Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

"Sistema de ayuda a invidentes basado en cámaras de profundidad"

Autor: Adrián Asensio Vicente.

Tutores: Daniel Pizarro Pérez y David Jiménez Cabello.

2016

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

"Sistema de ayuda a invidentes basado en cámaras de profundidad"

Autor: Adrián Asensio Vicente.

Directores: Daniel Pizarro Pérez y David Jiménez Cabello.

Tribunal:

Presidente: Manuel Mazo Quintas.

Vocal 1º: María Soledad Escudero Hernanz.

Vocal 2º: Daniel Pizarro Pérez.

Calificación: .....

Fecha: .....

"La única discapacidad en la vida es una mala actitud."
Scott Hamilton

Agradecimientos

A mis tutores, Daniel Pizarro y David Jiménez, por su ayuda, buenos consejos y dedicación para que este proyecto saliera adelante.

A mis compañeros de clase y laboratorio con los que he pasado estos últimos años.

A toda mi familia, en especial a mis padres Antonio y Gloria y, a mi hermano Daniel.

Resumen

El presente proyecto final de grado tiene como objetivo el desarrollo de un sistema de ayuda para personas con discapacidad visual empleando técnicas de visión artificial. El algoritmo desarrollado tiene un funcionamiento en tiempo real y permite al usuario el aprendizaje de objetos y la detección de los mismos mediante un dispositivo RGB-D. Una vez detectados, su localización en el espacio será transmitida a la persona invidente por medio de una técnica de localización sonora llamada técnica binaural.

El algoritmo ha sido desarrollado sobre el framework de ROS, a través del cual podemos obtener la información que nos proporciona el dispositivo RGB-D utilizado y realizar los diferentes aspectos de procesado de imagen que se han empleado, tales como: filtrado de los bordes de la imagen RGB-D, detección, reconocimiento y entrenamiento de objectos, suavizado temporal mediante el Filtro de Kalman y la obtención de las coordenadas cartesianas de los diferentes objetos.

Para la realización de este trabajo se ha profundizado en el análisis de los diferentes sistemas y métodos para la detección y reconocimiento de objetos que existen en la actualidad, la mejora de estos mediante técnicas de visión artificial y, por último, el estudio y aplicación de las diferentes técnicas de localización mediante sonidos binaurales. Esto representa una oportunidad para la aplicación de dichos sistemas a la ayuda a personas invidentes.

Palabras clave: Sistema para invidentes, RGB-D, técnica binaural, filtro de Kalman, ROS.

Abstract

The main objective of this BsC Thesis is to develop a system for helping visually impaired people using methods from computer vision. The proposed system works in real time and allows the user to detect previously learned objects with a RGB-D camera. Once an object is detected in the image, the system computes its 3D position from the user's reference frame and sends this information to the user by means of an acoustic signal that is generated using a binaural localization model.

The software has been developed using the ROS framework that provides access to the RGB-D device and the different image processing tasks: image edge filtering, object detection, recognition and learning, temporal smoothing with the Kalman Filter and the 3D localization of objects.

To develop this work, existing methods for object detection and recognition from RGB cameras have been studied and their improvement using depth cameras. User interfaces based on acoustic signals have also been studied and tested in a group of real users. This BsC Thesis thus represents an opportunity to improve quality of life of visually impaired people by using state-of-the-art computer vision technologies.

Keywords: Device for blind people, RGB-D, binaural technique, Kalman filter, ROS.

