiones, representando una vista de pájaro del mundo como un mapa. El mapa se puede dividir en cuadrados, creando una cuadrícula (x, y). Cada elemento de la matriz representa un cuadrado del espacio. Los objetos perceptibles en el mundo ejercen un campo de fuerza en el espacio circundante. El campo de fuerza es análogo a un campo magnético o de gravitación. Se puede pensar en el robot como una partícula que ha entrado en el campo expulsado por un objeto o entorno. El vector en cada elemento representa la fuerza, tanto la dirección para girar como la magnitud o velocidad para dirigirse en esa dirección, que un robot sentiría si estuviera en ese lugar en particular. Los campos de potencial son continuos porque no importa lo pequeño que sea el elemento. En cada punto en el espacio, hay un vector asociado.

Los campos de potencial no están exentos de críticas. Fue en la *IEEE international conference on robotics and automation* donde Koren and Borestein (1991) pusieron de manifiesto las carencias del método tales como, la presencia de mínimos locales en el campo de fuerzas, donde el robot queda atrapado, y conductas cíclicas oscilatorias. Como consecuencia de estos problemas se han propuesto diversas modificaciones al método. Por ejemplo, los denominados *Campos de potencial armónicos* que para evitar los mínimos locales se utiliza funciones armónicas. Fueron utilizados por Kim and Khostola (1992), Jacob and Slotine (1997), Masoud and Masoud (2000) y Fahimi and Ashrafiuon (2009). Otra propuesta fue los *Campos de potencial variantes en el tiempo* de Tianmiao and Bo (1992), Ferrara and Rubagotti (2009) y Goncalves et al. (2010). Ahora el campo tiene una componente más, la componente temporal, que se une a la espacial. Otra apuesta ha sido las llamadas *Fuerzas virtuales ficticias* de Ferrara and Ruba-

gotti (1996), Maravall and Serradilla (2000) cuya virtualidad obedece a que los mínimos locales repelen al robot. Aún más arriesgada es la propuesta de Clark and Ram (1992) y Ram et al. (1997). Los autores describen un sistema de control robótico reactivo que incorpora aspectos del aprendizaje automático para mejorar la capacidad del sistema para navegar con éxito en entornos desconocidos.

2.2.2. Histograma de campo vectorial

El VFH o *Vectorial Field Histogram* fue introducido por Borenstein and Koren (1991) y permite la detección de obstáculos desconocidos y evita colisiones mientras, simultáneamente, conduce el robot móvil hacia la meta. El método VFH utiliza una cuadrícula-histograma en dos dimensiones como modelo del "mundo". Este modelo del mundo se actualiza continuamente con la medida de los sensores. El método VFH emplea dos etapas de reducción de datos para computar los comandos deseados para el vehículo. En la primera etapa, la cuadrícula-histograma se reduce a un histograma polar de una dimensión estructurado alrededor de la pose momentánea del robot. Cada sector en el histograma polar contiene un valor que representa la densidad de los obstáculos polares. En la segunda etapa, el algoritmo selecciona el sector más adecuado de entre todos los sectores del histograma polar con una baja densidad de obstáculos polares, y la dirección del robot se alinea con esa dirección. Desafortunadamente, este método presenta los mismos problemas que los métodos de campo de potencial. Hay dos variantes que mejora este método VHF+ (Ulrich and Borenstein (1998)) y VHF* (Ulrich and Borenstein (2000)). El método VHF+ emplea un proceso de reducción de datos de cuatro etapas para calcular la nueva dirección de movimiento. Por su parte el algoritmo VHF* utiliza el algoritmo A* para para seleccionar el comando de movimiento que minimice la función de costo $f(c) = g(c)+h(c)$, donde $g(c)$ se calcula de manera similar a la función de costo utilizada en VFH+ para cada candidato dirección y $h(c)$ es un heurístico. Una versión modificada del método anterior del histograma de campo vectorial (VFH), se llama *Double-VFH* (DVFH) Yang et al. (2001). El método DVFH tiene en cuenta la huella rectangular del OmniMate (gran plataforma rectangular con una plataforma de carga plana que mide 18492 cm.) al calcular simultáneamente dos histogramas de campo vectorial centrados en dos puntos a lo largo del eje longitudinal del OmniMate.

2.2.3. Ventana dinámica

El enfoque de ventana dinámica, DWA (Fox and Burgard (1997)) se utiliza para evitar colisiones reactivas para robots móviles equipados con accionamientos sincronizados. El enfoque se deriva directamente de la dinámica de movimiento del robot y, por lo tanto, es particularmente adecuado para robots que operan a alta velocidad. Se diferencia de los enfoques anteriores en que la búsqueda de comandos que controlan la velocidad de traslación y rotación del robot se realiza directamente en el espacio de velocidades. La ventaja del enfoque es que incorpora de

forma correcta y elegante la dinámica del robot. Esto se hace reduciendo el espacio de búsqueda a la ventana dinámica, que consiste en las velocidades alcanzables en un corto intervalo de tiempo. Dentro de la ventana dinámica, el enfoque solo considera las velocidades admisibles que dan como resultado una trayectoria en la que el robot puede detenerse de manera segura. Entre estas velocidades, la combinación de velocidad de traslación y de rotación se elige maximizando una función objetivo.

La principal contribución del trabajo de Minguez et al. (2002) es una solución para incorporar a la dinámica en los métodos reactivos para evitar colisiones. Se propone utilizar las restricciones dinámicas para construir una nueva representación espacial *Ego-Dynamic Space*, donde las restricciones dinámicas están implícitamente representadas. Luego, con modificaciones menores, se pueden aplicar a este espacio métodos de navegación reactiva listos para usar que no tienen en cuenta las restricciones dinámicas. Los comandos de movimiento calculados implícitamente tienen en cuenta la dinámica específica del robot, lo que garantiza una ejecución factible del movimiento.

En Ögren and Leonard (2005) se ve el DWA como un método de control predictivo de modelo (MPC) y usa el marco de control de la función de Lyapunov (CLF) de Rimon y Koditschek, se inspira en un marco MPC/CLF presentado por Primbs para proponer una versión del DWA que sea manejable y convergente.

2.2.4. El centro de áreas total

El calificativo de total del centro de áreas (CA/AC) se aplica a un robot equipado con sensores de rango que generan medidas en 360° alrededor del propio robot. El centro de áreas es equivalente en dos dimensiones al centro de masas o de gravedad. La fórmula en coordenadas cartesianas para el caso de un polígono de n vértices $\{(x_j, y_j)\}$ es:

$$x_{CA} = \frac{1}{6A}\sum_{j=1}^{n-1}\left(x_j y_{j+1} - x_{j+1}y_j\right)\left(x_j + x_{j+1}\right) f\left(x_j, y_j, x_{j+1}y_{j+1}\right) \tag{2.1}$$

$$y_{CA} = \frac{1}{6A}\sum_{j=1}^{n-1}\left(x_j y_{j+1} - x_{j+1}y_j\right)\left(y_j + y_{j+1}\right) f\left(x_j, y_j, x_{j+1}y_{j+1}\right) \tag{2.2}$$

donde A es el área de polígono y f es la función de densidad de área que por lo general se utiliza $f \equiv 1$.

En Álvarez et al. (1999) se utiliza por primera vez el centro de áreas para conseguir un sistema de referencia para la integración de datos sensoriales. En la tesis doctoral de uno de los directores de este trabajo (De la Paz López (2003)) se comenzó a utilizar el centro de áreas para la navegación del robot. El centro de áreas total calculado ejercía una fuerza de atracción sobre el robot y a medida que el robot se acercaba al centro de áreas la fuerza se volvía repulsiva y como consecuencia el robot tenía que buscar otro centro de áreas. De esta manera la navegación

se producía a través de los sucesivos centros de áreas calculados.

Como quiera que el centro de áreas genera campos de fuerzas se podría llegar a la conclusión de que este método es una variante de los métodos de potencial antes expuestos. En *Mathematical Foundations of the Center of Area Method for Robot Navigation* (de la Paz, Félix and Álvarez, José R. and Rosado Sáchez, José Ignacio and Cuadra Troncoso, José Manuel (2009)) se demuestra que las fuerzas que mueven el robot no siempre derivan del potencial. La dirección seguida por el robot será dada por:

$$\boldsymbol{D}(r) = \int_{A(r)} \boldsymbol{F}\left(\tilde{\boldsymbol{r}}\right) dA \tag{2.3}$$

Donde $\boldsymbol{F}$ es la fuerza ejercida por un infinitesimal de área dA. Se denota la fuerza resultante por $\boldsymbol{D}$ para significar dirección y no el movimiento del robot que pudiera causar la fuerza. $\boldsymbol{D}$ depende de la posición del robot con respecto al centro de áreas del área libre. En general, no se puede asegurar que $\boldsymbol{D}$ sea derivable del potencial aún en el caso de que las fuerzas $\boldsymbol{F}$ sean derivables del potencial.

Se estudió casos particulares de campos de potencial como una pared recta Figura 2.1, una habitación cerrada Figura 2.2, una habitación abierta Figura 2.3 y un pasillo con una curva Figura 2.4 y Figura 2.5. Con respecto al pasillo con curva y por simetría se tiene que el el centroide estacionario está en la recta $x = y$. En esta recta el vector de campo $\boldsymbol{D}$ es:

$$D\left(x, x\right) = \frac{1}{6}\varphi\left(x\right)\left(1, 1\right) \tag{2.4}$$

donde $\varphi = 3a^3 + \left(a - x\right)^3 + x^3 - 3l^2a - 6a^2x + 2\left(l^2 - x^2\right)^{3/2} - 2\left(l^2 - \left(a - x\right)^2\right)^{3/2}$, siendo a el ancho del pasillo, $0 \leq x$,$y \leq a$ y $l > a\sqrt{2}$. l es el alcance de los sensores. Cuando $\varphi\left(0\right) \geq 0$ el centroide se localiza dentro del alcance de los sensores del robot Figura 2.4, cuando $\varphi\left(0\right) < 0$ el centroide estará fuera del alcance de los sensores del robot Figura 2.5.

En los casos de la habitación abierta y el pasillo con curva se demuestra que la dirección de la fuerza no es derivable del potencial. Además, los resultados en el caso de un pasillo con curva muestran que, debido a la aludida simetría especial de disposición, existe un valor máximo para la relación entre el alcance de los sensores y el ancho del pasillo para que el centroide esté dentro del pasillo y por lo tanto ser accesible.

Expresando la fuerza para la navegación como una integral del contorno, se computa la dirección de la fuerza virtual (por decir, el centro de áreas) utilizando la información disponible por el robot, que no es otra que el contorno creado por la medida de los sensores. Toda vez que se selecciona el espacio libre como una fuerza virtual llegamos a una fuerza de dirección de navegación equivalente al centro de áreas o centroide del polígono generado con los valores de las medidas de los sensores de rango.

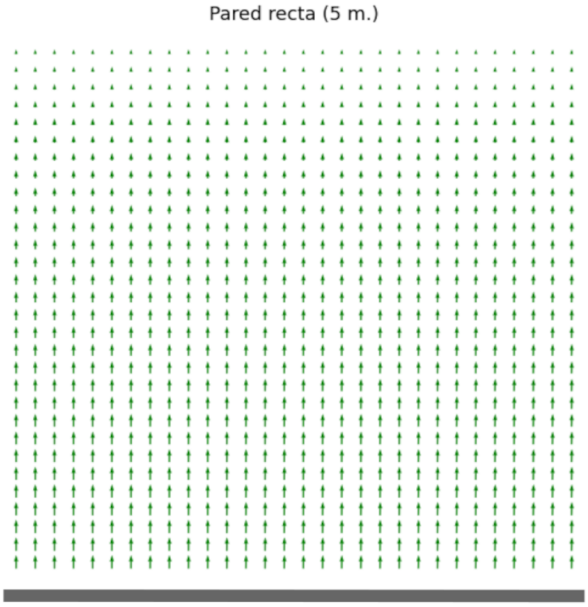

Figura 2.1: Campo de direcciones: Pared recta

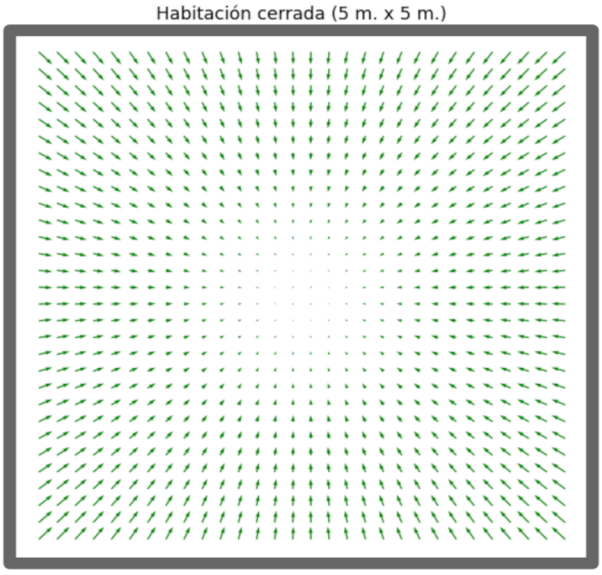

Figura 2.2: Campo de direcciones: Habitación cerrada

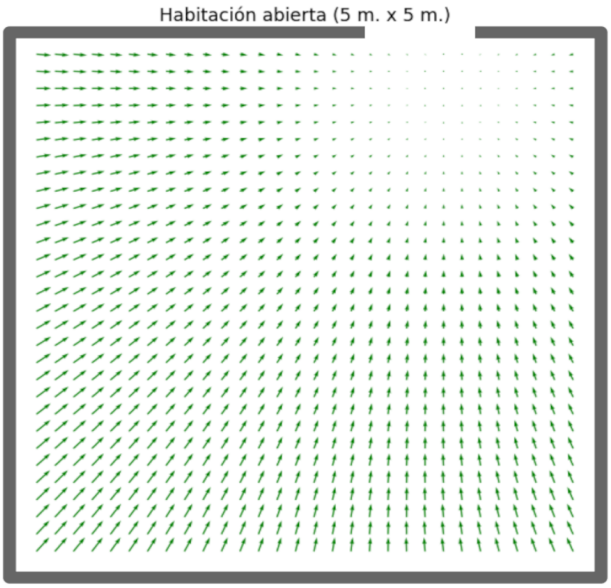

Figura 2.3: Campo de direcciones: Habitación abierta

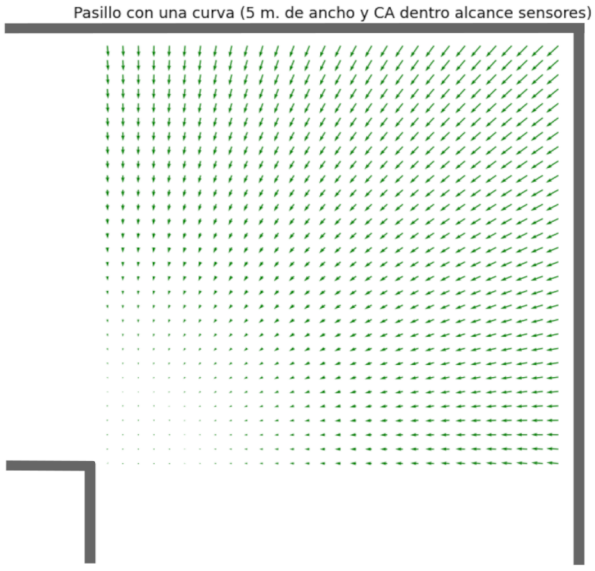

Figura 2.4: Campo de direcciones: Pasillo con una curva. CA dentro del alcance de los sensores

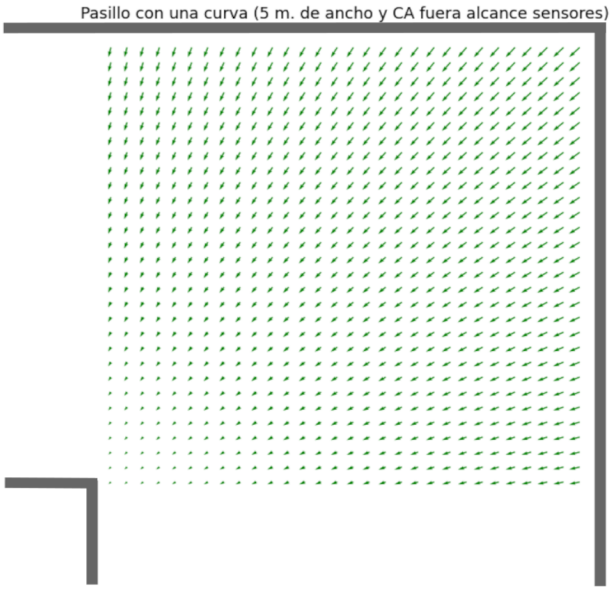

Figura 2.5: Campo de direcciones: Pasillo con una curva. CA fuera del alcance de los sensores

2.2.5. El centro de áreas básico

En Sánchez et al. (2009) y Sánchez et al. (2010) y en la tesis doctoral de uno de los directores de este trabajo (Cuadra Troncoso (2011)) se exponen las ideas sobre este método. Se construye un polígono global en base a la medida de los sensores (de rango, cámaras de mapa de profundidad). Se calcula el centro de áreas del polígono global. El robot navega hacia el centro de áreas mientras este sea accesible. Cuando el centro de áreas calculado no sea accesible, es decir, se sitúa en un obstáculo, se crea un punto de división - *split point* - como punto de intersección entre la recta que une al robot con el centro de áreas y el propio obstáculo. Este punto de división divide al polígono global en dos subpolígonos. Atendiendo a ciertos criterios se selecciona uno de ellos y pasa a ser el polígono de avance calculándose su centro de áreas. Cuando el robot sobrepasa el punto de división se vuelve al polígono global. La distancia de alcance del polígono de avance es función de su amplitud y esta es función del punto de división. También se tiene en cuenta el tamaño del robot.

Se ha introducido una modificación del método para el alcance de objetivos. Aquí la selección del polígono se realiza en función del objetivo a alcanzar.

Capítulo 3

Cámara de profundidad

El objetivo de este capítulo es introducir unas nociones teóricas de lo que significa una cámara de mapa de disparidad o mapa de profundidad, de las fases de rectificación, algoritmo estéreo matching, cálculo de la disparidad y su paso a profundidad. Se comienza detallando lo que significa el denominado *modelo de cámara Pinhole,* la base matemática de sus parámetros intrínsecos y sus parámetros extrínsecos. A continuación, se expone un sistema de visión estéreo y se realiza una breve incursión en lo que significa la *Geometría epipolar*, resultando la línea epipolar fundamental para encontrar la correspondencia entre los puntos en el plano de imagen de cada cámara, del punto del espacio visionado. Con la idea de mejorar el coste computacional, se recurre al proceso de *Rectificación* que no es más que los planos de imagen de las dos cámaras que componen el conjunto estéreo sean coplanares. Así, la correspondencia entre los puntos del plano de cada imagen se realiza en una única línea epipolar. El algoritmo encargado de buscar la correspondencia se denomina *Stereo matching*. La salida de este algoritmo es una imagen o matriz de número de columnas igual a la resolución horizontal de las cámaras y número de filas igual a la resolución vertical. Cada elemento de la matriz o píxel de la imagen almacena el valor de disparidad o distancia horizontal absoluta entre la posición de un píxel en la imagen izquierda con su correspondiente en la imagen derecha. La imagen así obtenida se denomina *Disparity Map* o mapa de disparidad. A través del proceso de *Triangulación* se convierte el Disparity Map en *Depth map* o mapa de profundidad. El conjunto binocular se denomina *Depth map camera* o Cámara de mapa de profundidad y en lo que respecta a este trabajo, es el "sensor" que interaccionará con el "mundo".

3.1. El modelo de cámara Pinhole

El modelo de cámara Pinhole representa una relación biyectiva entre las coordenadas de un punto en el espacio 3D y su proyección en el plano de imagen (Hartley and Zisserman (2004)). En la Figura 3.1 se representa esta relación. El punto C es el centro de la cámara y origen de

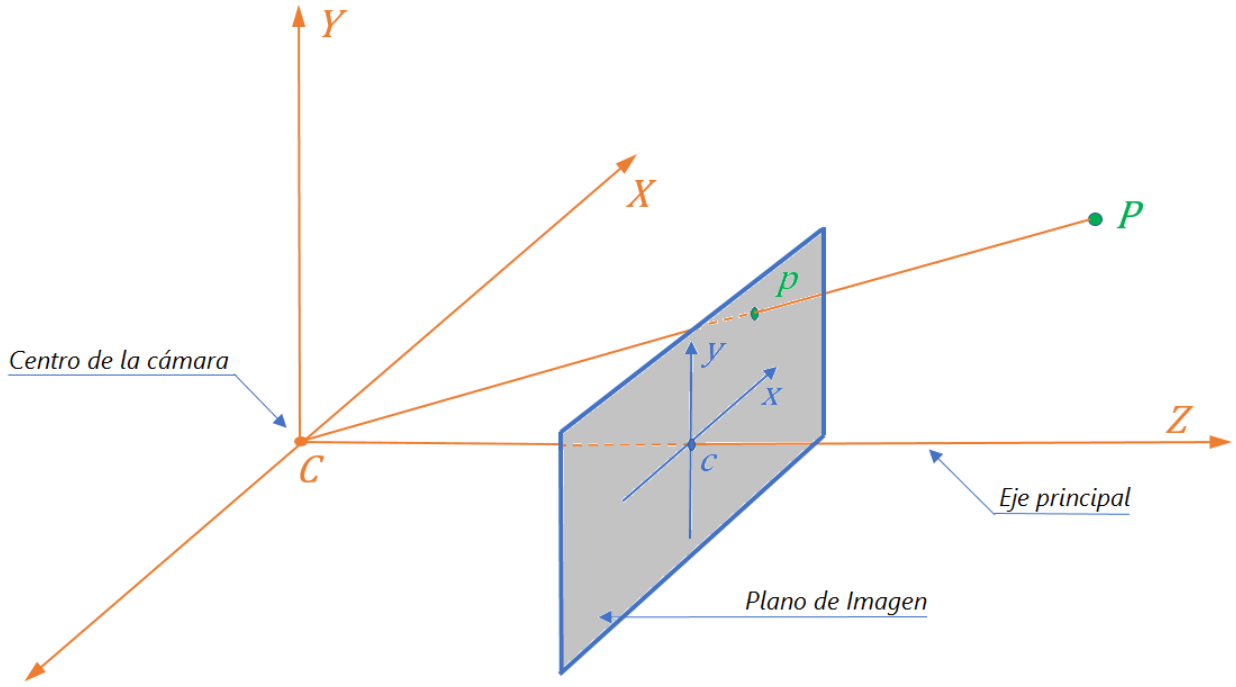

Figura 3.1: Geometría del modelo Pinhole

coordenadas. El eje principal del sistema de visión es el eje Z. El plano de imagen es paralelo al plano que pasa por los ejes X e Y a una distancia del origen de coordenadas C o centro de la cámara igual a la distancia focal $f$en la dirección del eje Z. El punto c o punto principal de la imagen, centro de la imagen, es la intersección del eje Z y el plano de la imagen.

Si representamos un punto P del espacio 3D en coordenadas homogéneas $(X, Y, Z, 1)^T$su proyección en el plano de imagen será el punto p intersección de la recta PC con dicho plano. Atendiendo a la Figura 3.2 y por semejanza de triángulos se establece la siguiente relación:

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{3.1}$$

Donde x e y son las coordenadas del punto proyección p con respecto al centro de imagen c. Desde aquí:

$$x = \frac{f \times X}{Z} \tag{3.2}$$

$$y = \frac{f \times Y}{Z} \tag{3.3}$$

En forma matricial:

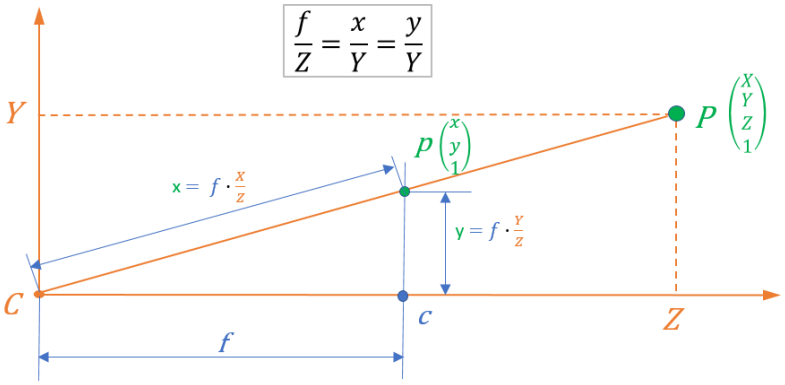

Figura 3.2: Geometría Pinhole. Relaciones

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f \times X \\ f \times Y \\ Z \end{pmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

Las coordenadas X, Y, Z, x, y la distancia focal f están en unidades de longitud (m., cm., mm., etc.). En la práctica, las coordenadas de la cámara se miden en distancias de píxeles. Por eso, hay que introducir un factor de escala:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} s_x \times f & 0 & 0 & 0 \\ 0 & s_y \times f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.5}$$

Finalmente hay que introducir un factor de sesgo α que tenga en cuenta el corte del sistema de coordenadas:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_x & \alpha & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.6}$$

Las coordenadas en píxeles no se dan con respecto al plano de imagen que está centrado en el eje principal del sistema o eje óptico, sino se dan con respecto al origen de coordenadas 2D del plano de imagen:

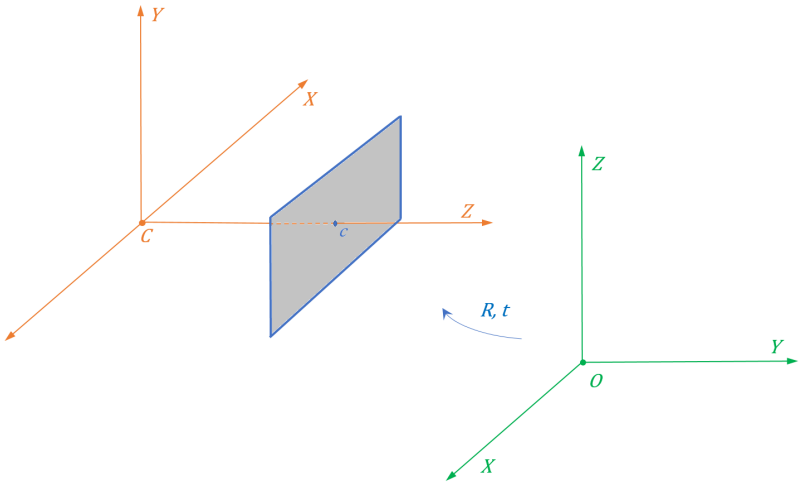

Figura 3.3: Relación coordenadas espacio 3D con las coordenadas de la cámara

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_x & \alpha & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.7}$$

o bien,

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim K \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.8}$$

Donde K es la denominada *matriz intrínseca* de la cámara:

$$K = \begin{bmatrix} f_x & \alpha & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.9}$$

La matriz K relaciona los puntos del espacio 3D con el plano de visión donde Z es el eje principal o eje óptico. Se está asumiendo que el centro de la cámara y el de coordenadas del espacio 3D es el mismo punto y en realidad no es así. Para solventar este problema se realiza una *R*otación seguida de una *t*ranslación. Ver Figura 3.3.

Así:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim K \cdot [R|t] \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.10}$$

Donde:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.11}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.12}$$

R y t son los *parámetros extrínsecos* de la cámara. En el modelo *Pinhole* se define la matriz de proyección P como:

$$P = K \cdot [R|t] \tag{3.13}$$

3.2. Geometría epipolar

Los sistemas de visión estéreo utilizan dos cámaras *Pinhole* unidas mediante una conexión física denominada *Línea base* (Orozco et al. (2017)).

En la geometría epipolar (ver Figura 3.4) un punto P del espacio 3D se corresponde a dos puntos p_i y p_d de los planos de la cámara izquierda y la cámara derecha respectivamente.

Con la calibración de cada cámara mediante un proceso de toma de imágenes de un tablero de ajedrez dispuesto en distintas posiciones se obtiene los parámetros intrínsecos de cada una de ellas, matrices K.

Con la denominada calibración estéreo por el mismo procedimiento, pero ahora con pares de imágenes se obtiene los parámetros extrínsecos, matrices R y t.

Finalmente, con esta calibración se obtienen las matrices de proyección de cada cámara P_i y P_d.

Así un punto Q del espacio 3D se traduce en:

$$q_i = P_i \cdot Q \tag{3.14}$$

$$q_d = P_d \cdot Q \tag{3.15}$$

Se ha de plantear la siguiente la siguiente cuestión: ¿cómo encontrar la correspondencia entre los puntos q_i y q_d? La respuesta es la línea epipolar: las coordenadas de un punto en una vista, sus coordenadas en la otra están limitadas en una línea epipolar.

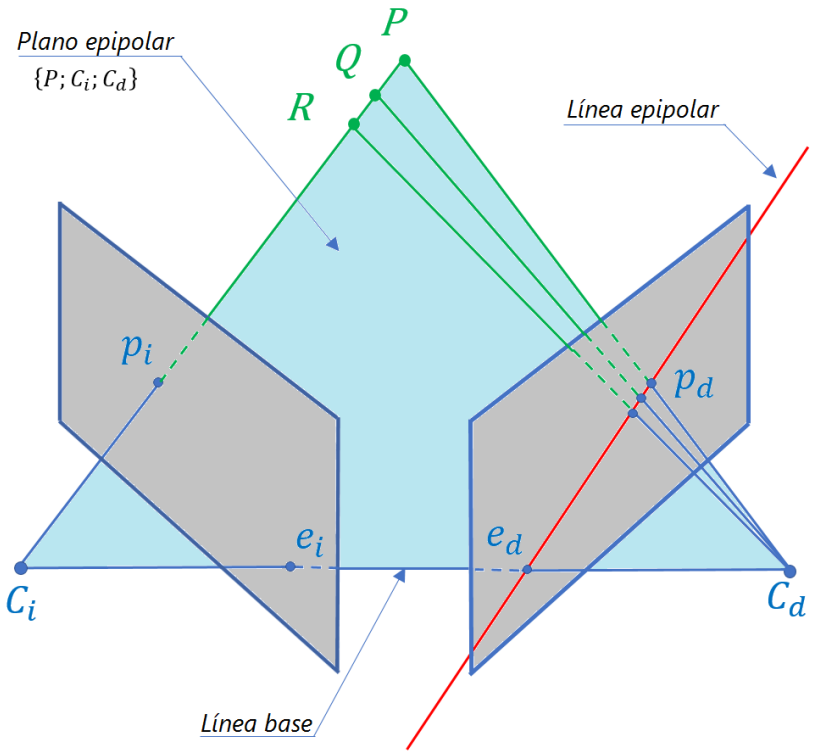

Figura 3.4: Geometría epipolar

3.3. Rectificación

Ahora bien, para que la computación de la correspondencia tenga un menor coste computacional se somete a esta geometría a un proceso de Rectificación. Mediante la Rectificación los planos de ambas cámaras son coplanares. Ver Figura 3.5.

Así el proceso de correspondencia se simplifica a una sola dirección: Una línea paralela a la Línea base.

3.4. Algoritmo *Stereo Matching*

El algoritmo *Stereo Matching* busca los píxeles de la primera imagen en la segunda siguiendo la misma línea epipolar.

La salida del algoritmo es una imagen de dimensiones *ancho* × *alto* que se corresponde a la resolución de las cámaras llamada *Disparity Map*. Cada pixel almacena el valor de disparidad o distancia horizontal absoluta entre la posición de un píxel en la imagen izquierda con su correspondiente en la imagen derecha.

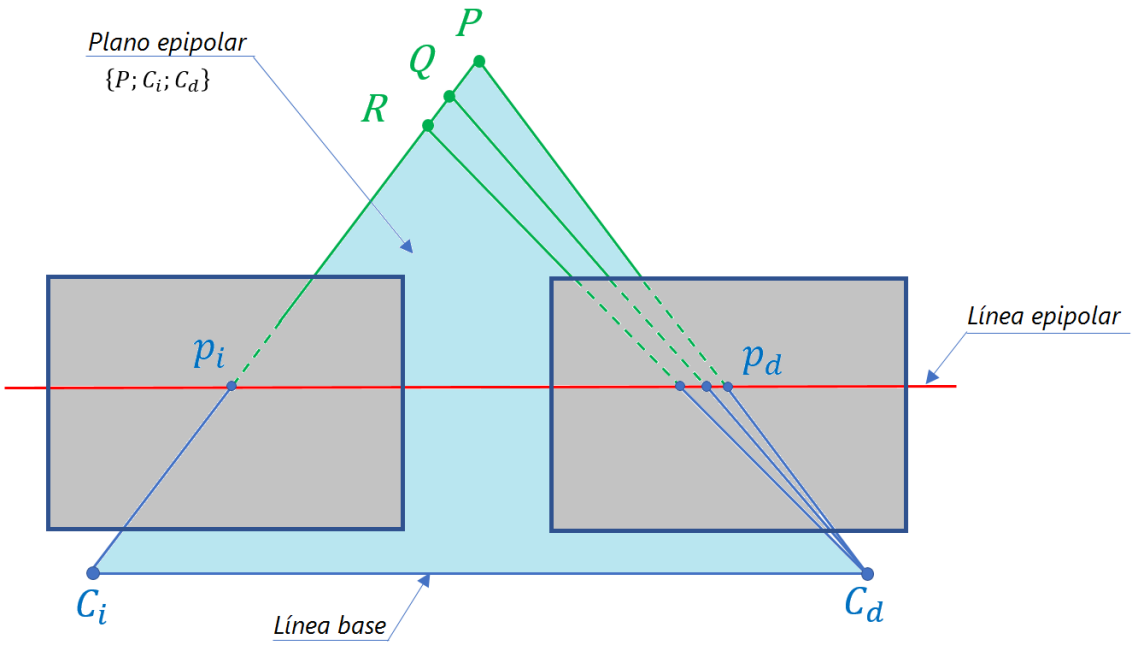

Figura 3.5: Rectificación

3.5. Cálculo de disparidad. Triangulación.

Para conseguir este valor almacenado en cada píxel, se hace uso de la triangulación. En la Figura 3.6 el triángulo $\{P; C_i; C_d\}$ es semejante al triángulo $\{C_d; p_d; p_i^{'}\}$ y se pueden deducir la siguiente ecuación:

$$\frac{Z}{b} = \frac{f}{x_i - x_d} \tag{3.16}$$

donde Z es la profundidad o distancia del punto P a la Línea base, f es la distancia focal y x_i, x_d es el valor de la posición del píxel en la imagen izquierda e imagen derecha respectivamente.

Así:

$$Z = \frac{f \times b}{x_i - x_d} \tag{3.17}$$

La imagen producida con el cálculo anterior se denomina *Depth map* o mapa de profundidad y el conjunto binocular se denomina *Depth map camera* o Cámara de mapa de profundidad.

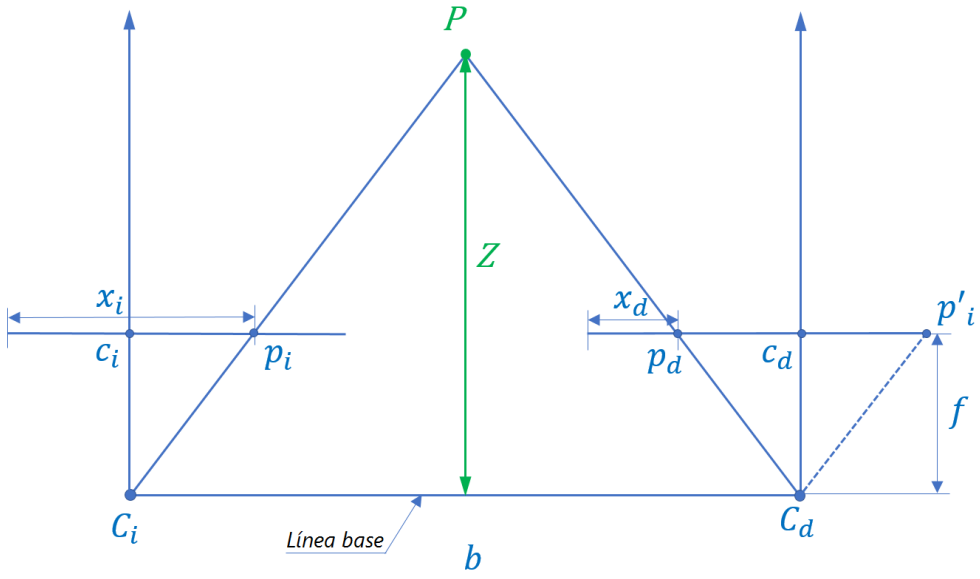

Figura 3.6: Triangulación

Capítulo 4

De la cámara de profundidad al vector de distancias

El presente capítulo trata de expresar el proceso que se ha llevado para conseguir un algoritmo que traduzca los elementos de imagen del mapa de profundidad en un vector de distancias a los objetos presentes en la imagen desde el plano de imagen. En realidad todo este proceso es un proceso de transformación de datos. Se parte de una matriz de distancias o mapa de profundidad y se ha verificado como cambian estos datos cuando ante el objetivo de la cámara aparecen distintos objetos en distintas posiciones. Basándonos en este conjunto de datos, el raciocinio de esta investigación va sugiriendo la idea de cómo transformar los datos de entrada en los datos de salida, que no es más que un vector que contendrá las distancias desde el plano de imagen a los objetos presentes en el plano de imagen. Al final se crea un algoritmo que realiza la transformación de los datos. Para visionar el proceso imaginemos un cubo que contiene todos los objetos en el campo de visión. Se toma una de las caras verticales del cubo como plano de visión. El cubo tiene tantas ranuras horizontales como resolución horizontal tenga la cámara y perpendiculares a la cara tomada como plano de visión. Ahora se proyecta todos los puntos pertenecientes a las aristas de los objetos sobre la base del cubo o plano horizontal inferior del cubo. Los puntos estarán en las ranuras del plano horizontal. Si tomamos el punto más cercano de cada ranura del plano horizontal a la arista inferior o intersección con el plano horizontal se habrá obtenido los puntos que formarán más tarde el polígono estrellado. Es el denominado vector de distancias.

4.1. Cámara de profundidad

La cámara de mapa de profundidad entrega una matriz con un número de filas correspondiente a su resolución (números de píxeles) vertical y un número de columnas correspondiente a su resolución horizontal, ver Figura 4.1. Los valores de los elementos de la matriz es la distancia

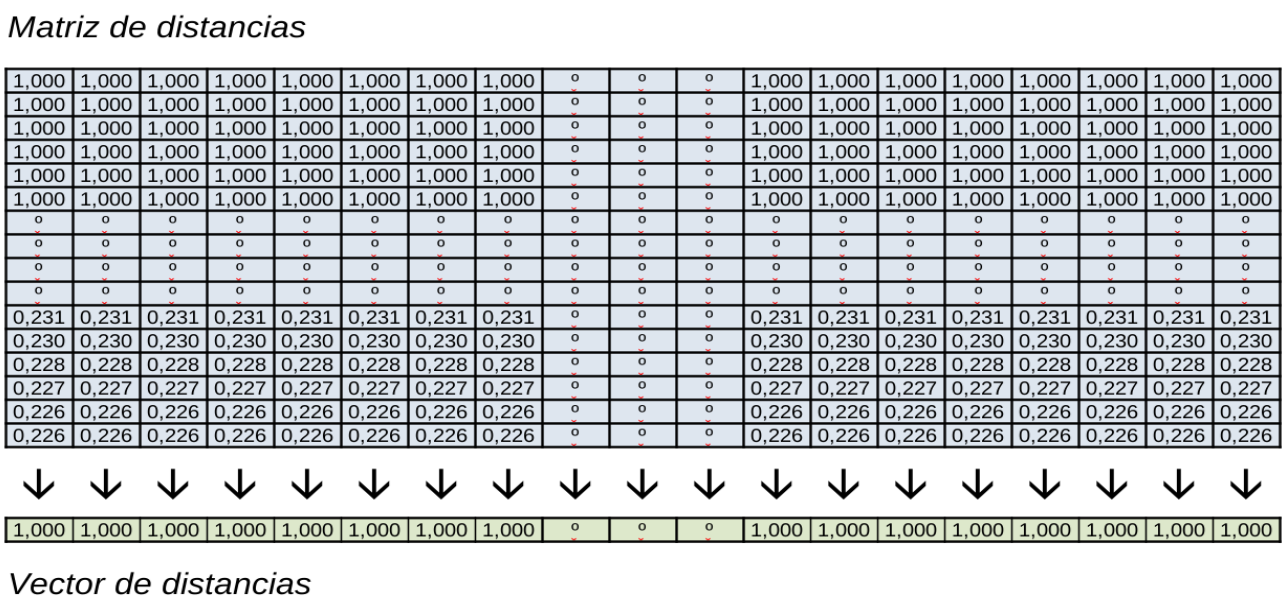

Figura 4.1: Matriz de distancias y Vector de distancias

desde la cámara al objeto visualizado en ese elemento. Estos valores suelen estar normalizados en un rango de 0.0 a 1.0. El valor 0.0 coincide con el denominado "near clipping plane" o plano de corte cercano y 1.0 coincide con el "far clipping plane" o plano de corte lejano. En los experimentos se ha fijado a 0.01 metros y 5.0 metros respectivamente. Otro parámetro de la cámara a tener en cuenta es el ángulo de visión que se ha establecido en 86º que es el valor de la cámara RealSense de Intel D435 (Depth Field of View (FOV): 86° × 57° (±3°)).

4.2. Imágenes en estudio

Los sucesivos apartados pretenden explicitar la línea de investigación que ha llevado a la consecución del algoritmo que trasforma la matriz de distancias generadas por la cámara a un vector de distancias que se utilizará para generar los puntos del polígono estrellado y el cálculo de su centro de áreas.

4.2.1. Imagen sin ningún objeto

La Figura 4.2 muestra una imagen sin objetos. De la matriz de esta imagen se puede graficar los valores pertenecientes a una columna.

La gráfica de la Figura 4.3 muestra una sucesión de valores iguales a 1.0 y sobre la fila 312 aproximadamente (horizonte) va decreciendo ese valor formando una curva.

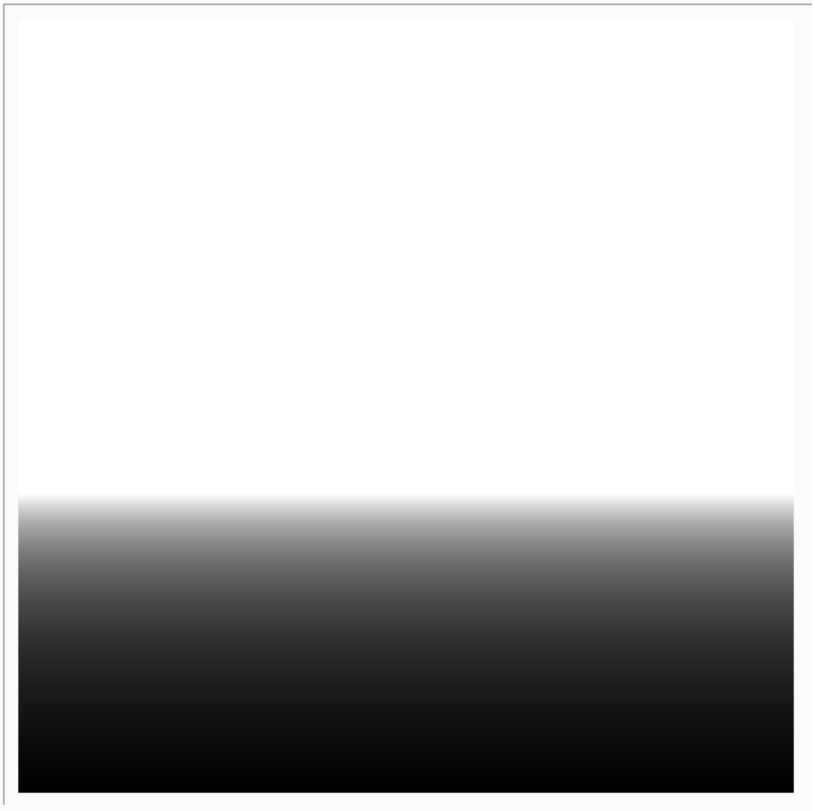

Figura 4.2: Imagen sin objetos

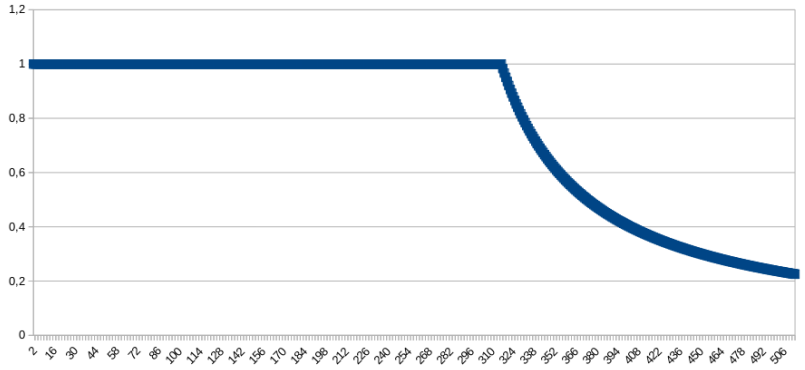

Figura 4.3: Gráfica de valores de una columna de la matriz

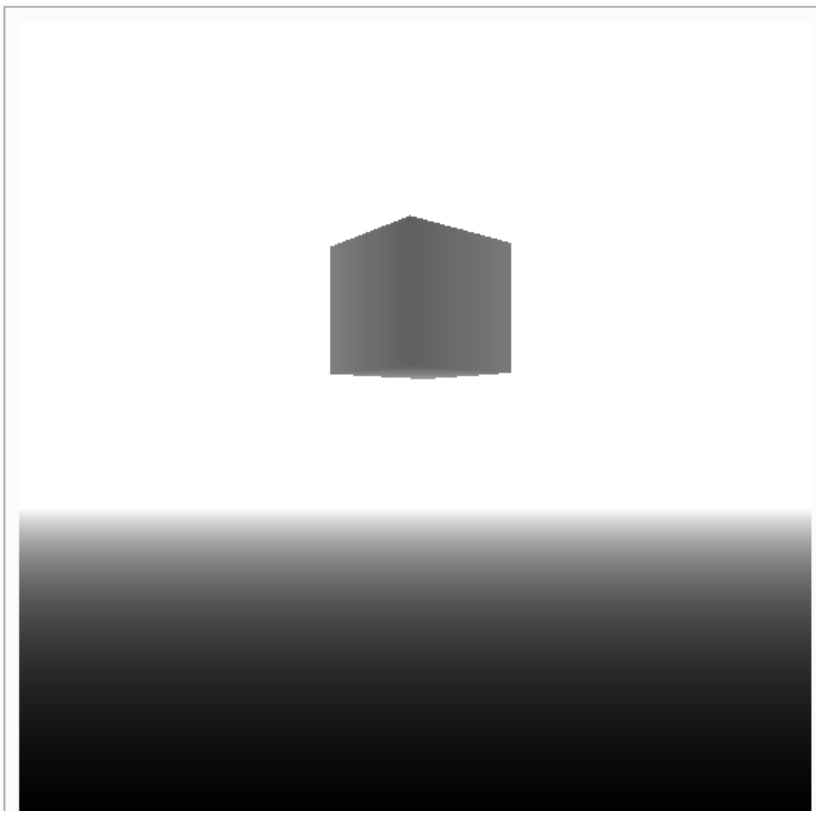

Figura 4.4: Imagen con objeto suspendido

4.2.2. Imagen con un objeto suspendido

La Figura 4.4 muestra una imagen con un objeto suspendido. Se procede como en el apartado anterior y se obtiene la gráfica de la Figura 4.5.

En la gráfica se puede apreciar la posición del objeto.

4.2.3. Imagen con un objeto por debajo del horizonte

La Figura 4.6 muestra una nueva posición de un objeto. Ahora el objeto está por debajo de la línea del horizonte. La correspondiente gráfica en la Figura 4.7.

4.2.4. Imagen con un objeto detrás de otro

La Figura 4.8 muestra dos objetos, el de menor dimensiones está más cerca de la cámara que el de mayores dimensiones. La gráfica de la Figura 4.9 muestra los valores de la columna central.

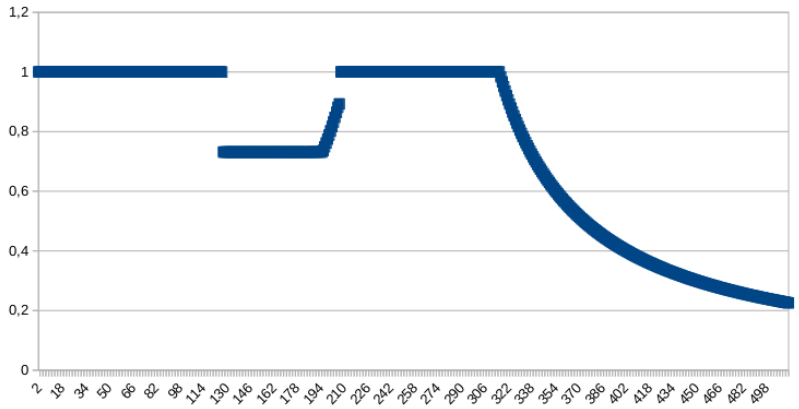

Figura 4.5: Gráfica de valores de una columna de la matriz de la imagen de la Figura 4.4

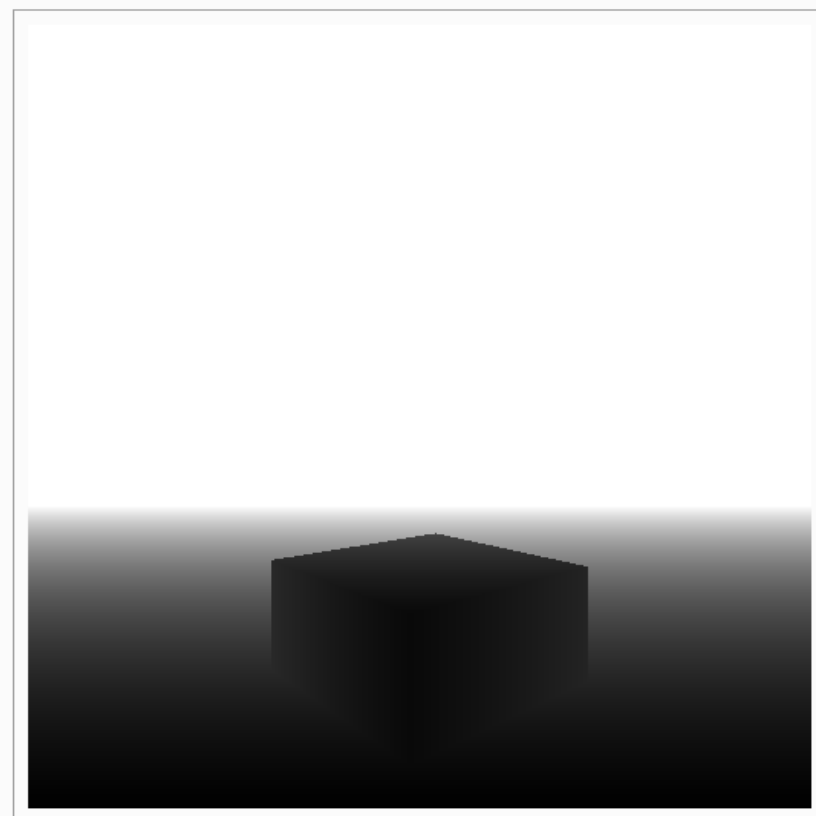

Figura 4.6: Objeto por debajo del horizonte

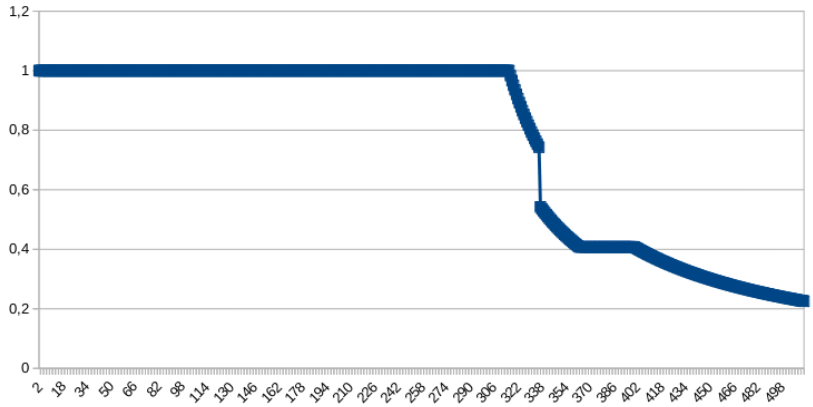

Figura 4.7: Gráfica de valores de una columna de la matriz de la imagen de la Figura 4.6