Resumen extendido

El presente proyecto final de grado tiene como objetivo el desarrollo de un sistema para la ayuda de personas con discapacidad visual a partir de cámaras RGB-D. El sistema permite reconocer en tiempo real un conjunto de objetos y obtener su posición tridimensional desde el sistema de referencia del usuario. La posición de dichos objetos se transmite al usuario mediante una señal acústica, generada mediante un modelo de localización binaural. Este sistema se encuentra constituido por tres elementos hardware que se detallan a continuación:

- **Cámara RGB-D.** Este dispositivo nos permite obtener de forma simultánea y en tiempo real tanto información de color como información de profundidad para cada uno de los píxeles de las imágenes. En el presente proyecto se ha empleado el dispositivo RGB-D Kinect v1 [1].
- **Sistema de procesamiento.** Permite la ejecución de la aplicación desarrollada sobre el framework de ROS.
- **Sistema auricular.** Nos permite la escucha de los sonidos que permiten la localización del objeto.

La aplicación se ha desarrollado a partir de un software ya existente [2] que se encarga de realizar un análisis del entorno en tiempo real a partir de la información en color. Nos permite la ejecución de las siguientes acciones.

- **Aprendizaje** en tiempo real de todo tipo de objetos, aunque especialmente diseñado para aquellos bajos en texturas.
- **Detección** de los objetos entrenados en tiempo real sobre entornos altamente poblados.

Para la elaboración del proyecto, que será explicado en profundidad en capítulos posteriores, se han desarrollado las siguientes tareas.

1. Mejora de la detección de objetos

Inclusión de bordes debidos a cambios significativos de profundidad obtenidos a partir del sensor de profundidad en el sistema de reconocimiento de objetos desarrollado en [2]. Estos bordes detectados son obtenidos con la nube de puntos que nos proporciona el sensor Kinect v1 a partir de establecer diferencias de profundidad entre píxeles vecinos. La inclusión de nuevos bordes, a los ya obtenidos por el algoritmo original a partir de color, nos permite completar los contornos de los objetos en determinas situaciones en las que estos no pueden ser identificados mediante los contrastes de color.

2. Seguimiento del objeto

Con la finalidad de poder proporcionar más información al usuario, se ha implementado un seguimiento del objeto mediante el Filtro de Kalman. Con ello se pretende poder predecir la posición del objeto en situaciones donde el detector es intermitente en cuanto a detecciones.

3. Obtención de las coordenadas en el espacio

A partir de la posición del objeto en el plano imagen indicada por el Filtro de Kalman y, junto a los datos de profundidad que nos proporciona el dispositivo Kinect, el algoritmo obtiene las coordenadas tridimensionales del objeto en el espacio cartesiano.

4. Localización en el espacio cartesiano

En función de la posición tridimensional que ocupe el objeto respecto a la cámara de color del dispositivo Kinect v1, se reproducirá una señal acústica que permita al usuario localizar el objeto en cuanto a orientación y profundidad. Esto se ha realizado mediante una técnica binaural de generación de señales acústicas.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Memoria

"¿Piedras en el camino? Guárdalas todas, un día construirás un castillo."

1.1 Introducción

Hoy en día más de mil millones de personas padecen algún tipo de diversidad funcional. Perteneciendo a este grupo aproximadamente 285 millones de personas, según datos de la OMS [3], padecen invidencia, ya sea del tipo parcial o total. La sociedad, cada vez más concienciada sobre este hecho, intenta solventar las dificultades a las que se enfrentan las personas con diversidad funcional, ya sea por medio de avances sociales o científicos. Es en estos últimos donde se sitúa el presente proyecto, que tiene como objetivo proporcionar un medio para facilitar el día a día de estas personas.

En concreto, el presente proyecto final de grado se ubica dentro del campo de **la visión artificial y las tecnologías de sensado tridimensional**. Uno de los mayores retos de la visión artificial son los sistemas de reconocimiento de objetos y la obtención de su pose tridimensional en el entorno [4]. En las últimas décadas se han desarrollado enormemente las tecnologías de reconocimiento basadas en el aprendizaje máquina, que consiste en el entrenamiento, a partir de imágenes tomadas de un objeto o categoría, de una función matemática capaz de clasificar y detectar dicho objeto en imágenes [5]. Los métodos desarrollados en la actualidad permiten la detección automática de objetos ante condiciones adversas de iluminación, oclusiones y variaciones en la apariencia del objeto [6].