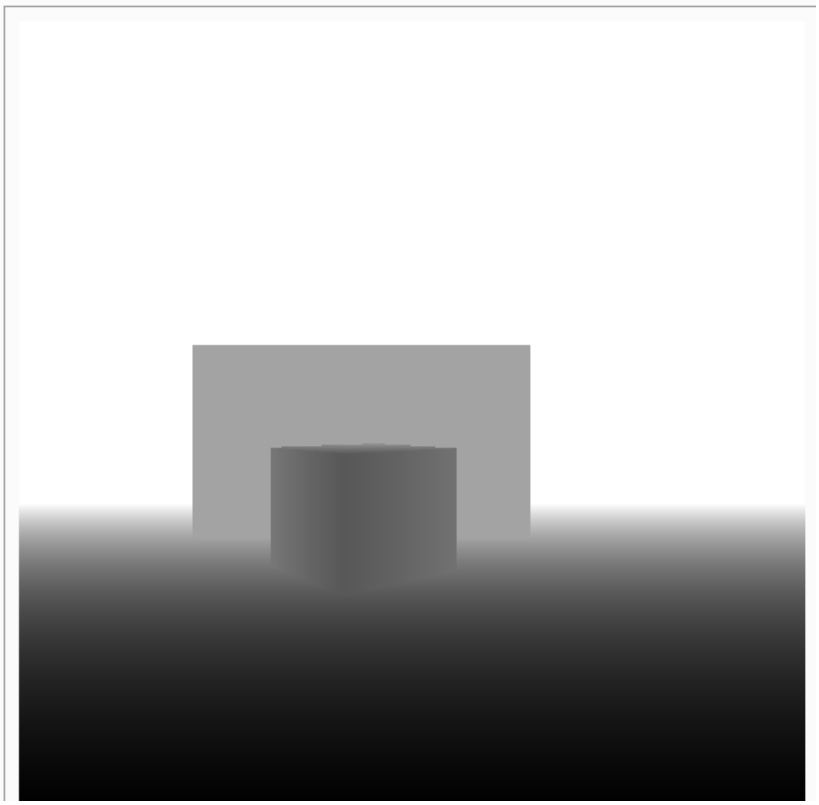

Figura 4.8: Un objeto detrás de otro

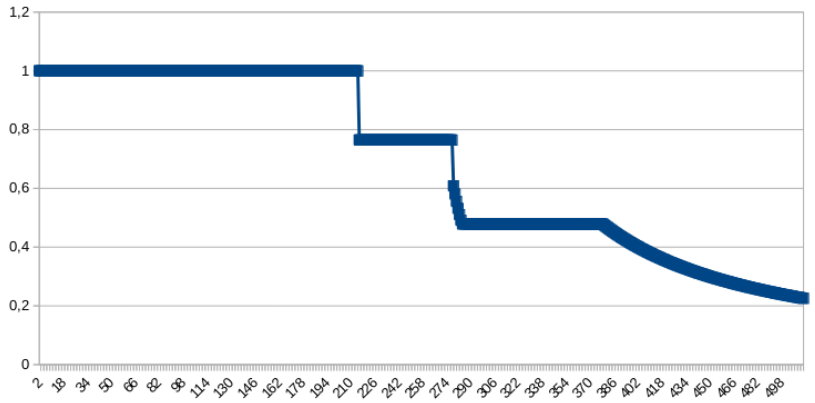

Figura 4.9: Gráfica de valores de la columna central de la matriz de la imagen de la Figura 4.8

4.2.5. Conclusiones del estudio de imágenes

Tras el estudio de las imágenes anteriores se podría concluir que los objetos se pueden detectar si consideramos dos píxeles con igual valor. Como quiera que se está tratando de cálculos en números reales (*double*) y en determinadas circunstancias pudiera ocurrir que no se encuentre dos valores consecutivos iguales, pero sí muy próximos para un objeto. Por ello, se cambia el criterio de dos píxeles consecutivos iguales por el que el valor absoluto de la diferencia de dos píxeles consecutivos sea menor de 0.0001 (valor experimental). Si se considera la cámara RealSense D435 de 1280 $\times$ 720 píxeles (Depth Output Resolution: Up to 1280 $\times$ 720; Depth Field of View (FOV): 86° $\times$ 57° $\pm$3°) y atendido a la Figura 4.10 y en relación con el plano vertical, si dividimos la altura de dicho plano por el número de pixeles y lo multiplicamos por dos resulta un valor de 15 cm para un objeto situado a 5 metros de la cámara. Un caso especial es el fondo, es decir, más allá del "far clipping plane" considerándolo un objeto de valor 1.0 o máxima distancia de visualización. Esta conclusión es dependiente de la resolución de la imagen, pero esto no es un problema para la cámara considerada. Luego, ya tenemos tres premisas para el posible algoritmo:

1. Recorrer la matriz por columnas.

2. Detectar objetos mediante el valor absoluto de la diferencia de dos píxeles consecutivos sea menor de 0.0001.

3. En caso de detectar varios pares de píxeles en una columna, se toma el de menor valor.

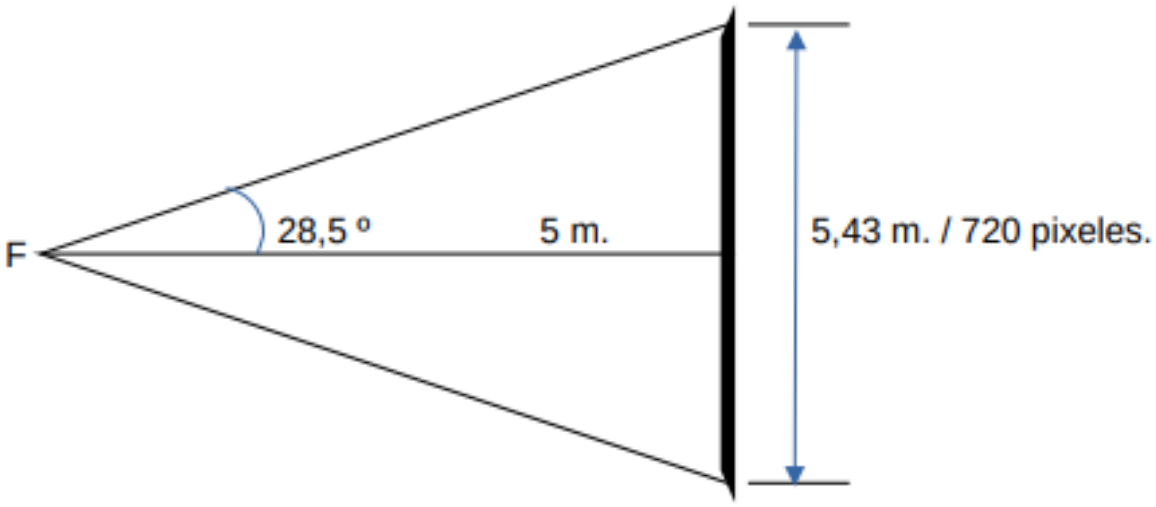

Figura 4.10: Plano vertical de la cámara

Capítulo 5

Generación del polígono estrellado

Toda vez que se ha conseguido un vector con las distancias a los objetos presentes en la imagen, se ha de convertir esas distancias en puntos del polígono estrellado. Pero en el trascurso de los ensayos realizados se manifiesta el problema, que se expone en el siguiente apartado, de utilizar una sola cámara de mapa de profundidad. Al utilizar dos cámaras se soluciona el problema y además se cubre al final un sector circular de 172° cercano al sector del plano de barrido de la medida de rango láser que, como se sabe, es de 180°. Como quiera que el vector de distancias están referidas al plano de visión de cada cámara se ha de referenciar esas distancias al origen de coordenadas locales del robot. Así, emulamos los haces de la medida de rango láser. Realizada esta transformación, se observa que, sin obstáculos, resultan dos triángulos isósceles con un lado común que no son adecuados para calcular el centro de áreas ya que la función de densidad de área no es constante. Dado que la medida de rango láser es un sector circular donde la densidad de área es constante, sugiere acotar los triángulos a un sector circular. Dicho de otra manera, se soluciona el problema acotando las medidas a un sector circular con los cálculos que se exponen más abajo. Ahora sí, se puede calcular el centro de áreas del sector circular así generado. Pero aún no se ha aprovechado toda la información que genera una cámara de mapa de profundidad. Si bien se ha generado el polígono teniendo en cuenta la distancia de cualquier objeto a cualquier altura. Aquí está la clave, la altura sobre el suelo de los objetos. Efectivamente se puede extraer esta información de la cámara. Utilizando esta información, el robot puede navegar por debajo de un obstáculo que esté unos centímetros (por seguridad) por encima del plano horizontal que contiene el eje de visión de la cámara. Con esta información ya no se generaría una superficie sino un volumen, se estaría en 3D. Para un "mundo" sobre el suelo no se tiene la necesidad de pasar a un 3D real, es suficiente con lo que se ha denominado acotado-3D. Ni si quiera hará falta calcular la altura. Con descartar toda la información del semiplano superior de visión (menos algunas filas por seguridad, unos centímetros por encima de la cámara) es suficiente.

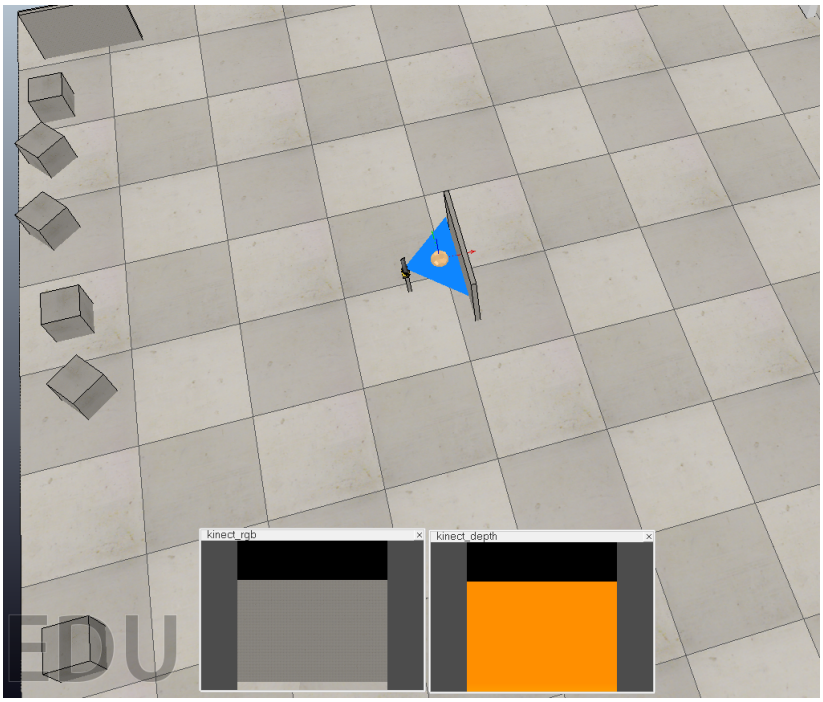

Figura 5.1: Polígono generado por una cámara

5.1. Necesidad de dos cámaras

Para acercarnos al ángulo de barrido del láser, que recuérdese son 180º, una cámara del modelo elegido sólo cubre 86º. Además, se tendría el problema de que el centro de áreas pertenece siempre al polígono para objetos cercanos a la cámara y no de anchura excesiva. La Figura 5.1 ilustra el problema.

Para solucionar la situación anterior, es fácil acoplar dos cámaras, la cámara izquierda orientada (eje z) $+86/2 = 43°$ y la derecha orientada $-43°$con respecto al eje O-x. La siguiente Figura 5.2 explicita esta solución. Ahora el centro de áreas (punto) del polígono total no pertenece a dicho polígono para el mismo objeto de la Figura 5.1 anterior frente a la cámara.

Como conclusión sería necesario utilizar dos cámaras de la manera que se ha descrito.

5.2. Del vector de distancias a los puntos del polígono estrellado

Al utilizar dos cámaras, el vector de distancias será la concatenación de los vectores de ambas cámaras. En el caso de la utilización del láser, el polígono generado sin obstáculos es un sector circular. En el caso de dos cámaras es semejante a dos triángulos isósceles con un lado común (el eje OX). Ver Figura 5.3.

Cada punto del plano de corte de cada cámara (trazo grueso en la Figura 5.3) se corres-

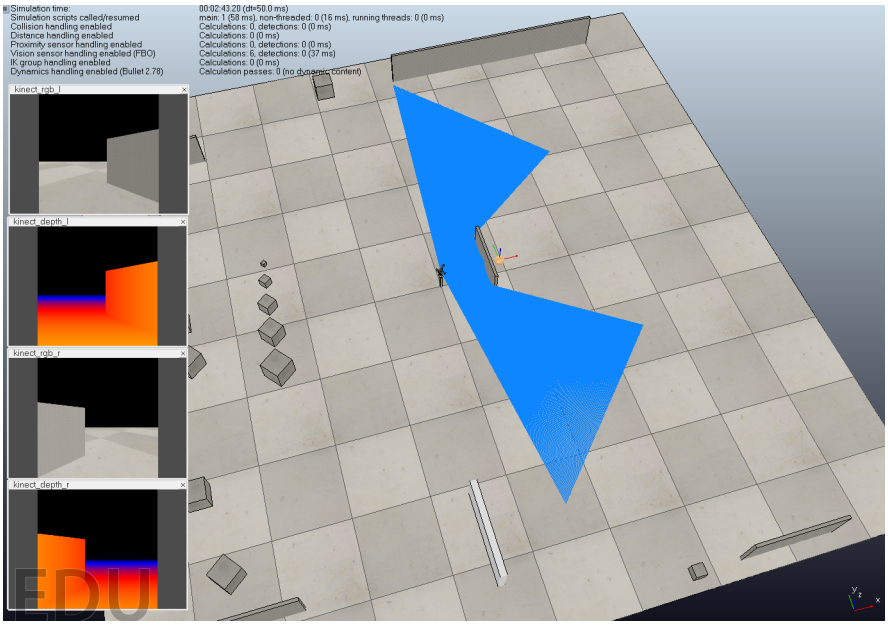

Figura 5.2: Polígono generado por dos cámaras

ponde con cada píxel de resolución horizontal, en los experimentos 512 píxeles. Dado que cada cámara tiene un determinado ángulo de visión horizontal o perspective angle (recuérdese, para la RealSense D435 Depth Field of View (FOV): 86° × 57° ±3°) se puede establecer la siguiente relación:

$$degrees\_by\_pixel = {}^{perpective\_angle}/_{(camera\_number\_pixels-1)} \tag{5.1}$$

Si se recorre el vector de distancias cada pixel corresponderá a unos determinados ψ radianes. Así las coordenadas de los puntos x e y serán:

$$d = profundidad \tag{5.2}$$

$$h = \frac{d}{\cos\psi} \tag{5.3}$$

$$x = h \times \cos\psi \tag{5.4}$$

$$y = h \times \sin\psi \tag{5.5}$$

La Figura 5.4, pretende aclarar este cálculo. Así compuesto el polígono como una sucesión de puntos se ha de tener en cuenta que los puntos se sucedan en sentido de las aguas del reloj, así como el polígono generado sea cerrado, es decir, el primer punto del polígono (el origen de coordenadas) coincida con el primero, que será este mismo punto y así facilitar el cálculo del centro de áreas.

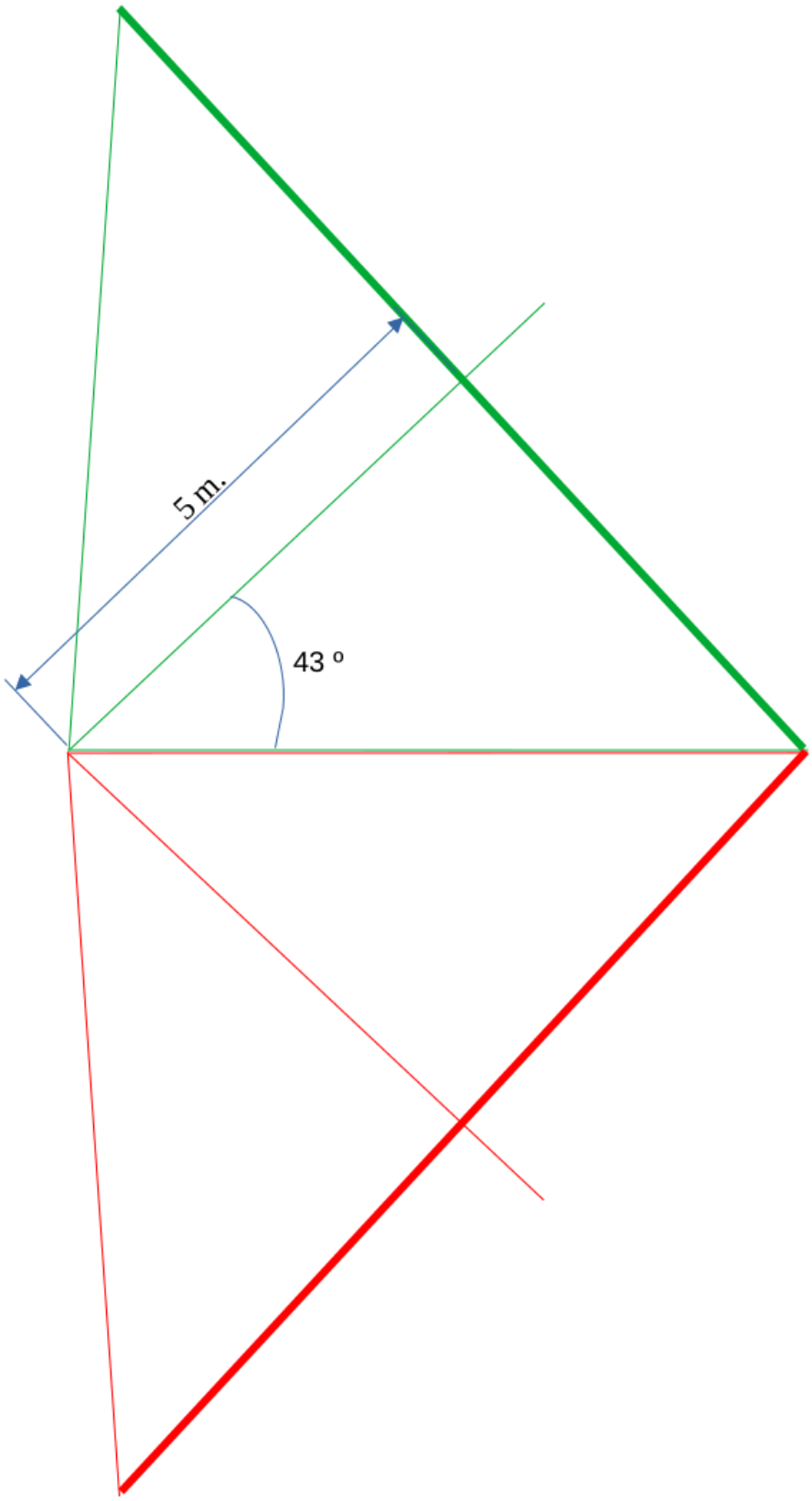

Figura 5.3: Polígono generado por dos cámaras (sin obstáculos)

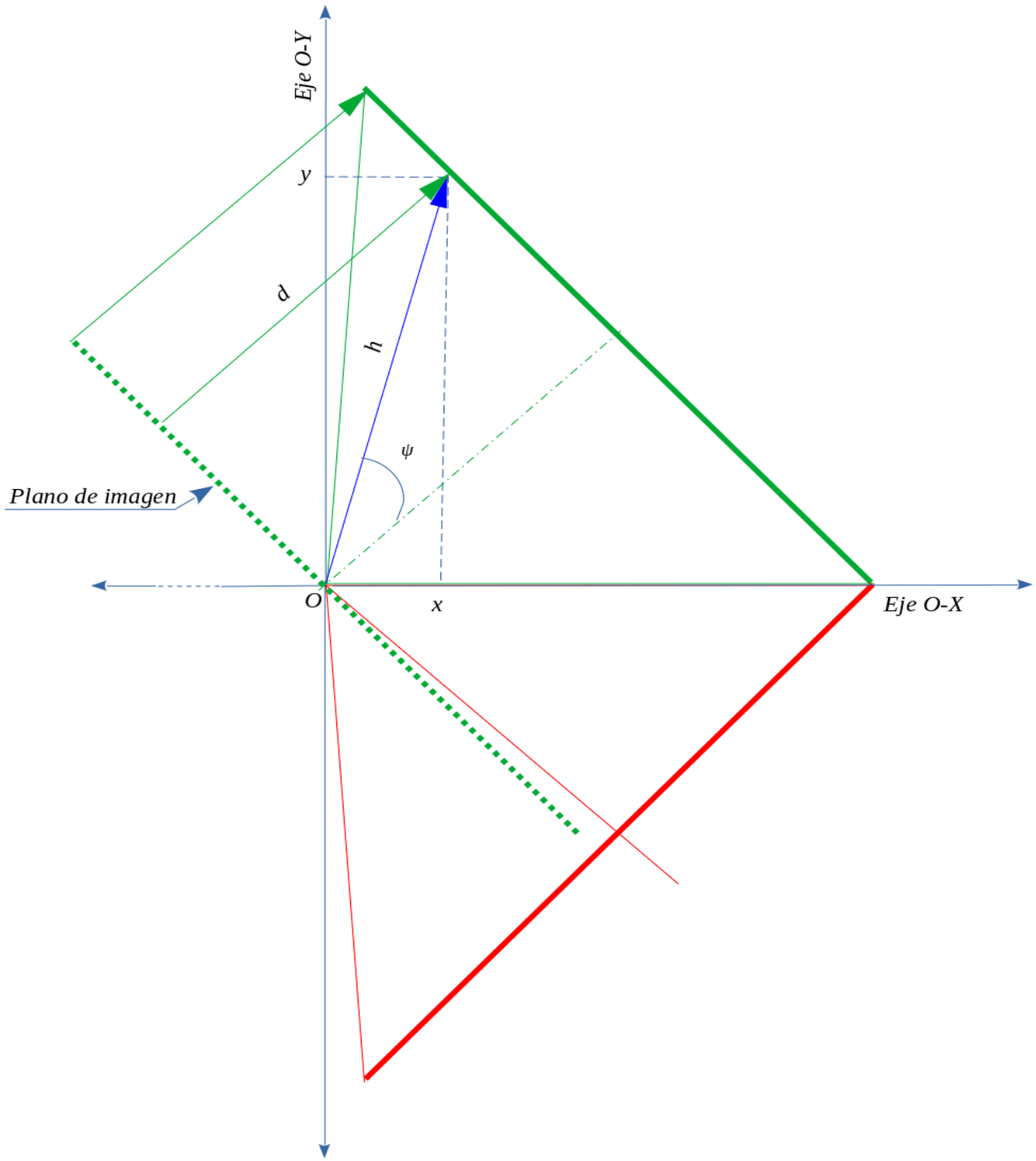

Figura 5.4: Cálculo de un punto del polígono

Un parámetro dinámico del polígono será su amplitud y desde este parámetro se calculará la máxima distancia que se han de considerar los objetos vistos por la cámara.

Generado el polígono se calcula su centro de áreas equivalente en dos dimensiones al centro de masas o de gravedad. La fórmula en coordenadas cartesianas para el caso de un polígono de n vértices $\{(x_j, y_j)\}$ es:

$$x_{CA} = \frac{1}{6A}\sum_{j=1}^{n-1}\left(x_j y_{j+1} - x_{j+1}y_j\right)\left(x_j + x_{j+1}\right) f\left(x_j, y_j, x_{j+1}y_{j+1}\right) \tag{5.6}$$

$$y_{CA} = \frac{1}{6A}\sum_{j=1}^{n-1}\left(x_j y_{j+1} - x_{j+1}y_j\right)\left(y_j + y_{j+1}\right) f\left(x_j, y_j, x_{j+1}y_{j+1}\right) \tag{5.7}$$

donde A es el área de polígono y f es la función de densidad de área. En el polígono generado por las cámaras la densidad de área no es constante. Para explicarlo véase la Figura 5.5. Se ha dividido el polígono superior de 10 en 10 grados desde el eje OX. Lógicamente la última división tiene 6 grados. Se puede apreciar que la división marcada con 1 tiene más área que la marcada con 2. La solución es generar un polígono en forma de sector circular como si fuera el generado mediante medidas de rango láser como muestra dicha Figura 5.6.

La teoría de la navegación por centro de áreas indica que, en función de la amplitud del sector del polígono estrellado se ha de modificar la distancia máxima a la que se sitúa el plano de visión. Si el centro de áreas queda fuera del polígono (punto inaccesible) se produce un *split point* y se divide el polígono por ese punto y como consecuencia se reducirá su amplitud ($angle\_width$) y se debe modificar la máxima distancia ($max\_distance$):

$$max\_distance = \frac{far\_clipping\_plane \times angle\_width}{2 \times perspetive\_angle} \tag{5.8}$$

siendo $far\_clipping\_plane$ la máxima distancia a la que la cámara puede visionar un objeto y $perspective\_angle$ el ángulo de visión de una cámara. En los experimentos 5 m. y 86 grados respectivamente.

Ahora bien, para que el polígono esté contenido en un sector circular de radio $max\_distance$ se ha de realizar los siguientes cálculos:

$$\psi = degrees\_by\_pixel \times pixel_n \tag{5.9}$$

$$x\_max = max\_distance \times \cos\psi \tag{5.10}$$

$$y\_max = max\_distance \times \sin\psi \tag{5.11}$$

La distancia d a un objeto calculada desde el vector de distancias $distances\_vector$:

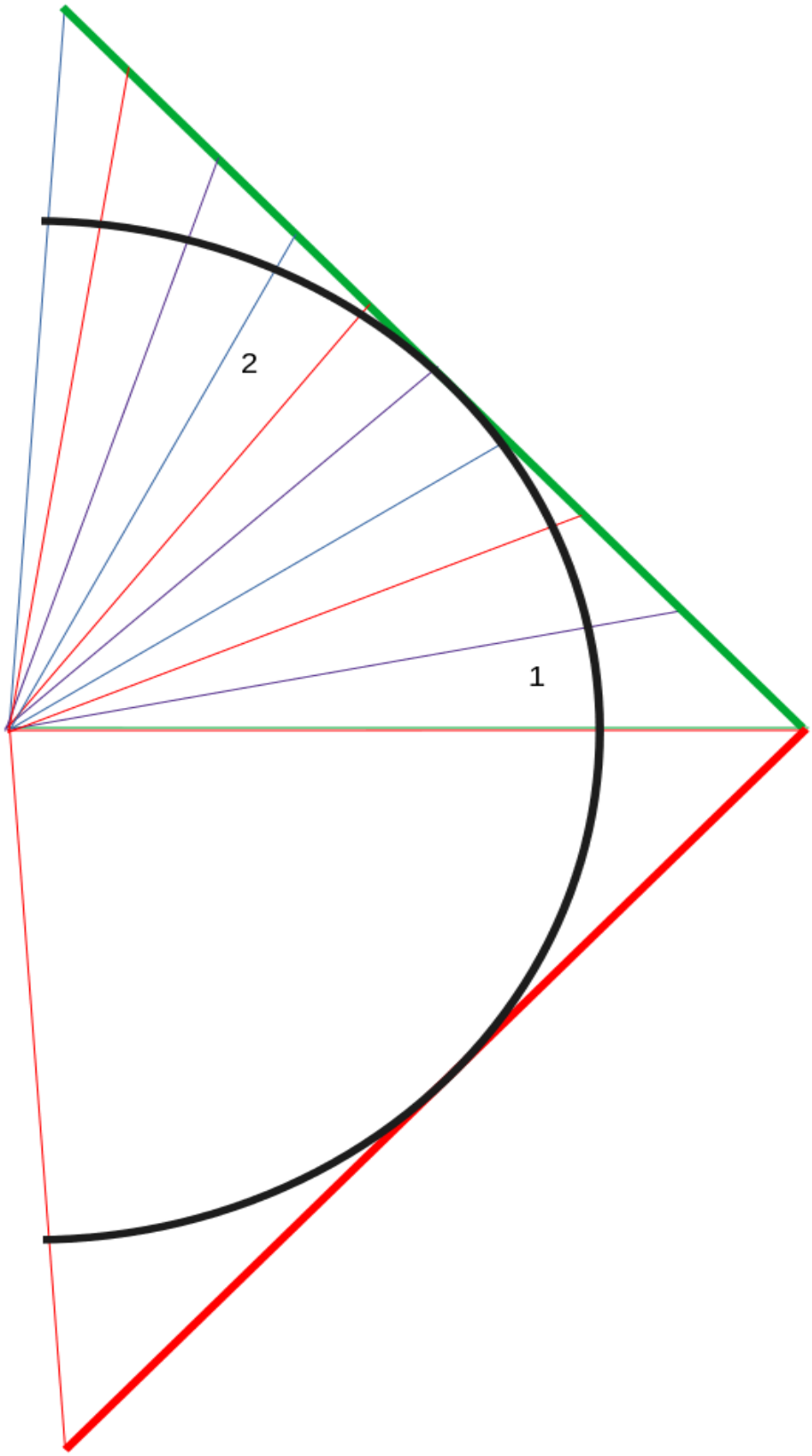

Figura 5.5: Necesidad de cambio en la figura del polígono generado

$$d = \frac{far\_clipping\_plane \times (distances\_vector\,[pixel_n] + min\_val)}{(max\_val + min\_val)}$$

donde max_val y min_val son los valores máximo y mínimo del rango de la cámara de mapa de profundidad. En los experimentos 1.0 y 0.0 respectivamente.

Las coordenadas de un punto del polígono (ver Figura 5.4):

$$d = profundidad \tag{5.12}$$

$$h = \frac{d}{\cos\psi} \tag{5.13}$$

$$x = h \times \cos\psi \tag{5.14}$$

$$y = h \times \sin\psi \tag{5.15}$$

$$x\_max = far\_clipping\_plane \times \cos\psi \tag{5.16}$$

y

$$y\_max = far\_clipping\_plane \times \sin\psi \tag{5.17}$$

$$circular\_sector\_max\_distance = \sqrt{x\_max^2 + y\_max^2} \tag{5.18}$$

$$to\_object\_distance = \sqrt{x^2 + y^2} \tag{5.19}$$

Las coordenadas de un punto (p_x, p_y) serán:

$$p_x = x \tag{5.20}$$

$$p_y = y \tag{5.21}$$

si $to\_object\_distance \leq circular\_sector\_max\_distance$ y

$$p_x = x\_max \tag{5.22}$$

$$p_y = y\_max \tag{5.23}$$

si $to\_object\_distance > circular\_sector\_max\_distance$.

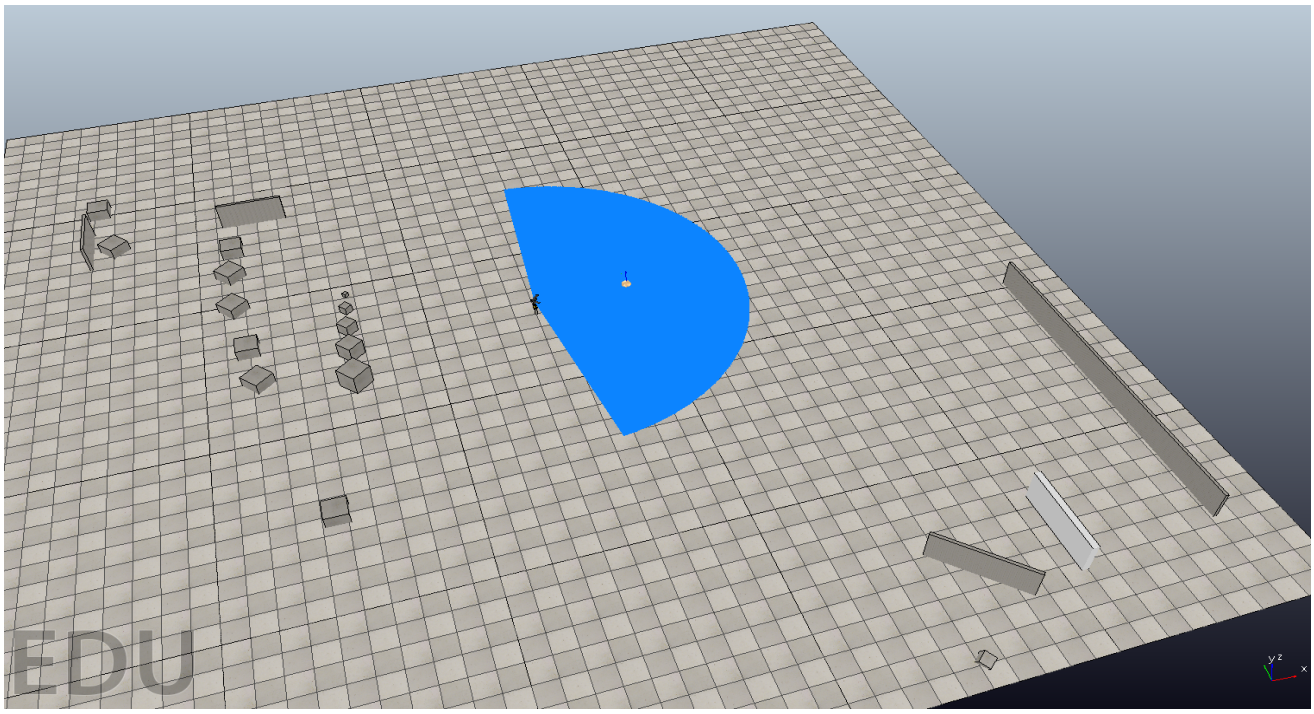

Figura 5.6: Polígono emulación láser

Ahora el polígono estrellado (emulación láser) sin obstáculos tiene la forma que muestra la Figura 5.6

En este punto hay que decir que los valores entregados por la cámara simulada son erróneos y se ha realizado una corrección en aras del funcionamiento de los experimentos. Se remite al lector al capítulo 8 donde se realiza la oportuna exposición de los problemas encontrados en el simulador.

5.3. Desde 2D ½ a acotado-3D

Aún no se ha aprovechado toda la información que genera una cámara de mapa de profundidad. Si bien se ha generado el polígono teniendo en cuenta la distancia de cualquier objeto a cualquier altura. Aquí está la clave, la altura sobre el suelo de los objetos. Efectivamente se puede extraer esta información de la cámara. Utilizando esta información, el robot puede navegar por debajo de un obstáculo que esté unos centímetros (por seguridad) por encima del plano horizontal que contiene el eje de visión de la cámara. Véase la Figura 5.7.

De la citada Figura 5.7 se deduce:

$$height = distance \times \tan\alpha + height\_optical\_axis\_to\_ground$$

Con esta información ya no se generaría una superficie sino un volumen, se estaría en 3D. Para un “mundo” sobre el suelo no se tiene la necesidad de pasar a un 3D real, es suficiente con

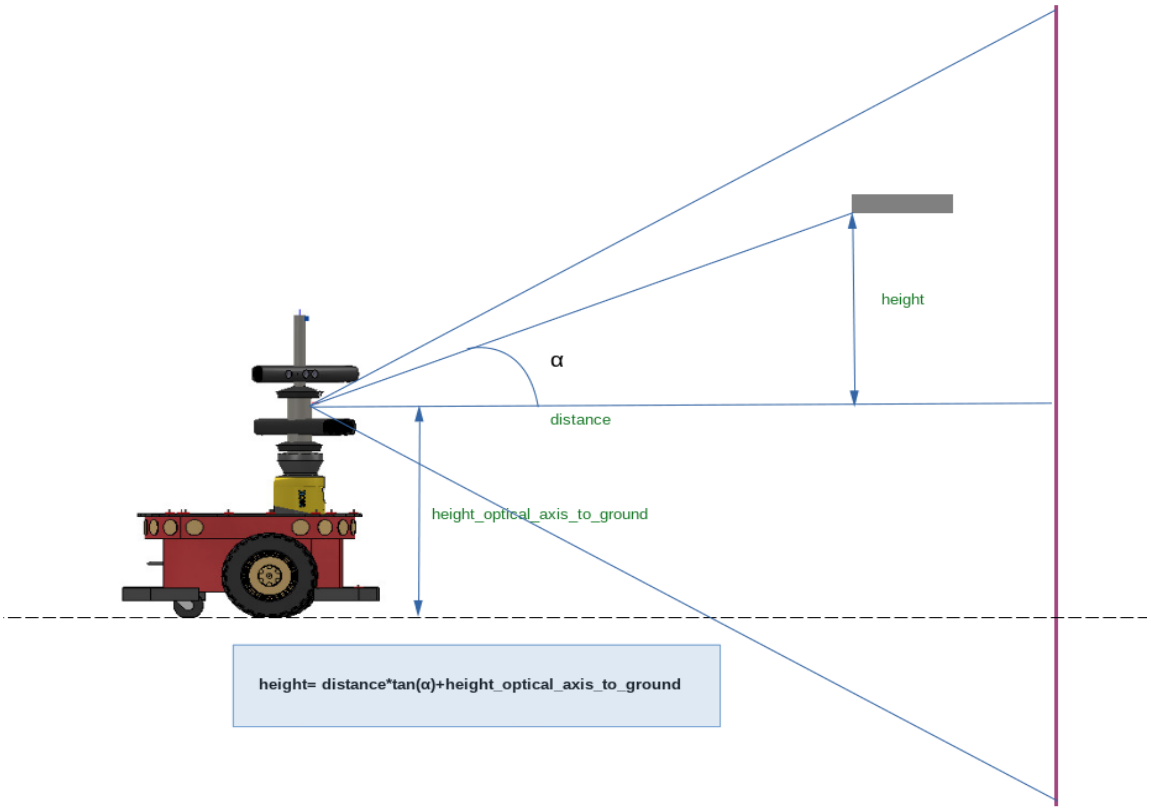

Figura 5.7: Cálculo de la altura de los objetos

lo que se ha denominado acotado-3D. Ni si quiera hará falta calcular la altura. Con descartar toda la información del semiplano superior de visión (excepto algunas filas por seguridad, unos centímetros por encima de la parte superior del vehículo) es suficiente.

Capítulo 6

Control del robot

Hasta ahora se ha conseguido el conjunto de medidas que provienen de las cámaras. Para generar el polígono estrellado y controlar el movimiento del robot se ha utilizado la librería *libareacenter* del profesor D. José Manuel Cuadra Troncoso. Originalmente escrita para las medidas generadas por el láser y sensores sonar que cubren 360° de amplitud, se ha realizado una pequeña adaptación a la amplitud generada por las cámaras que, como se dijo, son 172°. También se han creado determinados métodos y variables para la comunicación con la aplicación *Centroid_navigation* para conocer el estado del polígono de avance, la distancia normalizada, el ángulo del CA, los puntos del polígono global y de avance y los polígonos generados en la librería.

Un resumen del intercambio de datos entre la aplicación *Centroid_navigation* y la librería *libareacenter* se muestra en la Figura 6.1 La caja ***Robot data*** inicializa determinados parámetros como: *robotDiameter*, *wheelsSep*, *saplingPeriod*, *sensorMaxVal* y *adavanceAmplitude*. En términos de la librería es el ***Init Robot Data***. Luego se entra en un bucle y la primera acción del bucle es solicitar las medidas que producen las cámaras con su algoritmo asociado. Estas medidas generaran lo que se ha dado en llamar ***Global Polygon*** o Polígono Global. Con ***Velocities measurements*** se envían a la librería los velocidades actuales del robot de las ruedas izquierda y derecha que en la librería son las ***Actual velocities***. ***Goal Data*** son dos parámetros, *goalProximity* y *goalDirection* que normalmente estarán a cero; pero en el caso de alcance de objetivos serán los valores de distancia y ángulo al objetivo respectivamente. Con estos valores a cero ***Calculate Goal side*** dejará la selección del polígono de avance después de una división (*split point*) a la accesibilidad del subpolígono, a su área y a un parámetro generado aleatoriamente. Ahora bien, en el caso de la búsqueda de objetivos ***Calculate Goal side*** seleccionará el subpolígono que conduzca el robot al objetivo. En ***Calculate Advance Polygon*** se generará el polígono de avance y desde él su Centro de áreas en ***Calculate Area Center*** que se devuelve a la aplicación para convertir las coordenadas locales del Centro de áreas en coordenadas globales por ***AC from local coordinates to global*** y así representarlo

en el "mundo". ***Calculate_velocities*** calcula las velocidades de las ruedas izquierda y derecha del robot en función de la posición del CA y su ángulo con respecto al robot. Se tendrá en cuenta para este cálculo el denominado en la librería *AdvanceStatus* que son los cinco estados que puede estar el polígono de avance: *All*: el polígono de avance coincide con el global, *Right*: se ha seleccionado el polígono derecho, *Left*: se ha seleccionado el polígono izquierdo, *Conservative* y *Lost* son estados especiales que tienen que ver si el robot está estancado o el polígono no se ha podido generar por errores en la medida. Las velocidades así calculadas se devuelven a la aplicación y ***Velocities_to_robot*** se encarga de llevar estos valores a las ruedas del robot. El método que se encarga de este control originalmente era un método sin parámetros. En este trabajo se han introducido dos parámetros *robotMaxVel* que normalmente tiene el valor de 0.75 m/seg. de velocidad máxima del robot y *velLimiter* cuyo valor es 0.5. Al situar estos parámetros en este método se puede experimentar con ellos desde la aplicación *Centroid-Navigation*.

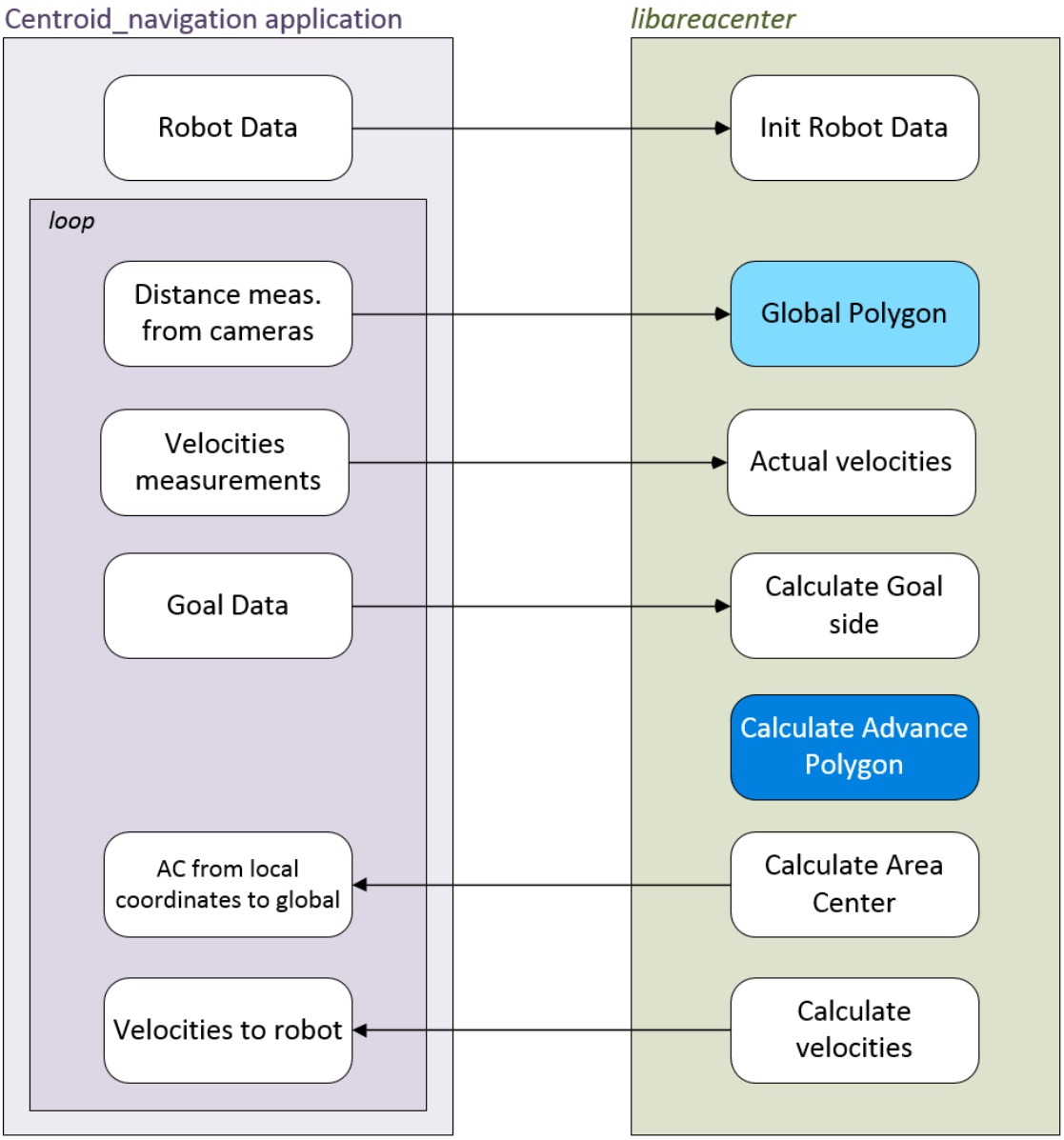

Figura 6.1: Procesamiento de datos

Capítulo 7

Aplicación Centroid navigation

Para manejar los distintos experimentos y pruebas se ha diseñado una aplicación en C++ utilizando el entorno de programación Qt Creator. En este capítulo, se hace un repaso por las diferentes ventanas de la aplicación describiendo sus partes componentes de cada una de ellas: *Ventana principal de la aplicación*, *Depth map testing. Una cámara*, *Depth map testing. Dos cámaras*, *Depth map testing. Dos cámaras con libareacenter, Pestaña Time Tagged Data* y finalmente *Error: experimento en el simulador distinto al seleccionado*. Se debe decir que aunque el aspecto de la aplicación ha sido muy parecido en el transcurso de la investigación hay un antes y un después de la integración con la librería *libareacenter.* Previo a la integración se utilizó el andamiaje suficiente para poder realizar las primeras pruebas y simulaciones. Después de integrar *libareacenter,* se eliminó parte de ese andamiaje, fundamentalmente en lo relativo al control del robot. Quedó las pruebas de una y dos cámaras, pero se añadió la prueba de dos cámaras a través de *libareacenter.* Esta aplicación se apoya en una base de datos para guardar los parámetros de cada experimento: Velocidad, Limitador de velocidad, ángulos mínimo y máximo de actuación del robot, retardo de actuación y los píxeles por encima del eje óptico para generar el acotado-3D. Para cada experimento se corresponde a un registro de la base de datos que se apunta con la selección del número del experimento. Al final todos los experimentos tienen un valor parecido, por no decir iguales, en los distintos parámetros, pero así se permite experimentar con ellos, por ejemplo, modificando el limitador de velocidad para un experimento concreto.

7.1. Ventana principal de la aplicación

En la siguiente Figura 7.1 se muestra la ventana principal al arrancar la aplicación. En dicha Figura 7.1 se remarcan las distintas secciones o funcionalidades. En la marcada como el número 1 se aprecia de izquierda a derecha los botones de arranque y paro de la simulación. Continúa con un círculo de color amarillo a modo de "piloto". Tiene tres estados o colores, el amarillo de la figura significa que la simulación está parada, verde que están en marcha y rojo que hay

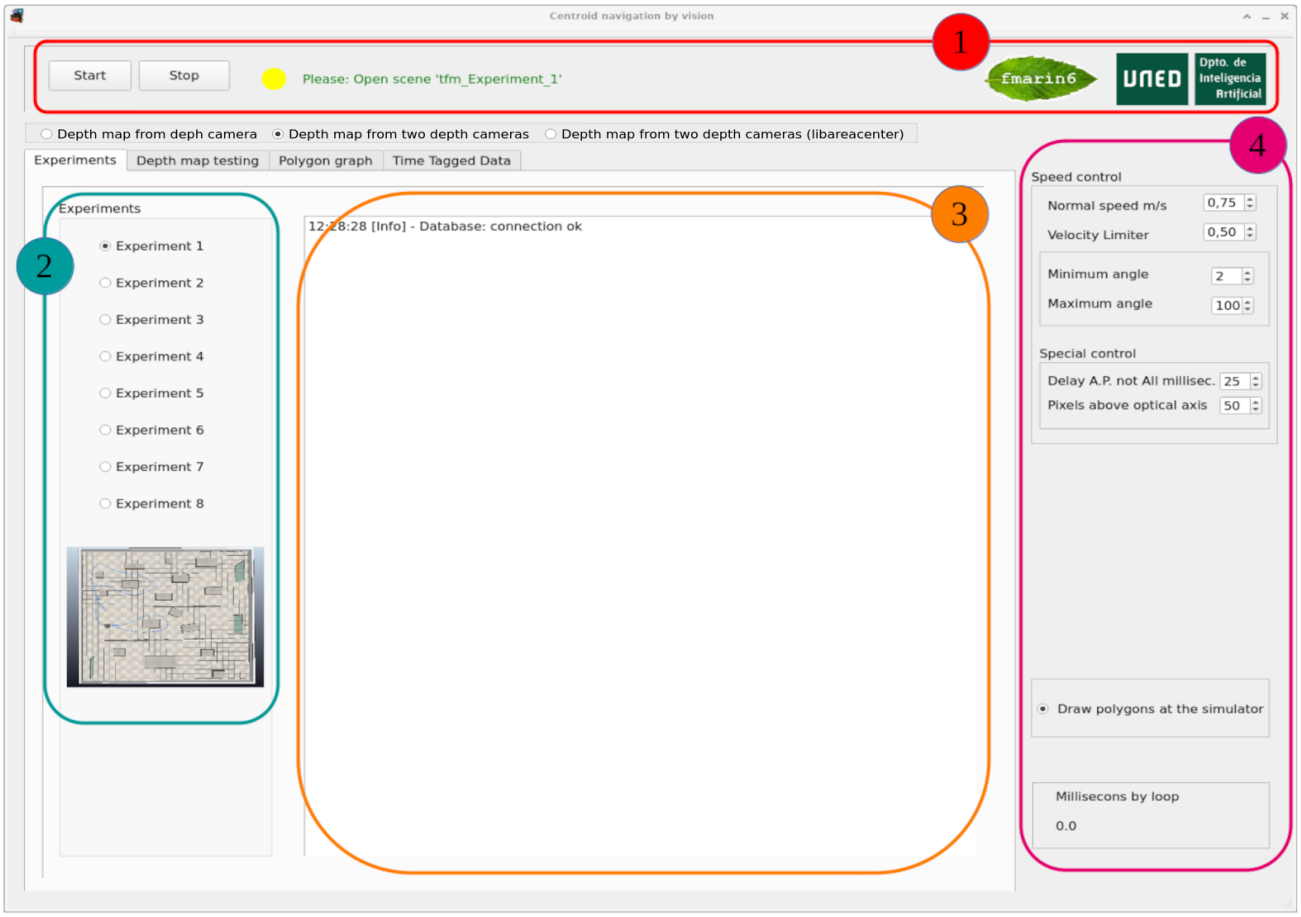

Figura 7.1: Aplicación Centroid navigation. Ventana principal

algún problema, normalmente que se pretende realizar un experimento en el simulador que no se corresponde con el seleccionado. Le sigue un texto en verde que indica el nombre del experimento a situar en el simulador CoppeliaSim. Finalmente, dos logos, el que utilicé en mi PFC y el del Departamento. En la sección numerada con el número 2 es el selector de experimentos. Se mostrará la figura del experimento seleccionado. La sección 3 en un espacio "memo" donde aparecerán los eventos que se generen en el funcionamiento de la simulación. En la presente Figura 7.1 aparece el texto 12:28:28 [Info] - Database: connection ok. La aplicación tiene una base de datos SQLite para almacenar la parametrización de cada uno de los experimentos. Por último, la sección numerada con el número 4, de arriba a abajo, se aprecia los parámetros relativos al control de velocidad y navegación del robot. Le sigue un selector a demanda si se quiere pintar o no los polígonos global y de avance en el simulador. En la parte inferior, la indicación de los milisegundos de la media de 1000 muestras del bucle de simulación.

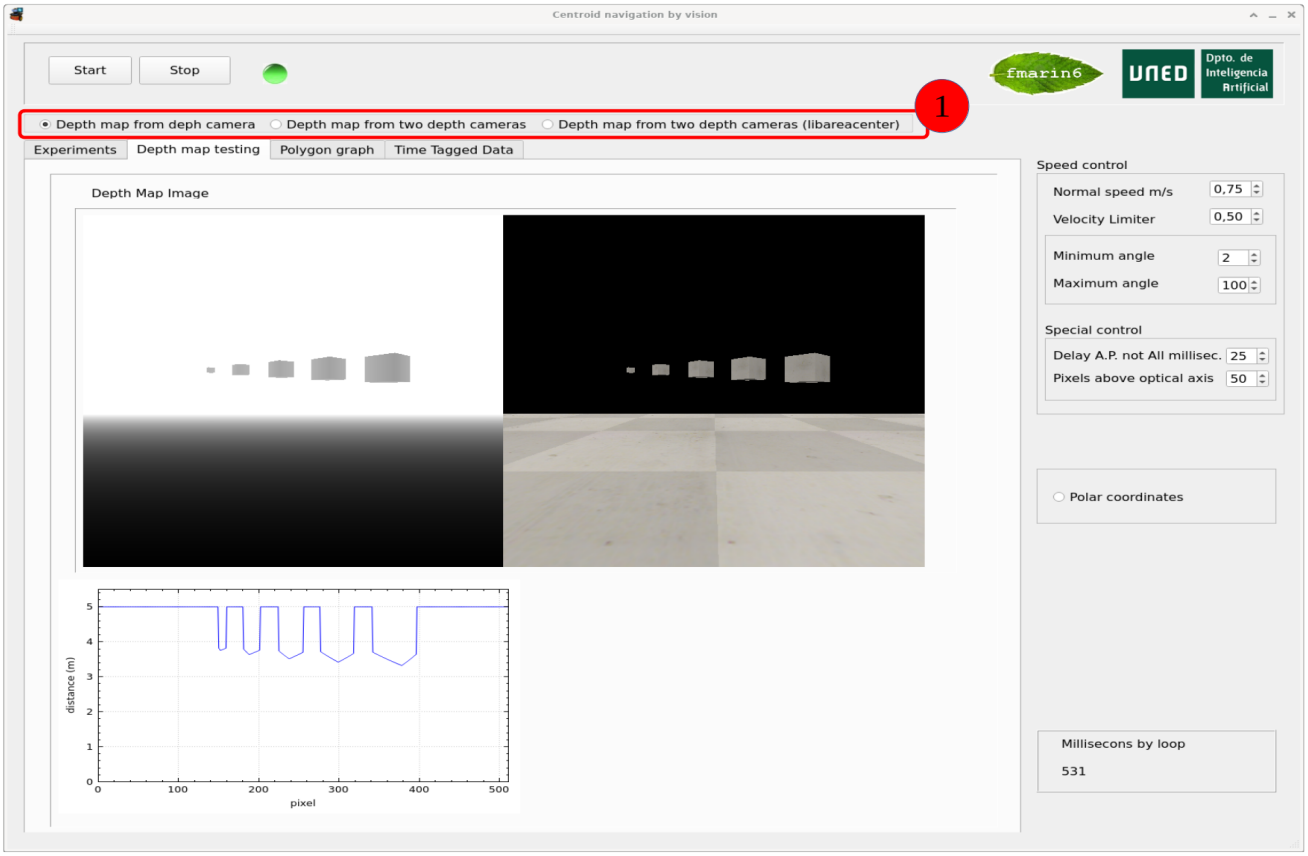

Figura 7.2: Aplicación Centroid navigation. Pestaña Depth map testing. Una cámara

7.2. Depth map testing. Una cámara.

Si se selecciona la pestaña Depth map testing se habilita tres posibilidades de comprobación, como se remarca con el número 1: Generar el depth map desde una cámara, generar el depth map desde dos cámaras y por último se puede seleccionar la prueba a través de la librería *libareacenter*. En la Figura 7.2 se ha seleccionado una cámara.

En el cuerpo de la pestaña se distinguen tres zonas. La superior izquierda es la imagen que genera la cámara de mapa de profundidad reseñada con el rótulo *Depth Map Image*. La parte superior derecha es la misma imagen, pero en RGB. Abajo a la izquierda una gráfica con la medida de las distancias a los objetos en metros.

Hacer notar que en la parte derecha hay un botón selector “Polar coordinates”. Si se selecciona se calcula y dibuja en el simulador un sector circular como se comentó en la sección 5.2. El valor 531 milisegundos obedece a 31 milisegundos de procesamiento y 500 milisegundos de retardo para no saturar la parte gráfica de la aplicación.

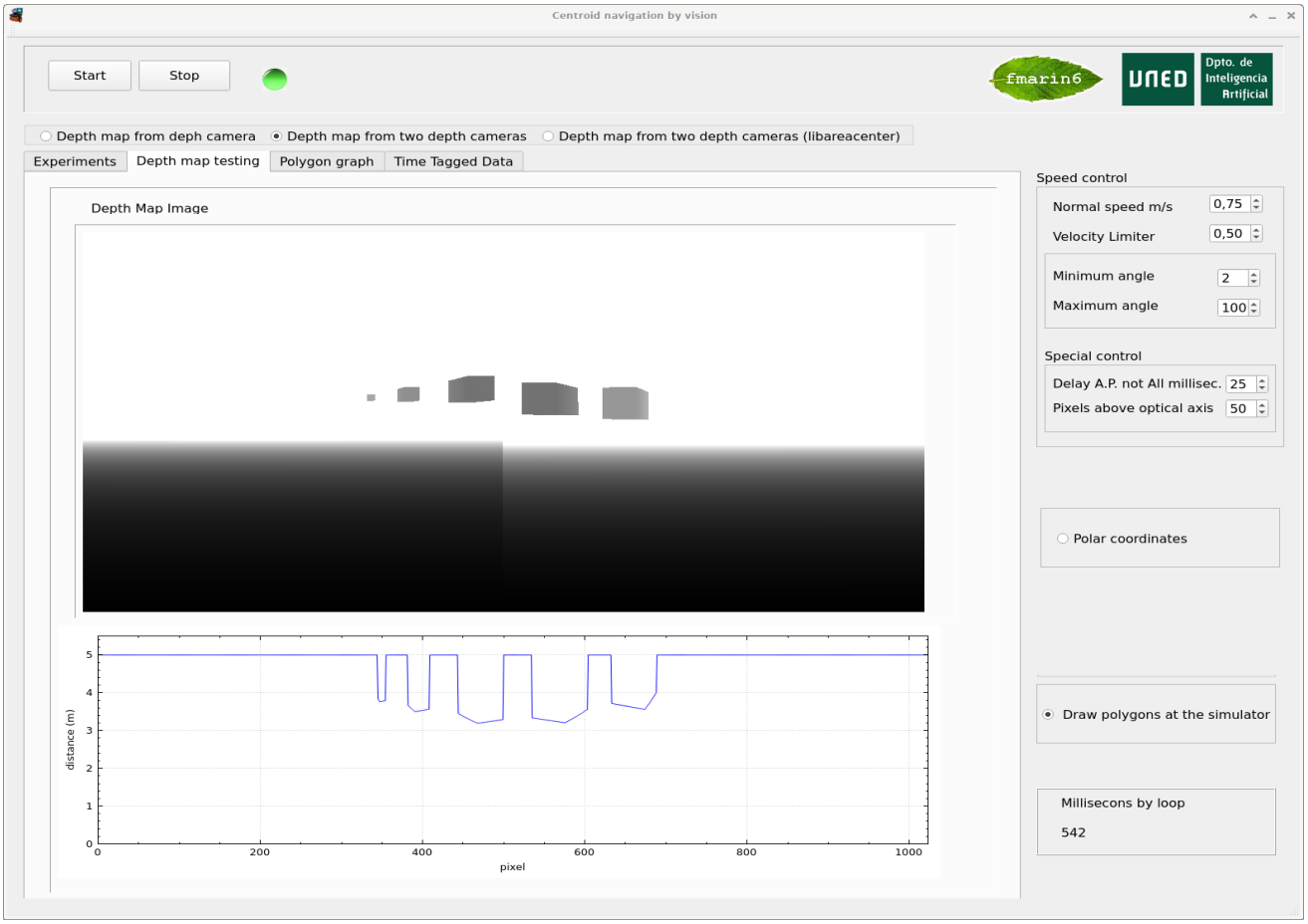

Figura 7.3: Aplicación Centroid navigation. Pestaña Depth map testing. Dos cámaras

7.3. Depth map testing. Dos cámaras.

En la Figura 7.3 se ha seleccionado dos cámaras. En el cuerpo de la pestaña se distinguen dos zonas. La superior son las dos imágenes que generan las dos cámaras de mapa de profundidad reseñada con el rótulo *Depth Map Image* y que corresponden a la cámara izquierda y cámara derecha según se presentan en la Figura 7.4. Abajo a una gráfica con la medida de las distancias a los objetos en metros extraída del denominado vector de distancias.

7.4. Depth map testing. Dos cámaras con *libareacenter*

En la siguiente Figura 7.4 se ha seleccionado dos cámaras (*libareacenter*). En la parte derecha hay un nuevo control sobre graficar los polígonos, el global y el de avance que genera la librería *libareacenter*.

En el cuerpo de la pestaña se distinguen dos zonas. La superior son las dos imágenes que generan las dos cámaras de mapa de profundidad reseñada con el rótulo *Depth Map Image* y que corresponden a la cámara izquierda y cámara derecha según se presentan en la Figura

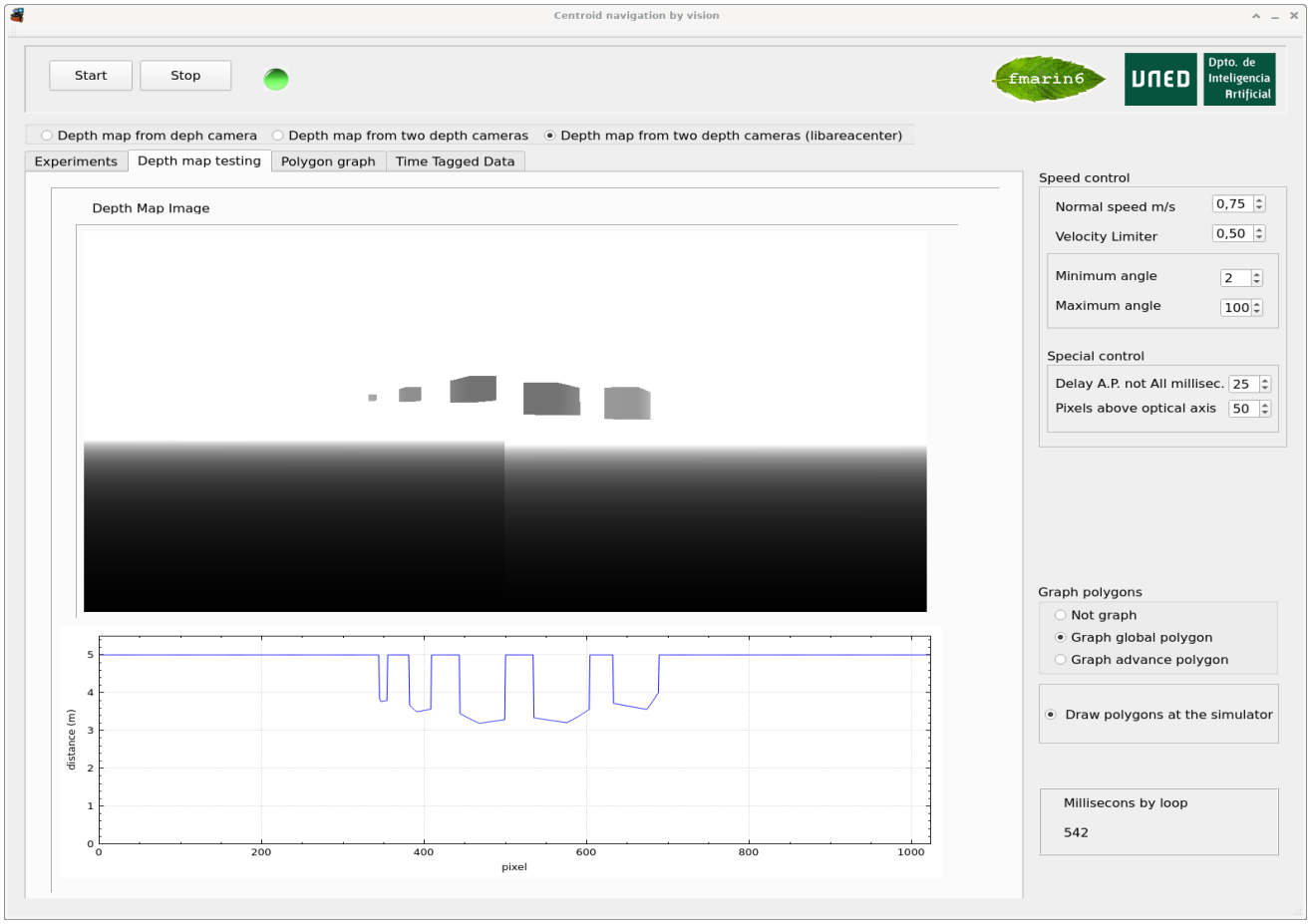

Figura 7.4: Aplicación Centroid navigation. Pestaña Depth map testing. Dos cámaras libareacenter

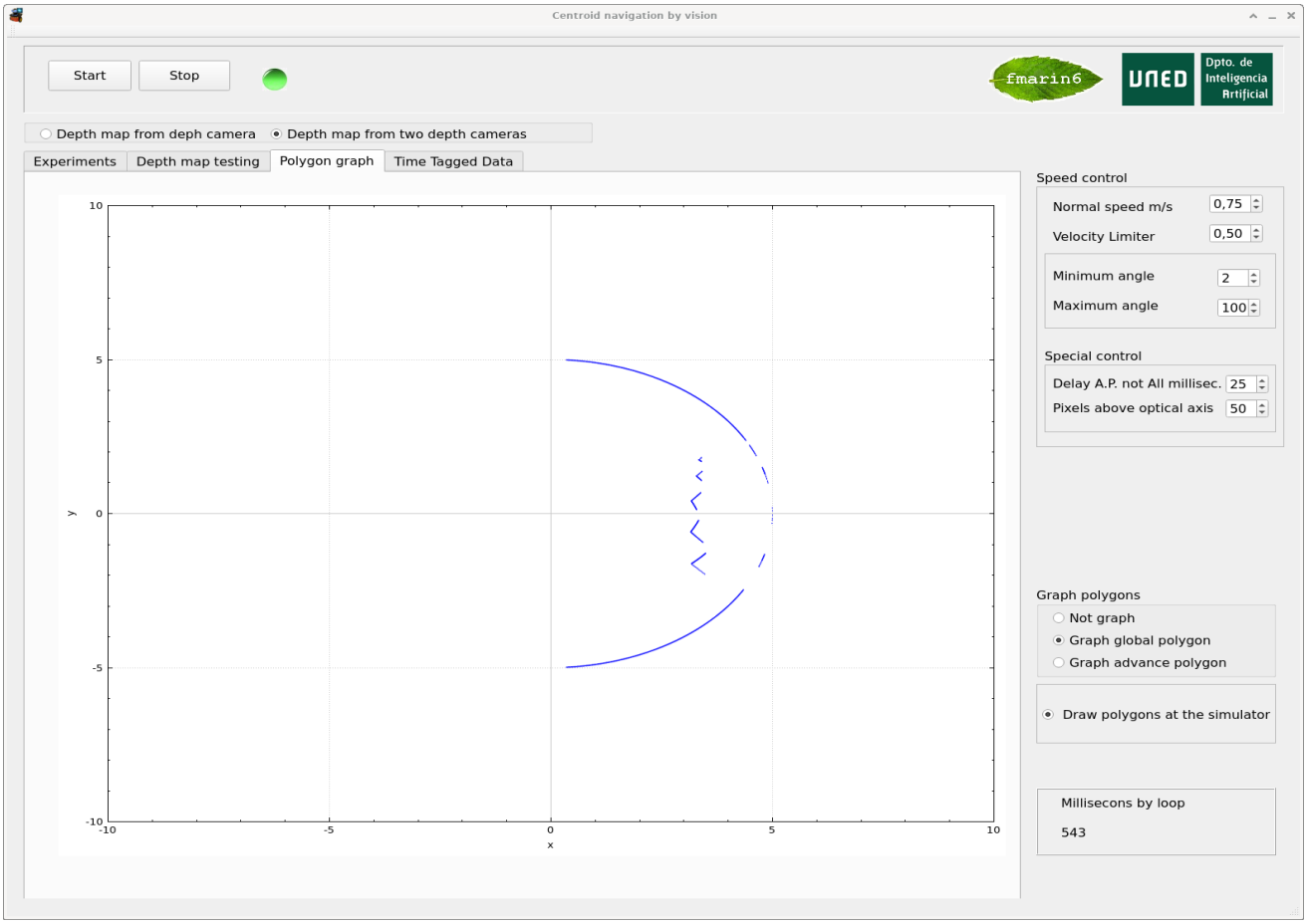

Figura 7.5: Aplicación Centroid navigation. Pestaña Polygon graph

7.4. Abajo a una gráfica con la medida de las distancias a los objetos en metros extraída del denominado vector de distancias.

Para visualizar un polígono se selecciona en el control de selección de polígonos y la pestaña Polygon graph. En la Figura 7.5 se puede ver el gráfico del polígono global debido a la selección realizada en Graph Polygon de la parte derecha de esta Figura 7.5 Se puede observar con facilidad el sector circular modificado por lo objetos presentes en la imagen y como la traza del sector de color azul no cruza el eje Y. Esto es debido a que sólo se cubren 172° como se manifestara en el Capítulo 5.

7.5. Pestaña Time Tagged Data

Durante la simulación se puede seleccionar la pestaña Time Tagged Data. Se puede ver en la siguiente Figura 7.6 la evolución en el tiempo de las variables velocidad del robot (Speed), la distancia normalizada a la que se sitúa el centro de áreas (AC Norm. distance) y el ángulo (AC angle) en coordenadas locales. Es habitual y de mucha utilidad en el campo de la Instrumentación y Control graficar las variables a controlar. Ello permite no solo ver su comportamiento

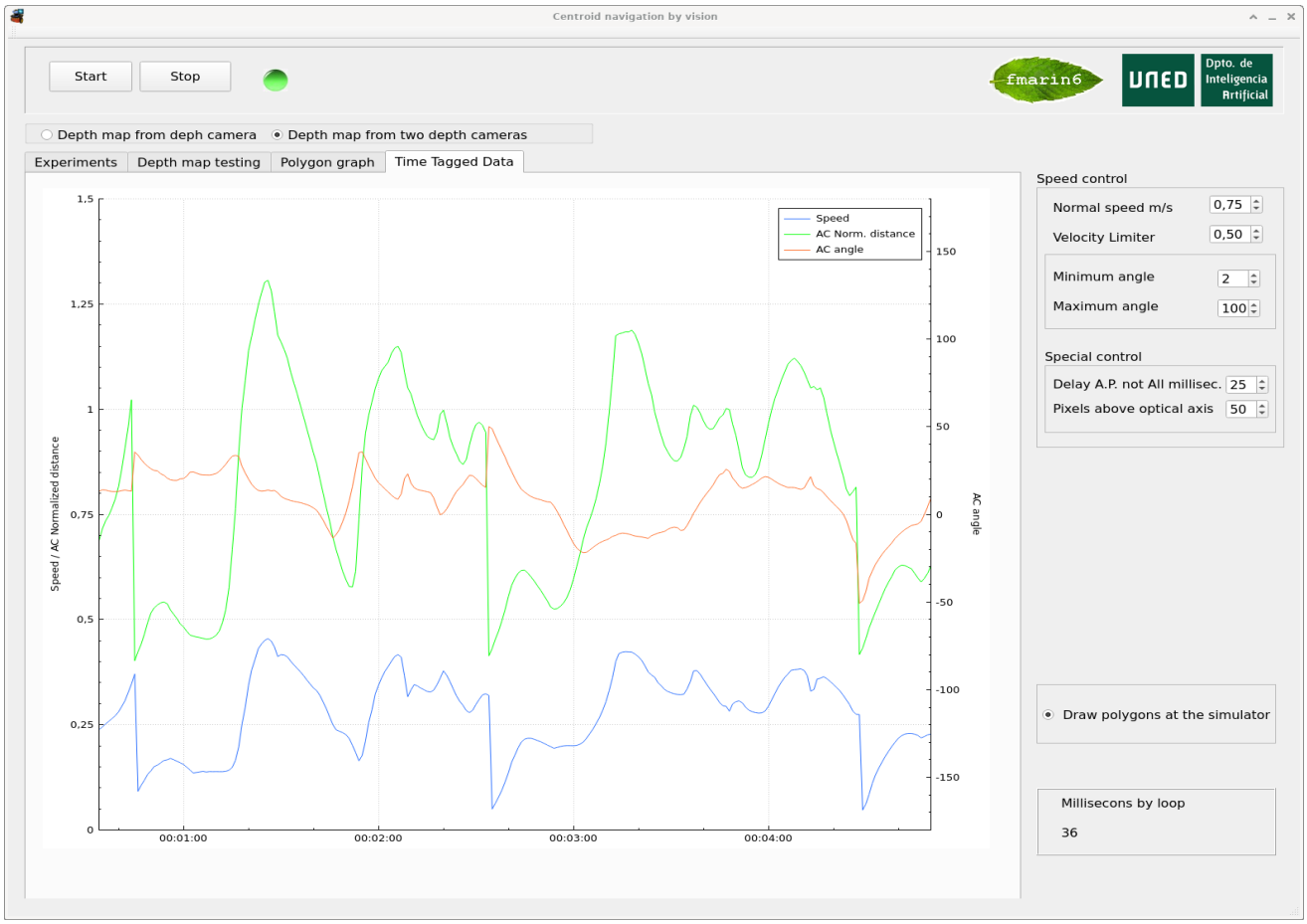

Figura 7.6: Time Tagged Data: Gráfica de tendencias

sino sugerir otras estrategias de control si se observara un control oscilatorio, por ejemplo. En este trabajo ha parecido de interés generar este tipo de gráfica con se ha denominado Time Tagged Data o datos etiquetados en el tiempo. Como se ha dicho la gráfica son tres curvas con dos escalas. En la parte izquierda la escala de la velocidad del robot y la distancia normalizada desde el robot al Centro de áreas cuyo rango es 0-1.5 metros y a la derecha la escala del ángulo del Centro de áreas cuyo rango es de -180° a +180°. En la parte inferior un eje de tiempos. Completa la gráfica un recuadro con los colores de cada variable a modo de leyenda.

7.6. Error: experimento en el simulador distinto al seleccionado

Si se selecciona un experimento y en el simulador se tiene uno distinto, la simulación no arranca. El problema suele ser normalmente que no se encuentran los manejadores (*handles*) necesarios para el experimento seleccionado. Esta situación se reporta con el "piloto" en rojo y la información que se muestra en el memo. Ver Figura 7.7. Aunque el código principal de

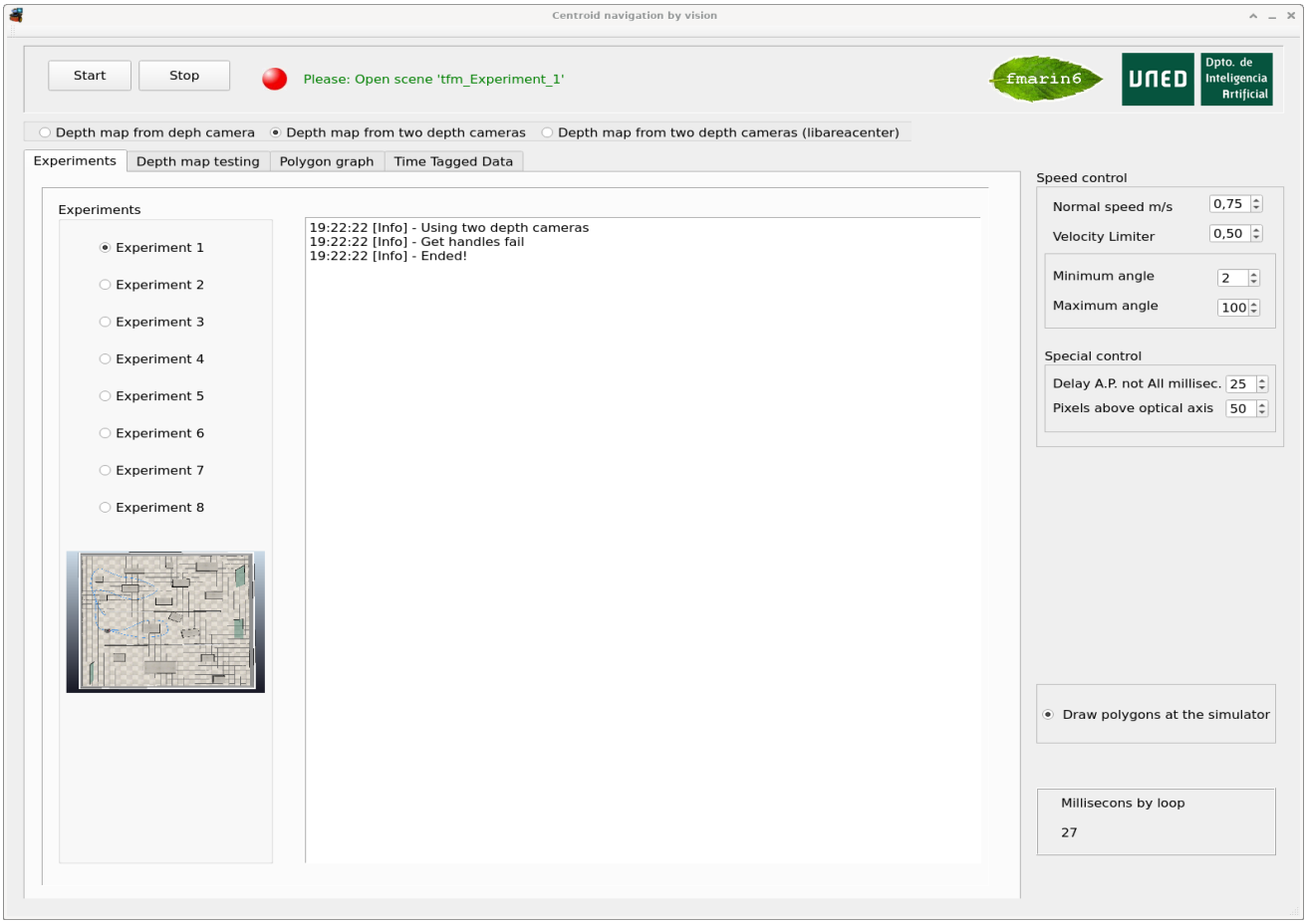

Figura 7.7: No correspondencia entre selección de experimento y experimento en el simulador

cada experimento es el mismo, cada "mundo" o escena en el simulador puede tener elementos especiales con los cuales hay que interaccionar y por ello se reporta el error. También la selección correcta del experimento introduce determinadas líneas de código (por ejemplo, el alcance de objetivos) para la consecución del mismo. Los parámetros de Normal speed, Velocity Limiter, Delay A.P. not All y Pixels above optical axis también son en dicha selección particulares de cada experimento.

Capítulo 8

Simulador CoppeliaSim

El simulador utilizado en los experimentos es el CoppeliaSim Edu, Versión 4.2.0 (rev.2) 64 bits, es una herramienta amigable y que puedo aconsejar su utilización, pero como enseña la Ingeniería de Software todas las aplicaciones tienen fallos. Según Gómez Palomo and Moraleda Gil (2020) se define fallo como el hecho de que un elemento de un programa no funciona correctamente produciendo un resultado (parcial) erróneo. A lo largo de este trabajo se ha encontrado dos fallos que se describen a continuación. El primer fallo por increíble que parezca es el retardo que se introduce en la llamada de un método de propósito general simxCallScriptFunction. Se realiza una comparativa de utilizar este método y no utilizarlo y se ve que el cambio es abismal. Se ha tenido que utilizar un metodo ad-hoc para reemplazar el método de propósito general citado. Otro fallo es lo relativo al mapa de profundidad del sensor de visión del simulador. Se constata que ante un muro circular la gráfica de distancias no es una recta como se debiera esperar sino dos curvas. Mediante Python se ha creado una cámara de mapa de profundidad en base, obviamente, a dos cámaras RGB. El resultado es concluyente, el muro circular aparece como una línea recta con un pequeño error en la parte izquierda de la traza debido a la manera que se ha utilizado para generar la cámara de mapa de profundidad. Estos errores se reportaron a Coppelia Robotics y han puesto a un técnico para solucionar estos fallos. En principio está el técnico de Coppelia trabajando sobre el fallo de las cámaras y ya ha habido un cruce de correos. En cualquier caso, estos fallos no son relevantes para este trabajo de fin de máster ya que se han solucionado de alguna manera. Ya habrá tiempo para dar una solución definitiva trabajando junto con el técnico de Coppelia Robotics.

8.1. Retardo en el método simxCallScriptFuction

El método simxCallScriptFunction es un método de propósito general que se utiliza para enviar y/o recibir datos de intercambio entre la aplicación y el simulador cuando no hay un método específico para tal intercambio. Por ejemplo, recibir la matriz de transformación del

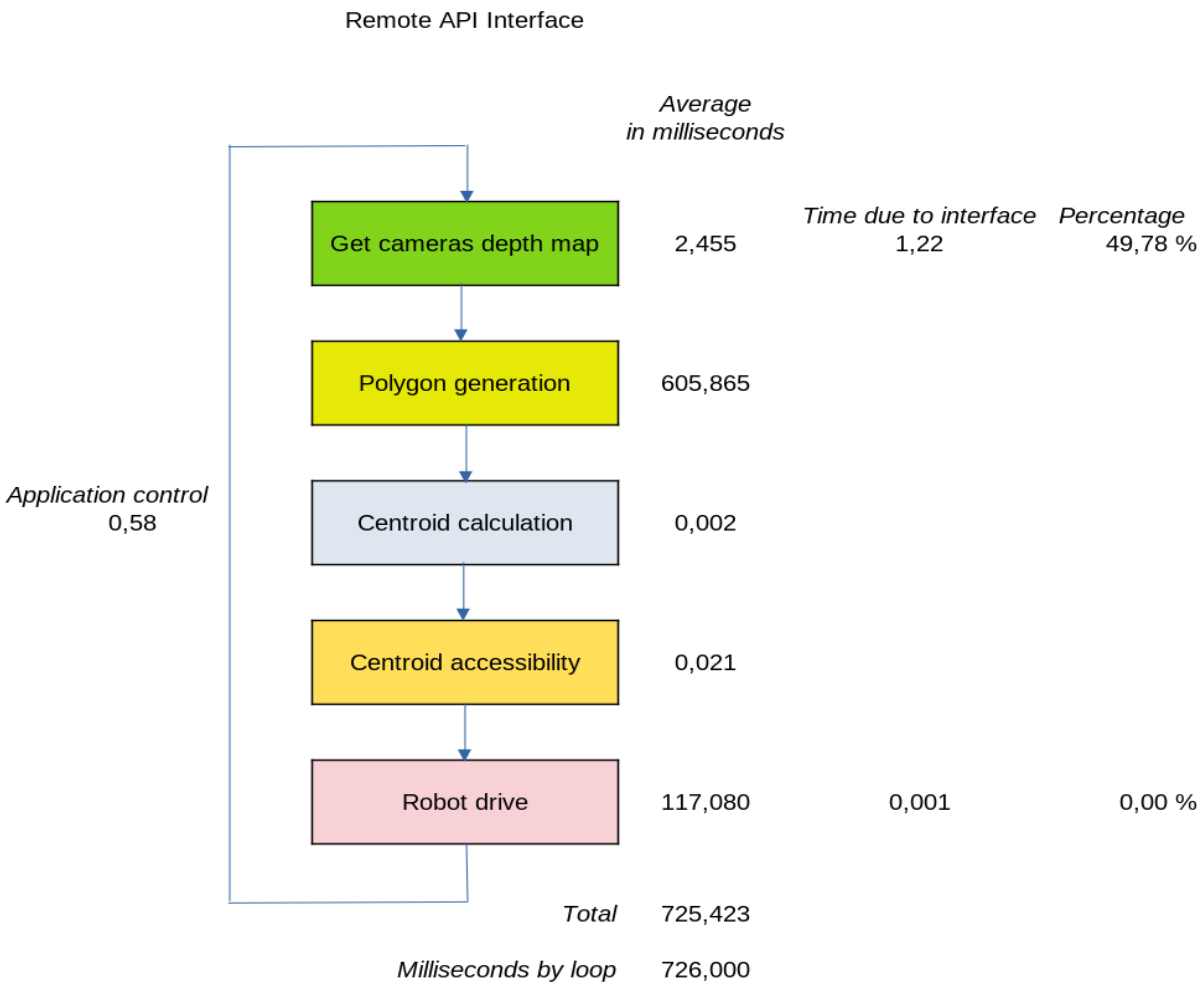

Figura 8.1: Tiempos utilizando simxCallStriptFuction

robot para convertir coordenadas locales en globales. Son 12 reales y cuesta 162 milisegundos traerlos desde el simulador a la aplicación. También, si se ha de enviar datos desde la aplicación ocurre el mismo problema, por ejemplo, enviar los puntos que constituyen las líneas que formarán el dibujo del polígono en el simulador. La Figura 8.1 muestra los tiempos en milisegundos del bucle de control de la navegación en una fase temprana de este trabajo. Es decir, sin utilizar la librería *libareacenter*.

Ante este problema, se ha utilizado un script de lua en conjunción con la parte correspondiente en la aplicación en C++ que trae los 12 reales de la matriz de transformación individualmente. Para el envío de datos para dibujar los polígonos se ha utilizado otro script lua y su contraparte en C++. En la siguiente Figura 8.2 se ve que el cambio es abismal. Nótese que se ha cambiado de milisegundos a microsegundos, de 726 milisegundos a 5,58 microsegundos.

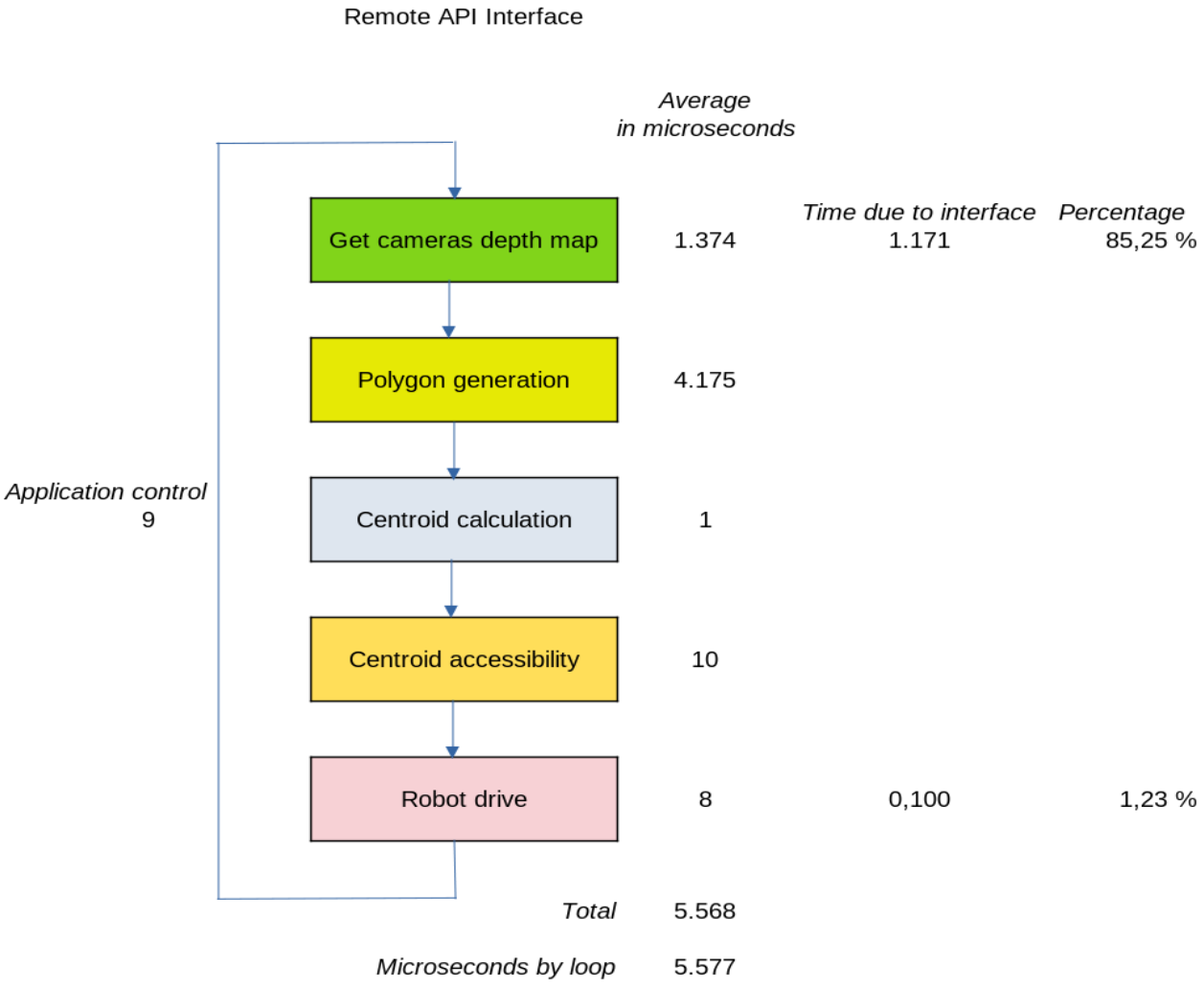

Figura 8.2: Tiempos utilizando script lua ad-hoc

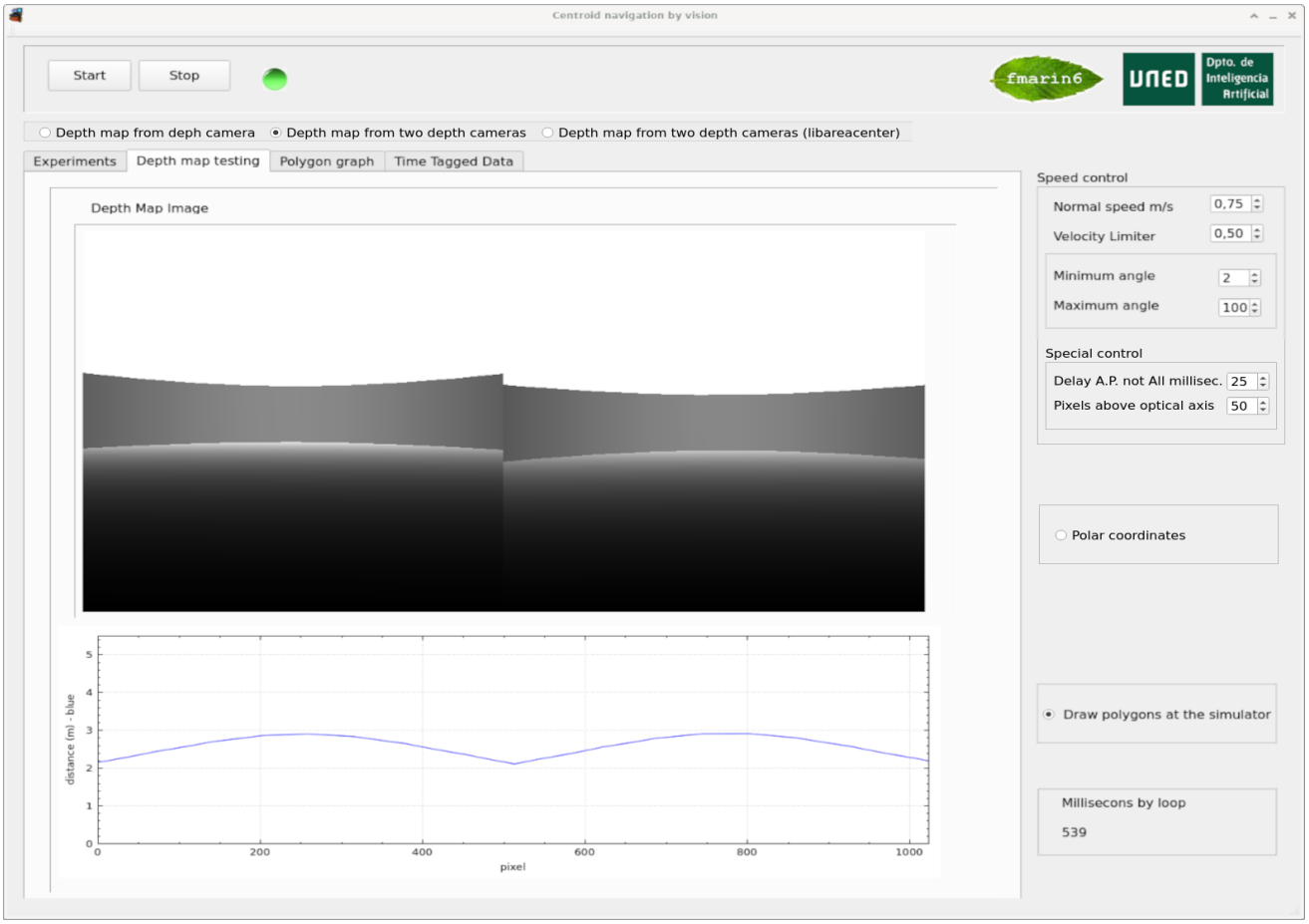

Figura 8.3: Medida frente a un muro circular

8.2. Error simxGetVisionSensorDepthBuffer

En este trabajo se utiliza dos cámaras simuladas kinect. En realidad, la simulación de la cámara kinect se realiza en base al, denominado por CoppeliaSim, Visionsensor. Un Visionsensor produce dos imágenes, una RGB y otra depth map. Para extraer el mapa de profundidad se utiliza el método simxGetVisionSensorDepthBuffer. En el experimento del circuito cerrado se observó que el centro de áreas no se situaba en aproximadamente el centro de un pasillo recto. El profesor José Manuel Cuadra sospechó de la bondad de la medida del mapa de profundidad. Por ello, sugirió situar las cámaras en el centro de una pared circular. En la gráfica de la Figura 8.3 se observa que se describe una curva y no una línea en tal experimento.

Se realizaron diferentes ensayos para intentar crear una función que tuviera en cuenta el ángulo y la distancia a los objetos. Todo apuntaba a una función cuadrática pero el profesor sugirió finalmente una función lineal. El resultado de aplicar la función se puede ver en la gráfica de la Figura 8.4.

Para corroborar el problema se realizó un prototipo rápido en Python creando una cámara estéreo con OpenCV y hacer la comparación con el que genera CoppeliaSim. En la Figura 8.5 se

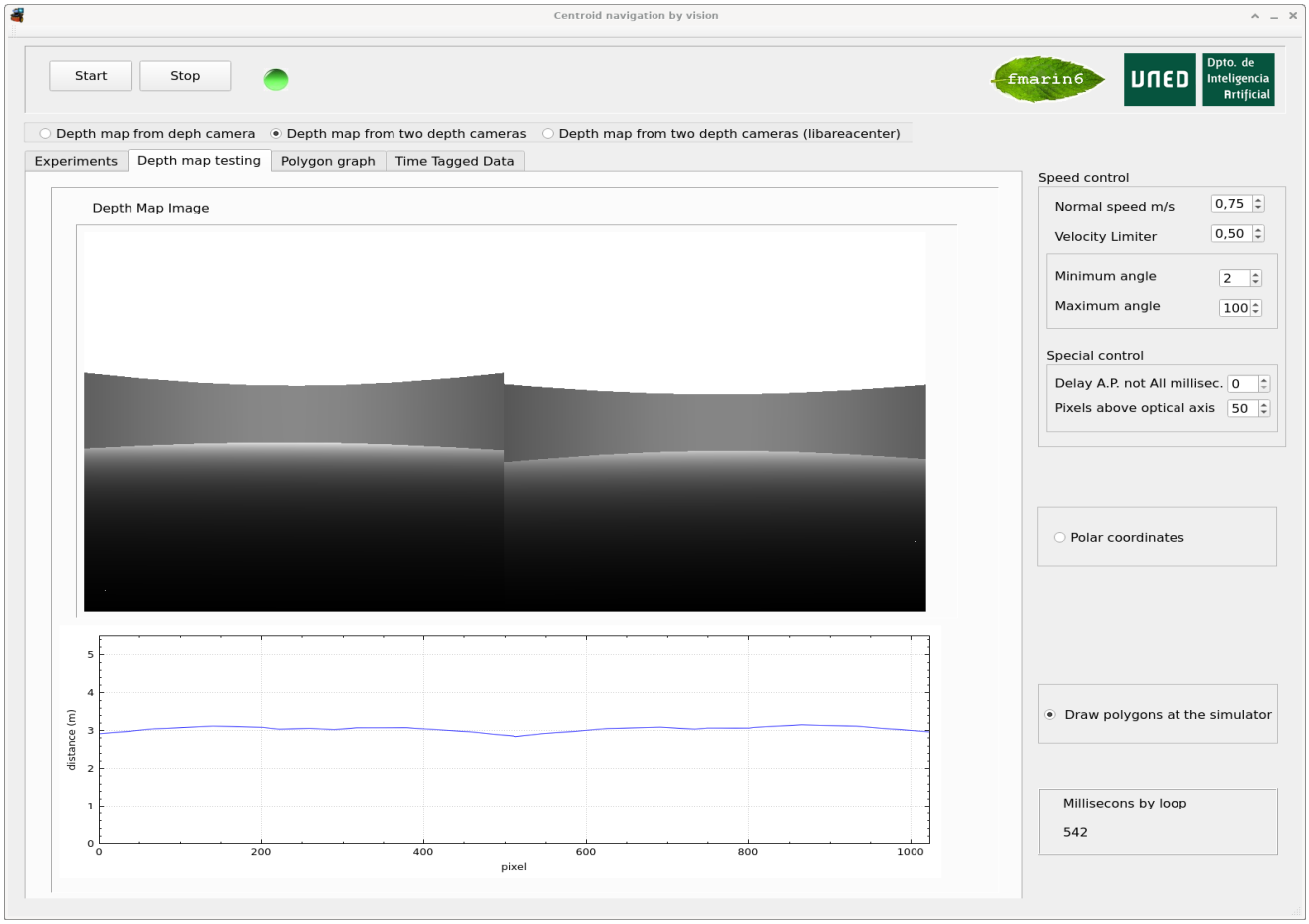

Figura 8.4: Medida frente a un muro corregida

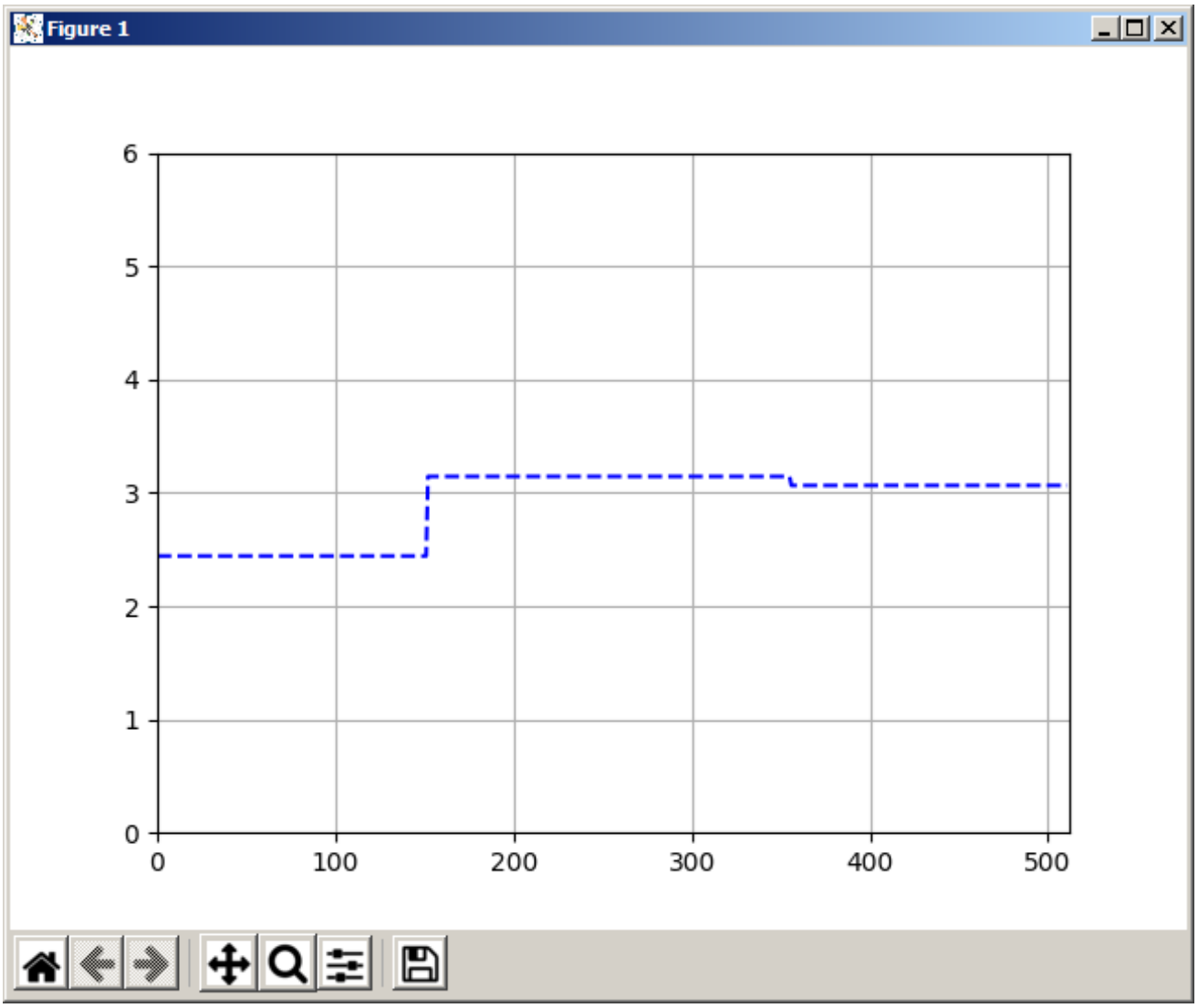

Figura 8.5: Medida frente a un muro con cámara estéreo

puede ver el resultado. No hay que tener en cuenta la parte izquierda. El algoritmo de OpenCV pierde los primeros píxeles, tantos como el valor del parámetro numDisparities. Se introdujo una modificación para recuperarlo, pero cambia el valor de profundidad para esos píxeles.

La siguiente Figura 8.6 muestra la medida producida por el Depth Map Buffer de CoppeliaSim. El muro circular se sitúa a 3 m. de la cámara.

Estos errores se han reportado a Coppelia Robotics vía correo electrónico y en el momento de redactar esta memoria Coppelia Robotics está trabajando para solucionar estos problemas.

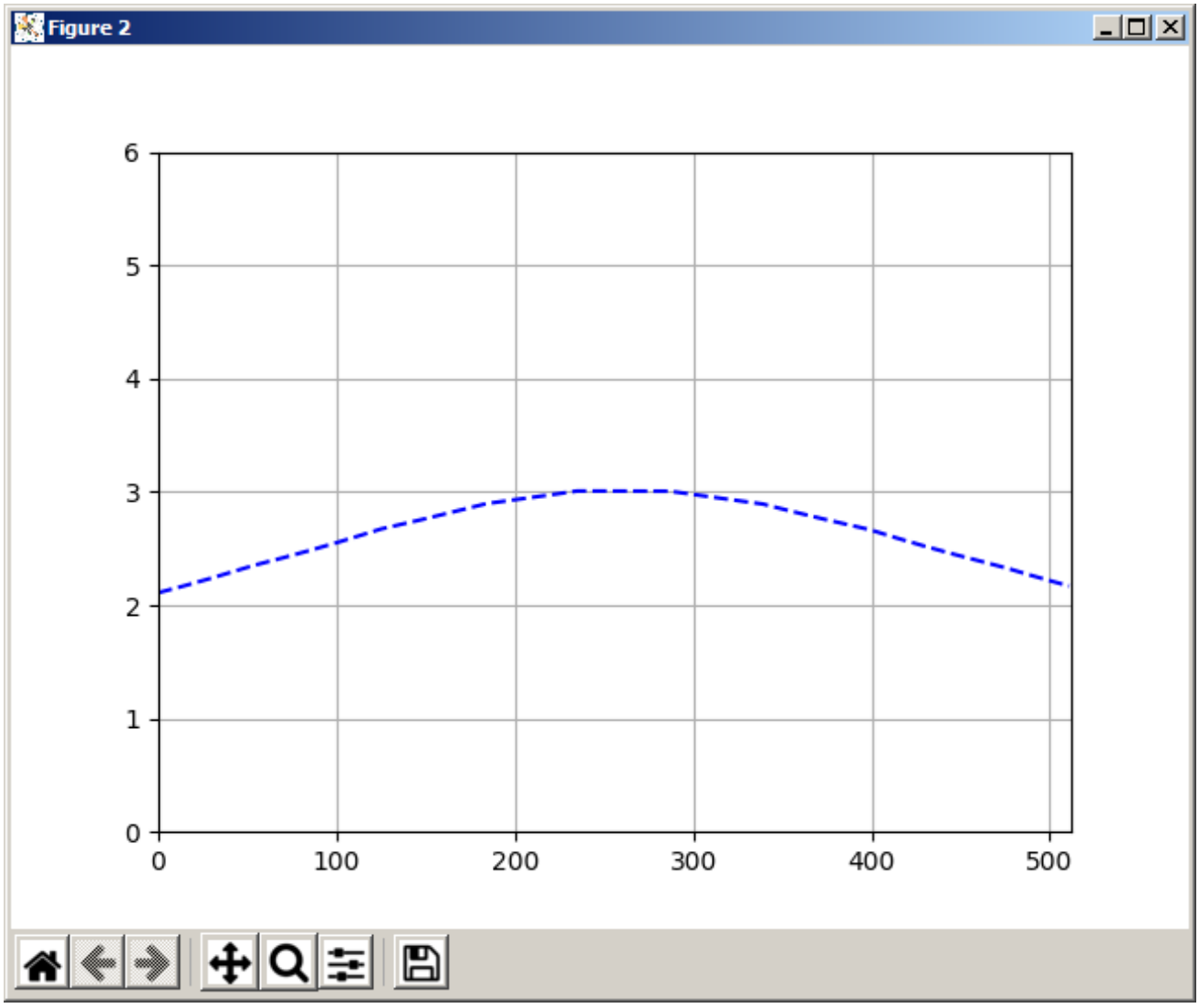

Figura 8.6: Medida frente a un muro Depth Map Buffer CoppeliaSim

Capítulo 9

Experimentos realizados

La Tesis Doctoral del profesor D. José Manuel Cuadra Troncoso Cuadra Troncoso (2011) ha servido de guía de los experimentos realizados toda vez que se trataba de comparar los experimentos de la Tesis realizados con la medida de rango láser con los realizados en este trabajo con las cámaras. Son ocho experimentos o, para ser exactos, diecisiete experimentos si se contabiliza las cuatro variantes de tres de ellos. En el primero de la lista se denomina *Divisiones sucesivas para pasar dos obstáculos*. En realidad, es una extensión de una Figura de la Tesis a la que se le ha introducido más obstáculos y tres paneles transparentes. La idea de introducir los paneles transparentes es para demostrar que son reconocidos como obstáculos en el caso de utilizar cámaras frente al problema que tiene al respecto la medida de rango láser. En *Alcance de objetivos usando el método del centro de áreas* se prueba la parte de la librearía *libareacenter* que se ocupa precisamente del alcance de objetivos mediante dos parámetros, la distancia al objetivo y su dirección. En *Trayectorias seguidas por el robot al deambular por un entorno doméstico o de oficina* se muestra el comportamiento emergente de llegar a todos los habitáculos. Sobre la *Comparación de recorridos en un circuito cerrado* se ha introducido como novedad, situar un objeto sobre el suelo y dos puentes, uno de ellos de dos niveles y así verificar la característica acotado-3D que proporciona la utilización de cámaras frente a la medida de rango láser 2D. *Evitación de obstáculos con el objetivo a diferentes distancias de objetos aproximadamente circulares* son cuatro experimentos jugando con el diámetro del objeto circular y la posición del objetivo. Otros cuatro experimentos de *Evitación de obstáculos con el objetivo a diferentes distancias de objetos cóncavos*. La diferencia entre ellos es la forma del objeto cóncavo y de nuevo la posición del objetivo. A*lcanzando objetivos en mundos complejos simulando sensores ideales* también son cuatro experimentos que se diferencian en la posición y la orientación del robot con respecto al "mundo". En *Experimento con robot real (simulación)* se ha tratado de emular el experimento que se realizó con el Pioneer-3AT. Aquí se vuelve a emplear la parte de la librearía *libareacenter* que se ocupa del alcance de objetivos.

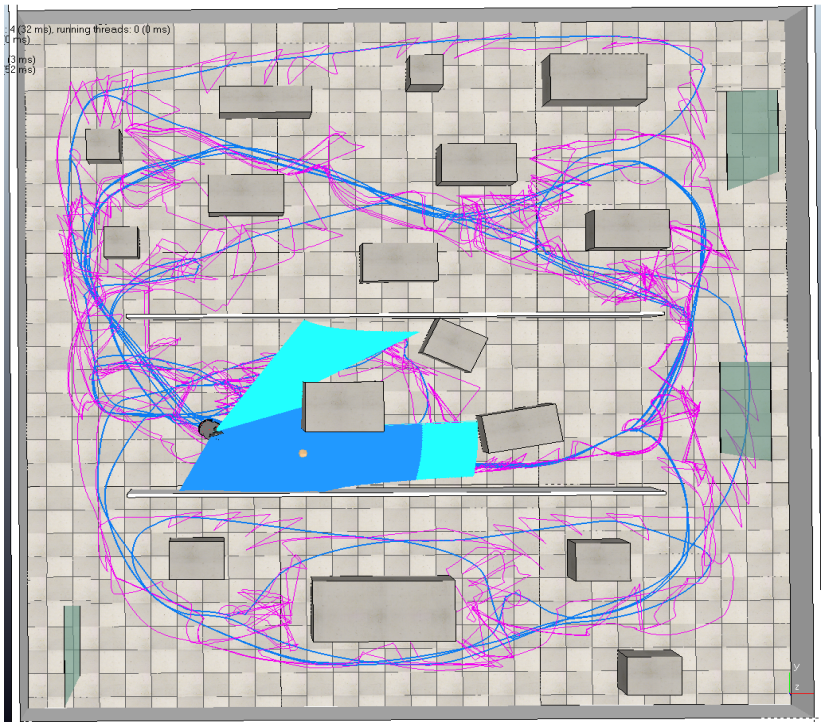

Figura 9.1: tfm_Experiment_1

9.1. Divisiones sucesivas para pasar dos obstáculos

En términos de este trabajo se ha denominado tfm_Experiment_1 (Figura 9.1).

En esta Figura 9.1 se puede observar la trayectoria del robot (azul) y la posición del centro de áreas (magenta). También se puede observar la figura de los dos polígonos que maneja la librería, el polígono global (azul claro) y el polígono de avance (azul). Como curiosidad se han llevado al mundo que representa esta Figura 9.1 tres paneles transparentes (color verde simulando una mampara de vidrio transparente) que como se sabe no serían vistos por el láser y así se constata una ventaja más de utilizar cámaras. El robot pasa por la parte derecha del panel transparente situado en la parte central derecha.

9.2. Alcance de objetivos usando el método del centro de áreas

En la siguiente tfm_Experiment_2 (Figura 9.2) se trata alcanzar un objetivo.

Con la imagen de una tercera cámara, se ha utilizado visión artificial para hallar los momentos de la imagen objetivo. Desde los momentos se calcula el área y el ángulo del objetivo en la imagen. Se conduce al robot al objetivo mediante obstáculos virtuales. Se ha establecido un valor de área del objetivo en la imagen para parar el robot en la “diana“ de color verde apoyada en el suelo. En la parte superior de la imagen se puede ver dicha diana y su correspondiente

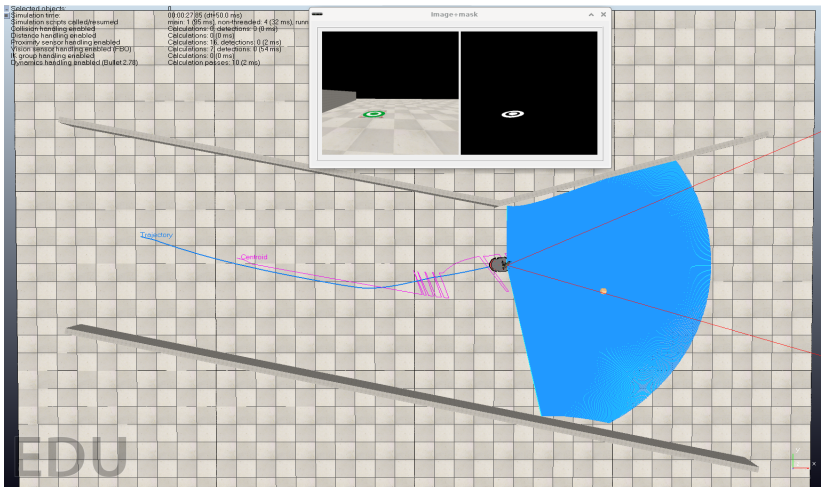

Figura 9.2: tfm_Experiment_2

imagen filtrada.

9.3. Trayectorias seguidas por el robot al deambular por un entorno doméstico o de oficina

En la siguiente tfm_Experiment_3 (Figura 9.3) se observa una vez más la trayectoria del robot y la posición de los cálculos sucesivos del centro de áreas. Como se vio en el mismo experimento de la tesis aparece el comportamiento emergente de deambular por todos los habitáculos del "mundo". En la parte inferior izquierda se puede observar con el polígono de avance (azul) coincidente con el polígono global que se adivina debajo del polígono de avance por las aristas cercanas al robot de color azul claro.

9.4. Comparación de recorridos en un circuito cerrado

En la siguiente tfm_Experiment_4 (Figura 9.4) tiene alguna diferencia con la Tesis. Para probar la característica acotado-3D se introducido un objeto "piedra" al lado del robot en la Figura 9.4 y dos puentes el primero con dos niveles.

Se puede observar como el robot esquiva la "piedra" cuando circula en el sentido de las agujas del reloj y también en sentido contrario, por eso las diferencias en las trayectorias. El puente de dos niveles es el primero más a la izquierda. Para ilustrar mejor el paso por los puentes se ofrece la siguiente Figura 9.5. En la parte superior una toma de la cámara situada en la zona y en la inferior a vista de pájaro.

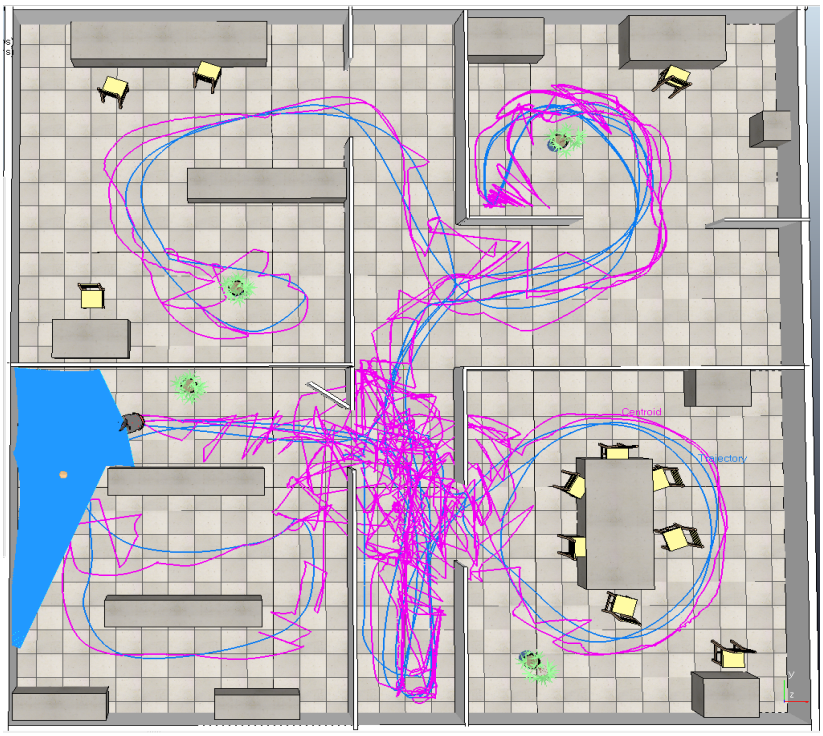

Figura 9.3: tfm_Experiment_3

Figura 9.4: tfm_Experiment_4

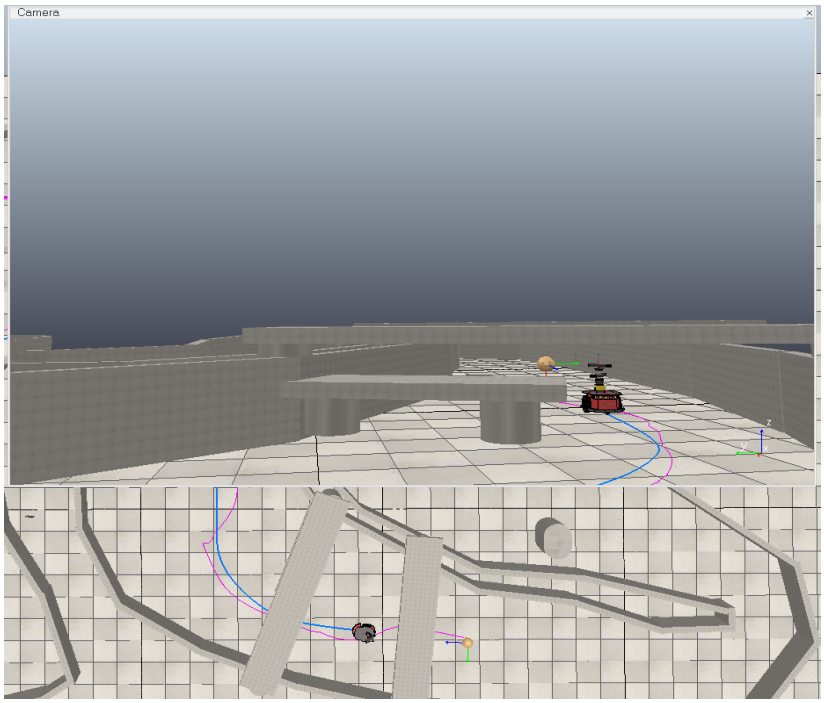

Figura 9.5: Paso por los puentes

9.5. Evitación de obstáculos con el objetivo a diferentes distancias de objetos aproximadamente circulares.

La importancia de los cuatro siguientes experimentos estriba en la capacidad de la navegación reactiva por centro de áreas de evitar obstáculos interpuestos entre el robot y el objetivo. Hay que decir que en estos experimentos se ha utilizado la navegación con los parámetros de proximidad y dirección que ofrece la librería *libareacenter* para alcanzar objetivos.

En la siguiente Figura 9.6, se intercala un objeto circular de 4 metros de diámetro y el objetivo se sitúa a 3,66 metros del centro de objeto circular. La posición inicial del robot se sitúa a 4,66 metros del centro del objeto circular. Se puede apreciar la trayectoria (trazo azul) del robot hasta alcanzar el objetivo y las variaciones del cálculo del centro áreas (trazo magenta) para rodear el objeto circular.

La Figura 9.7 es un nuevo experimento en el cual se ha conservado el objeto circular, pero se ha cambiado la posición de objetivo que ahora se sitúa a 6,20 metros del centro del objeto circular. La posición inicial del robot se sitúa a 4,44 metros del centro del objeto circular. De nuevo la trayectoria del robot y traza producida por los sucesivos cálculos del centro de áreas.

En siguiente experimento de la Figura 9.7 el objeto circular es de tan solo 2 metros de diámetro y el objetivo se sitúa 4,46 metros de distancia del centro del objeto circular. La posición inicial del robot se sitúa a 2,21 metros del centro del objeto circular. Se aprecia una vez más como el robot alcanza el objetivo.

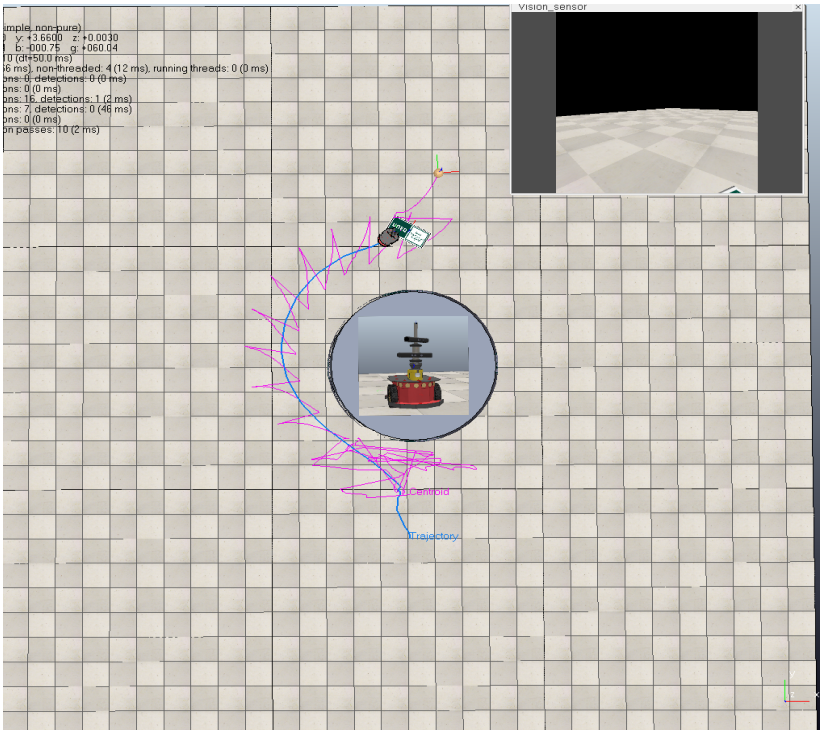

Figura 9.6: tfm_Experiment_5a. El diámetro del circulo es 4 metros

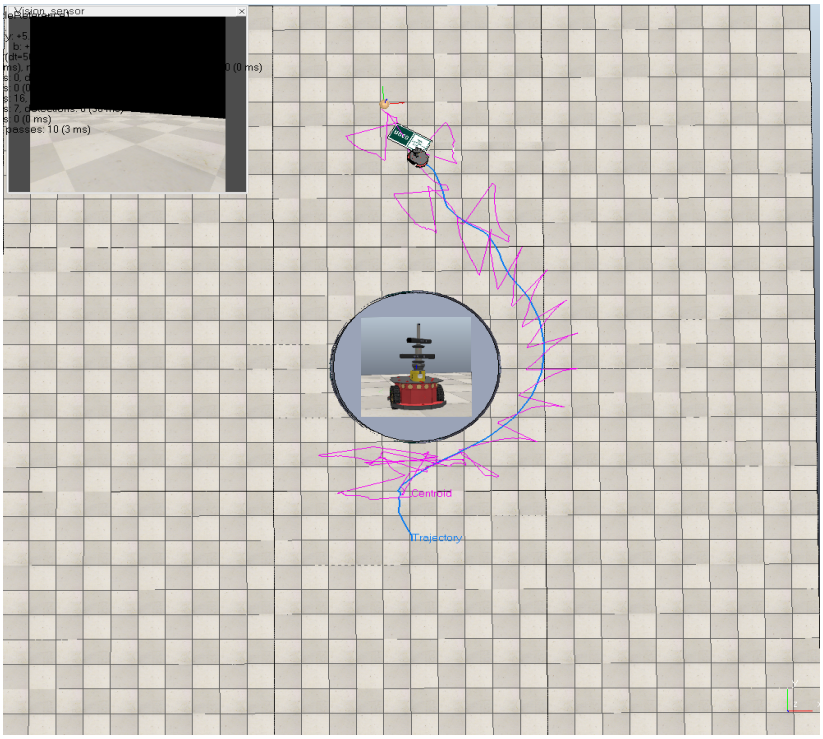

Figura 9.7: tfm_Experiment_5b. El diámetro del circulo es 4 metros

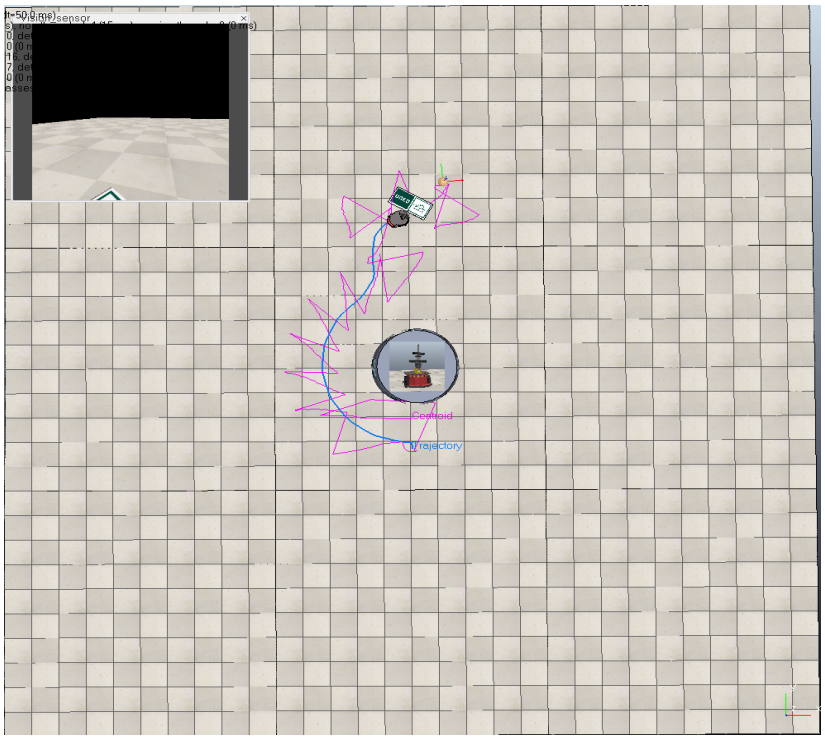

Figura 9.8: tfm_Experiment_5c. El diámetro del circulo es 2 metros

Cierra estos cuatro experimentos el experimento de la Figura 9.9. Aquí se utiliza el objeto circular de 2 metros de diámetro y el objetivo se sitúa 1.6 metros de distancia del centro del objeto circular. La posición inicial del robot se sitúa a 2.64 metros del centro del objeto circular. Nótese que la curvatura de la trayectoria provoca que el robot sobrepase levemente el objetivo. Es debido a la cercanía del objetivo al objeto circular.

9.6. Evitación de obstáculos con el objetivo a diferentes distancias de objetos cóncavos

Si los cuatro experimentos del punto anterior tienen su especial dificultad, los cuatro siguientes entrañan si cabe una mayor dificultad por ser objetos cóncavos. Las denominadas trampas en C y en U son un verdadero reto ya que el robot pudiera quedarse en el interior de los objetos en forma de C o de U si se le encamina a la cara cóncava del objeto. Los siguientes experimentos demostrarán que la navegación por centro de áreas logra esquivar estos objetos cóncavos y alcanzar el objetivo. Otra vez en estos experimentos se ha utilizado la navegación con los parámetros de proximidad y dirección que ofrece la librería *libareacenter* para alcanzar objetivos.

El primer experimento de esta serie se puede ver en la Figura 9.10. Se puede observar un objeto en forma de C de 5,375 m. de anchura y 2,050 m. de profundidad. El robot está situado a 5,58 metros de la parte más profunda del objeto y en línea con su eje de simetría. El objetivo

Figura 9.9: tm_Experiment_5d. El diámetro del circulo es 2 metros

se ha situado a 8,06 metros del punto inicial donde se sitúa el robot. Se puede observar que el robot intenta ir aproximadamente en línea recta hacia el objetivo, pero cuando está a cierta distancia del objeto comienza a esquivarlo consiguiendo salir de él y una vez rebasado alcanza el objetivo.

La única diferencia del experimento de la Figura 9.11 estriba en que se ha situado el objetivo en otra posición. Ahora se sitúa una distancia de 10,54 metros desde el punto inicial donde se sitúa el robot. El objeto en forma de C sigue teniendo las mismas dimensiones de 5,375 m. de anchura y 2,050 m. de profundidad. Una vez más el robot no cae en la trampa y alcanza el objetivo.

El experimento al que se refiere la Figura 9.12, el objeto tiene forma de U y sus dimensiones 3,7 m. de anchura y 4.5 m. de profundidad. La posición inicial del robot está situada a una distancia de 8,68 metros de la pared horizontal del objeto en forma de U. El objetivo está situado a 1,86 metros de la citada pared y a 10,54 metros del punto inicial donde se sitúa el robot. De nuevo se puede apreciar que el robot navega hacia el objetivo, pero cuando está cerca de la pared horizontal el polígono de avance es pequeño el robot rota y se sale de esta trampa y finalmente alcanza el objetivo.

La diferencia del experimento de la Figura 9.13 con el experimento anterior es la posición que se ha establecido del objetivo. Concretamente a 4,40 metros de la pared horizontal y a 13,08 metros del punto inicial donde se sitúa el robot. La evolución de la trayectoria del robot como siempre va aproximadamente en línea recta hasta la pared horizontal del objeto en forma de U, pero la salida de la trampa no es tan “limpia” como el experimento anterior. Aquí el robot

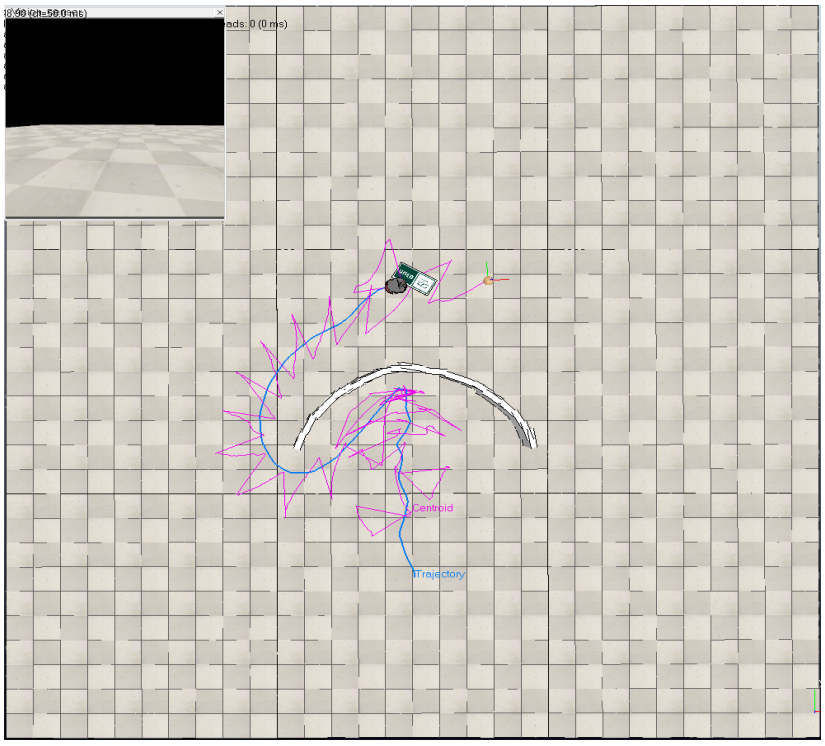

Figura 9.10: tfm_Experiment_6a. Las dimensiones del objeto en forma de C son 5,375 m. de anchura y 2.050 m. de profundidad.

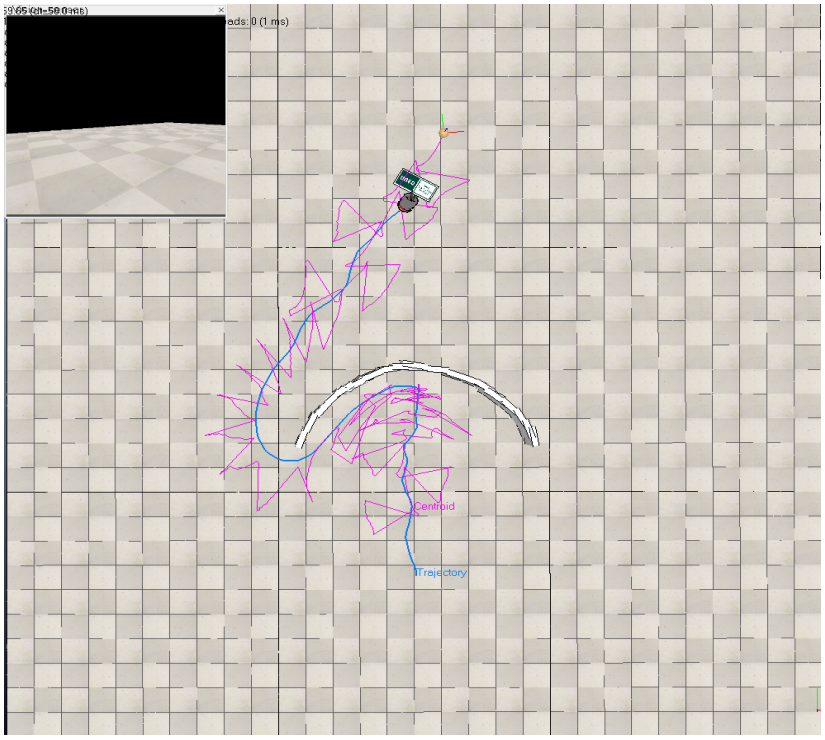

Figura 9.11: tfm_Experiment_6b. Las dimensiones del objeto en forma de C son 5,375 m. de anchura y 2.050 m. de profundidad.

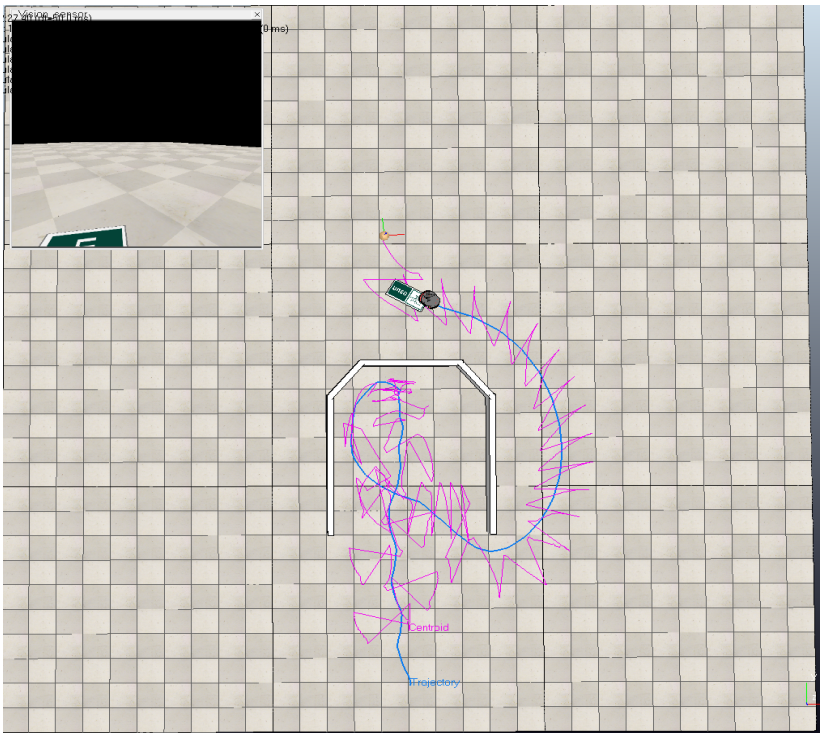

Figura 9.12: tfm_Experiment_6c. Las dimensiones del objeto en forma de U son 3,7 m. de anchura y 4.5 m. de profundidad.

vuelve en un primer ciclo hacia la pared horizontal para darse de nuevo la vuelta y finalmente salirse de la trampa y alcanzar el objetivo.

9.7. Alcanzando objetivos en mundos complejos simulando sensores ideales

En esta sección cuatro experimentos para probar la eficacia de la navegación por centro de áreas utilizando cámaras de profundidad a través de mundos complejos. Como en los anteriores experimentos se ha utilizado la navegación con los parámetros de proximidad y dirección que ofrece la librería *libareacenter* para alcanzar objetivos.

El primer experimento de esta serie de cuatro experimentos en la Figura 9.14. La posición inicial del robot es obviamente el inicio de trayectoria representada por la traza azul, pero no se puede observar la orientación que en este caso es orientación este. Si se traza una línea desde la posición inicial del robot hasta el objetivo y equidistante de los objetos por donde discurre el robot se ve que la navegación discurre por esa línea o trayectoria cuasi-óptima hasta alcanzar el objetivo. Se puede afirmar la que la eficacia de la navegación por centro de áreas es plena.

Se está ante un nuevo experimento que se muestra en la Figura 9.15. El robot parte de otra posición, pero con orientación sur. De nuevo si se imagina una línea o trayectoria cuasi-óptima

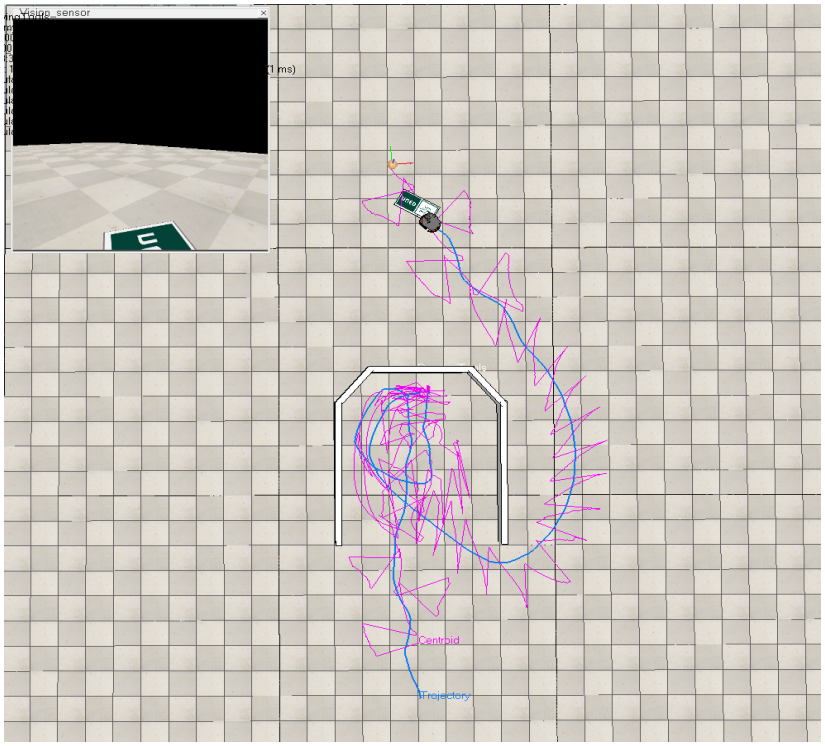

Figura 9.13: tfm_Experiment_6d. Las dimensiones del objeto en forma de U son 3,7 m. de anchura y 4.5 m. de profundidad.

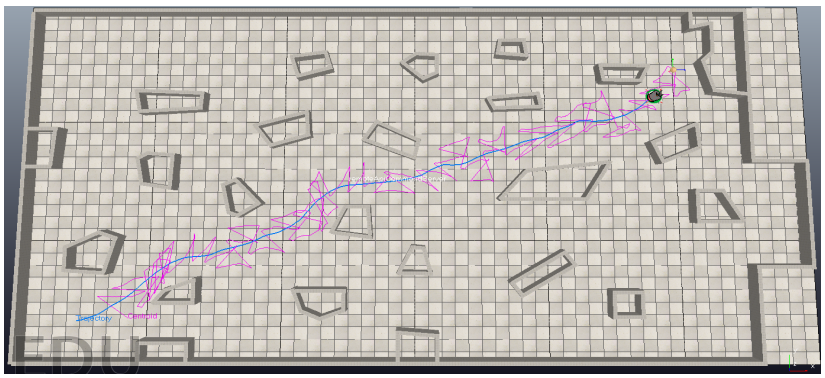

Figura 9.14: tfm_Experiment_7a

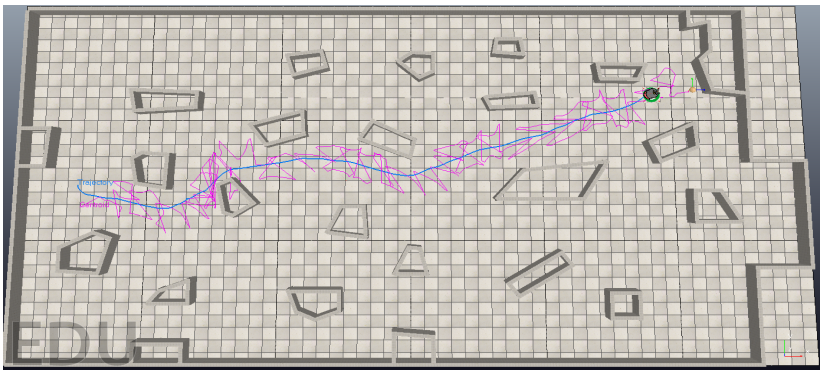

Figura 9.15: tfm_Experiment_7b

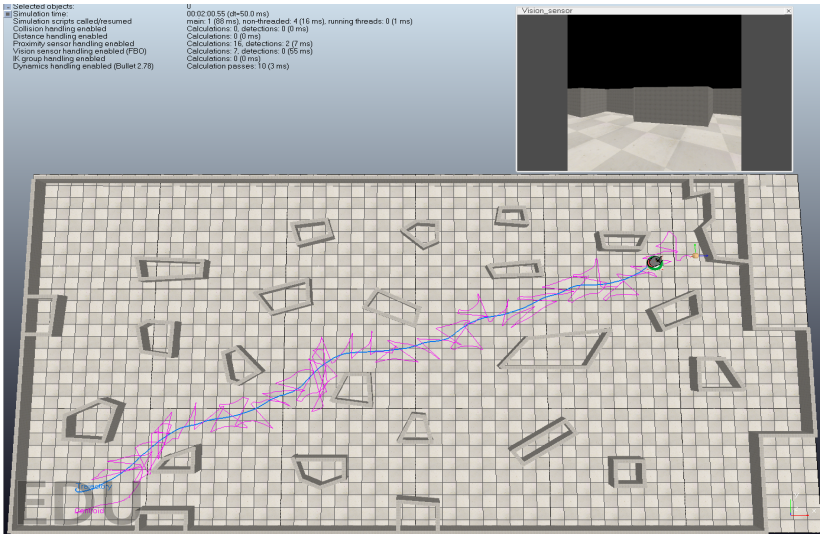

Figura 9.16: tfm_Experiment_7c

desde la posición inicial del robot y equidistante a los objetos por donde discurre el robot y hasta el objetivo la trayectoria seguida por el robot se acerca a la trayectoria cuasi-óptima. El experimento demuestra su eficacia en otra posición y orientación.

El presente experimento de la Figura 9.16 es semejante al primer experimento de esta sección. La única diferencia es que la orientación del robot es sur. Su posición inicial es la misma. Aplica decir aquí lo que se dijo en el primer experimento, pero habría que reseñar como el robot gira rápidamente (pequeña curva del trazo azul) y retoma la orientación este del primer experimento.

El experimento de la Figura 9.17 es una repetición del segundo experimento.

9.8. Experimento con robot real (simulación)

En este experimento, Figura 9.18, se simula el experimento que se realizó en la Tesis con el robot Pioneer 3-AT. Como en los anteriores experimentos se ha utilizado la navegación con los parámetros de proximidad y dirección que ofrece la librería *libareacenter* para alcanzar

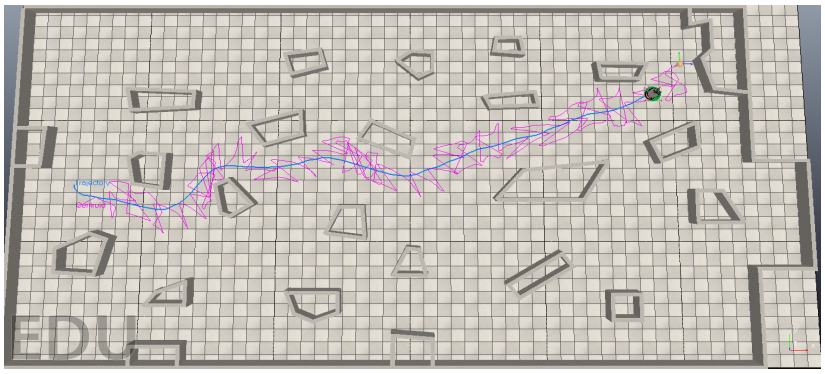

Figura 9.17: tfm_Experiment_7d

objetivos. Se ha simulado una cámara Canon VC 50i, pero sólo el movimiento panorámico de ± 100°. De nuevo visión artificial y momentos. Se calcula la distancia por medio de una función exponencial de la raíz cuadrada del área, previamente ajustada utilizando otra técnica. La dirección se determina por la posición del objeto en la imagen. Se busca el objetivo con el movimiento panorámico de la cámara y cuando se visualiza (área>0) se lleva el pan de 0.005 en 0.005 grados a la posición 0 grados+/-0.1 grados. Se para el robot cuando el valor de proximidad llega al valor establecido.

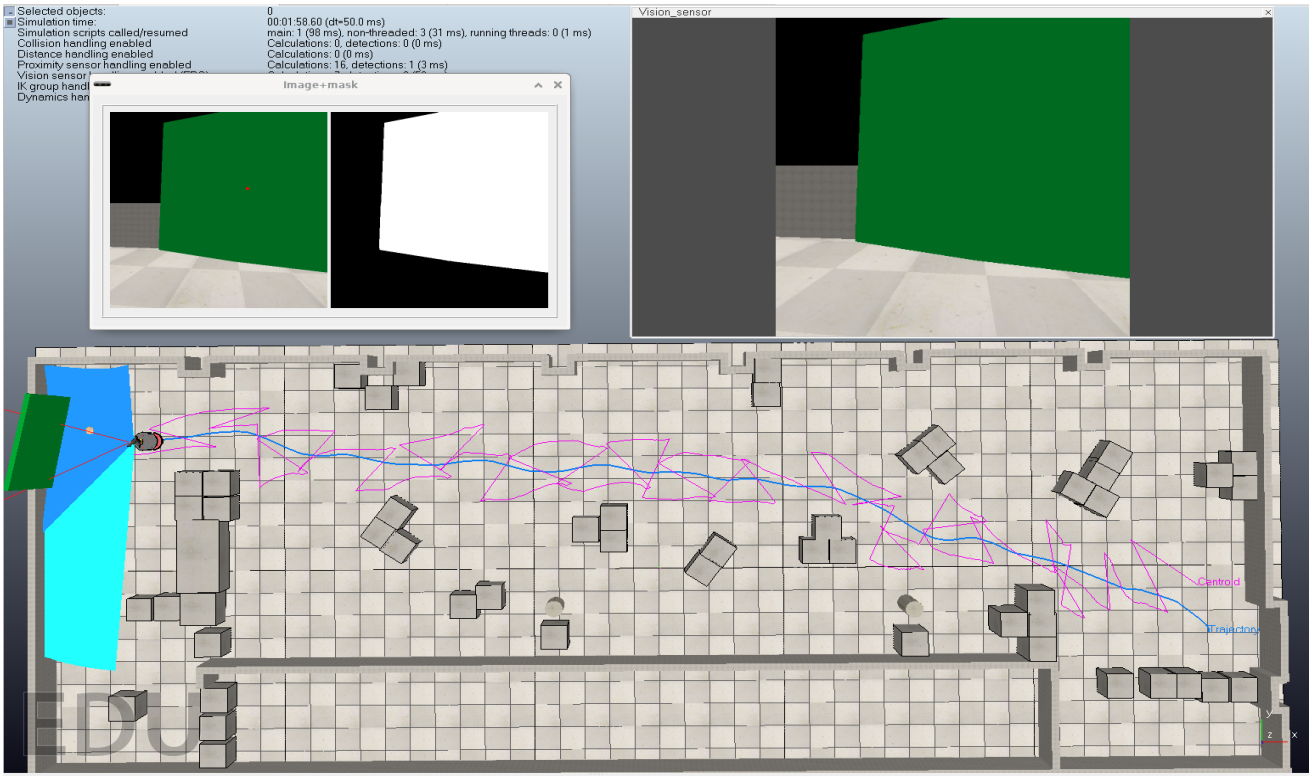

Figura 9.18: tfm_Experiment_8.

Capítulo 10

Conclusiones y trabajos futuros: 3D

En este capítulo se tratará de las conclusiones. Al principio de este trabajo se fijó unos objetivos y es el momento de verificar si se puede llegar a la conclusión de que se han logrado. Como suele ocurrir en cualquier proceso de investigación, aparecen problemas que se han resuelto, pero lo más importante es que aparezcan nuevos elementos que originan o puedan originar un nuevo hilo de investigación que se recoge en trabajos futuros. Las cinco asignaturas que preceden a la elaboración de este trabajo te invitan a recorrer mentalmente otras posibilidades. Se pretende decir con ello que, cualquier trabajo en investigación en IA no es un trabajo cerrado sino que se puede mejorar introduciendo otras técnicas, otras posibilidades. Espero que la navegación por centro de áreas tenga su recorrido. Si la comparamos con técnicas de navegación globales se podría decir que esta técnica de navegación por centro de áreas es más pura en el sentido de lo que conocemos como inteligencia artificial.

10.1. Conclusiones

En este punto hay que recordar los objetivos planteados y se va verificar si se han cumplido uno por uno:

1. Mejorar la visión del "mundo" que genera la medida de rango láser mediante el empleo de cámaras de mapa de profundidad.

   Al inicio de este trabajo se planteó el objetivo de mejorar la utilización de la medida de rango láser mediante la utilización de la técnica denominada 2D ½. Como se dijo, el láser 2D sólo detecta objetos en su plano de barrido. Incluso los actuales sólo llegan a 128 planos de barrido. Con la citada técnica, los objetos pueden estar en tantos planos como resolución vertical tenga la cámara. La resolución horizontal equivale al número de haces del láser pudiendo ser un número superior a 180. Luego mayor sensibilidad o precisión. El experimento 9.4 demuestra que se puede sortear un obstáculo de pequeñas dimensiones

sobre el suelo. Se puede navegar por debajo de un puente con dos niveles pasando el robot por debajo del puente de mayor altura.

2. Aprovechar el uso de cámaras para evolucionar la medida a 3D.

   La característica de las cámaras de 2D ½ se puede convertir mediante un simple cálculo a 3D. En otras palabras, se puede extraer la altura de los objetos. Como se manifestó en 5.3 un "mundo" sobre el suelo no necesita esta característica 3D, es suficiente con lo que se ha denominado acotado-3D. Precisamente esta característica es la que ha permitido la navegación del experimento 9.4 del objetivo anterior.

3. Anular el defecto de la variación de la medida láser en función de ángulo de incidencia del haz láser/superficie del objeto para ciertos tipos de materiales y superficies transparentes mediante la utilización de dichas cámaras.

   Como es conocido, la medida de rango láser se basa en un pulso de luz infrarrojo o haz (tiempo de emisión) que se proyecta sobre el objeto a medir su distancia y este refleja el haz que es recibido por el dispositivo láser (tiempo de recepción). La distancia al objeto se determina calculando el tiempo entre la emisión del haz y la recepción del haz reflejado que viaja a la velocidad de la luz. El ángulo de incidencia del haz con la superficie del objeto para determinados materiales distorsiona la medida. Una pared transparente no reflejan los haces del láser, luego no la puede detectar como objeto. Estos problemas no existen al utilizar cámaras de mapa de profundidad. Todos los puntos de todos los objetos perteneciente al "mundo" son proyecciones perpendiculares al plano de visión de la cámara. En el experimento 9.1 se evidencia como los paneles transparentes situados en ese "mundo" son tomados por las cámaras como objetos. Todos lo objetos con diferentes tamaños y formas son perfectamente "detectados" como lo demuestra el experimento 9.7 .

10.2. Trabajos futuros

Como se manifestó en la sección 5.3, se puede extraer información 3D de las cámaras. Si bien no ha sido el objetivo de este trabajo, esta característica se podría utilizar en la navegación por centro de áreas en drones. Ahora se obtendría una superficie en coordenadas esféricas en vez de polígonos en coordenadas polares.

Desafortunadamente debido a la situación que estamos viviendo ha sido imposible realizar el experimento (9.8) con un robot real pero quiero expresar mi firme compromiso de, como trabajo futuro, diseñar un dispositivo para la navegación por centro de áreas, con dos cámaras de mapa de profundidad como entrada y las velocidades al robot como entrada/salida y experimentar con el Pioneer-3AT.

Aunque, como se ha dicho, el experimento con un robot real es un trabajo futuro, si se ha realizado un exhaustivo estudio de la viabilidad del computador de una sola tarjeta (SBC) Raspberry Pi 4 (ARMv8) para que sea el nodo del futuro dispositivo (CND, Centroid Navigation Device) y se ha llegado a la conclusión que la arquitectura ARM no es la más adecuada para un desarrollo porque pudiera verse coartada la libertad de utilizar librerías que no estén compiladas para ARM. Además, la potencia espacial y temporal de dicho computador es limitada para propósitos de visión artificial. El estudio propone, como solución, utilizar una arquitectura x86, SO Debian y el SBC UDOO BOLT V8 en su su ejecución kit UDOO GEAR que incluye wifi. Lo completaría un disco duro del tipo bus M.2, 32 GB de RAM y un hub de comunicaciones. También un convertidor DC/DC 12 VDC a 19 VDC. Las librerías del lado del Pioneer, Aria y ArNetworking, se demuestra en el estudio que, son compilables en la última versión de Debian y por ende integrables en la aplicación final que se pudiera denominar Centroid Navigation Application para la captura de imágenes, cálculo de las medidas de distancias a los objetos y el control de la navegación.

Bibliografía

Álvarez, J. R., de la Paz, F., and Mira, J. (1999). On virtual sensory coding: An analytical model of the endogenous representation. In Mira, J. and Sánchez-Andrés, J. V., editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks*, pages 526–539, Berlin, Heidelberg. Springer Berlin Heidelberg.

Arkin, R. C. (1998). Behavior-Based Robotics. *MIT Press*.

Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288.

Brooks, R. A. (1991). Intellingence without representation. *Artificial Intelligence*, (47):139–159.

Clark, R. Arkin, R. C. and Ram, A. (1992). Learning momentum: On-line performance enhancement for reactive systems. pages 111–116.

Cuadra Troncoso, J. M. (2011). *Modelado adaptativo del medio para la navegación de robots autónomos utilizando algoritmos basados en el centro de áreas*. PhD thesis.

de la Paz, Félix and Álvarez, José R. and Rosado Sáchez, José Ignacio and Cuadra Troncoso, José Manuel (2009). Mathematical Foundations of the Center of Area Method for Robot Navigation. volume 2, pages 419–428.

De la Paz López, F. (2003). *Una arquitectura que integra el modelado endógeno del medio y la navegación para un robot de dos ruedas*. PhD thesis.

Fahimi, F. Nataraj, C. and Ashrafiuon, H. (2009). Real-time obstacle avoidance for multiple mobile robots. *Robotica*, (27):189–198.

Ferrara, A. and Rubagotti, M. (1996). A navigation system for mobile robots using visual feedback and artificial potential fields. pages 1159–1164.

Ferrara, A. and Rubagotti, M. (2009). ICRA 2007 Workshop: Planning, Perception and Navigation for Intelligent Vehicles.

Fox, D. and Burgard, W. (1997). The dynamic window approach to collision avoidance. *Robotics and Automation Magazine, IEEE*, 4:23 – 33.

Goncalves, V. M., Pimenta, L. C. A., Maia, C. A., Dutra, B. C. O., and Pereira, G. A. S. (2010). Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4):647–659.

Gómez Palomo, S. R. and Moraleda Gil, E. A. (2020). *Aproximación a la Ingeniería del Software*.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Jacob, H. Fede, S. and Slotine, J. (1997). Real-time path planning using harmonic potential functions in dynamic environment. pages 874–881.

Kim, J. and Khostola, P. (1992). Real-time obstacle avoidance using harmonic potencial functions. *IEEE Transactions on Robotics and Automation*, (8(3)):338–349.

Koren, Y. and Borestein, J. (1991). Potential methods and their inherent limitations for movile robot navigation. volume 2, pages 1398–1404.

Krogh, B. (1984). A generalized potential field approach to obstacle avoidance control. *Technical report*.

Maravall, D. de Lope Asiain, J. and Serradilla, F. (2000). Combination of model-based and reactive methods in autonomous navigation. pages 2328–2333.

Masoud, S. and Masoud, A. (2000). Constrained motion control using vector potential fields. *IEEE Transactios on Robotics and Automation*, (30(3)):251–272.

Minguez, J., Montano, L., and Khatib, O. (2002). Reactive collision avoidance for navigation with dynamic constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 588–594 vol.1.

Murphy, R. R. (2005). *Introduction to AI ROBOTICS*.

Orozco, R., Loscos, C., Martin, I., and Artusi, A. (2017). Chapter 3 - hdr multiview image sequence generation: Toward 3d hdr video. In Chalmers, A., Campisi, P., Shirley, P., and Olaizola, I. G., editors, *High Dynamic Range Video*, pages 61–86. Academic Press.

Ram, A., Arkin, R., Moorman, K., and Clark, R. (1997). Case-based reactive navigation: a method for on-line selection and adaptation of reactive robotic control parameters. *IEEE Transactions on Robotics. Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(3):376–394.

Sánchez, J. R. Á., de la López, F. P., Troncoso, J. M. C., and de Sierra, D. S. (2010). Reactive navigation in real environments using partial center of area method. *Robotics and Autonomous Systems*, 58(12):1231–1237.

Sánchez, J. R. Á., de la López, F. P., Troncoso, J. M. C., and Sánchez, J. I. R. (2009). Partial Center of Area Method Used for Reactive Autonomous Robot Navigation. In Mira, J. M., Ferrández, J. M., Álvarez, J. R., de la Paz, F., and Toledo, F. J., editors, *IWINAC (2)*, volume 5602 of *Lecture Notes in Computer Science*, pages 408–418. Springer.

Tianmiao, W. and Bo, Z. (1992). Time-varying potential field based 'perception-action' behaviors of mobile robot. volume 3, pages 2549–2554.

Ulrich, I. and Borenstein, J. (1998). Vfh+: reliable obstacle avoidance for fast mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 2, pages 1572–1577 vol.2.

Ulrich, I. and Borenstein, J. (2000). Vfh*: local obstacle avoidance with look-ahead verification. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, volume 3, pages 2505–2511 vol.3.

Yang, H., Borenstein, J., and Wehe, D. (2001). Sonar-based obstacle avoidance for a large, nonpoint, omnidirectional mobile robot.

Ögren, P. and Leonard, N. (2005). A convergent dynamic window approach to obstacle avoidance. *Robotics, IEEE Transactions on*, 21:188 – 195.