Otro de los mayores problemas de este campo consiste en la obtención de información tridimensional a partir de imágenes. En las imágenes obtenidas a partir de cámaras convencionales existe una pérdida de información, debido fundamentalmente a la proyección del mundo tridimensional en el espacio bidimensional de la imagen. Las tecnologías existentes de reconstrucción tridimensional a partir de imágenes utilizan información adicional, como puede ser el uso de múltiples imágenes de la misma escena tomadas desde puntos de vista diferentes. Una de las tecnologías de mayor éxito para la reconstrucción tridimensional son las denominadas cámaras de profundidad. Si bien las cámaras convencionales obtienen información de color e intensidad luminosa, estas cámaras nos permiten obtener la distancia entre la cámara y los objetos de la escena para cada uno de los píxeles de la imagen. Existen diversas tecnologías de cámaras de profundidad, destacando las tecnologías de cámaras en tiempo de vuelo o "Time-of-Flight Imaging" (ToF) [7] y las tecnologías basadas en el uso de patrones de luz [8].

Las tecnologías de reconocimiento de objetos y las cámaras de profundidad permiten, por primera vez en la historia, el desarrollo de sistemas portátiles de bajo coste que analizan el entorno tridimensional de forma automática. Esto representa una oportunidad para la aplicación de dichos sistemas a **la ayuda a personas invidentes**. Mediante el uso de estos sistemas es posible la **detección de objetos** de interés en una escena, incluyendo su posición y orientación en el entorno tridimensional. Esta información se puede hacer llegar a una persona invidente utilizando **señales acústicas**, lo cual permite solventar tareas básicas para personas invidentes en entornos inexplorados, como puede ser la búsqueda de diferentes objetos de interés. Estas tecnologías tienen por tanto la capacidad de mejorar la calidad de vida de las personas y presentan además un gran interés científico.

1.1.1 Objetivos generales del trabajo

El objetivo principal de este proyecto final de grado es **proporcionar una solución tecnológica que sirva de ayuda a personas invidentes**. Para tal hecho, se ha desarrollado un sistema que nos permita la detección de objetos en tiempo real indicando de forma precisa su posición respecto a la persona invidente a través de señales acústicas.

El sistema se encuentra compuesto de una cámara RGB-D, la cual permite la obtención de información en color y profundidad, conectada a un sistema de procesamiento (PC portátil), que además cuenta con una salida de audio para la emisión de comandos de voz y sonido. El algoritmo desarrollado puede descomponer en dos etapas diferenciadas:

Entrenamiento. Permite el aprendizaje de los objetos que desean ser detectados en la escena posteriormente, a partir del análisis en tiempo real de una secuencia de vídeo.

Validación. Fase de análisis de la secuencia de vídeo en tiempo real para la detección de objetos, previamente entrenados, en la escena. Una vez detectados, se procede a la emisión de señales acústicas en función de su posición en el espacio respecto al sistema RGB-D.

Este proyecto requiere del conocimiento de los diferentes sistemas y métodos para la detección y reconocimiento de objetos que existen en la actualidad, la mejora de estos y, por último, el estudio y aplicación de las diferentes técnicas de localización mediante sonido binaural.

1.1.2 Organización de la memoria

La presente memoria se organiza en cinco capítulos de la siguiente manera.

- El primer capítulo del presente proyecto es la **memoria**. Se encuentra el estudio teórico y el experimental de todos los conceptos que se han utilizado para lograr el desarrollo del proyecto.
- El segundo capítulo es el **pliego de condiciones**, donde se nos especifican los requisitos tanto de hardware, como de software necesarios para el funcionamiento del sistema desarrollado.
- El tercer capítulo es el **presupuesto**. Sección en la cual se ha calculado el coste que ha supuesto cada etapa del proyecto, elaborando así un presupuesto final del mismo.
- El cuarto capítulo es el denominado **manual de usuario**, para la compresión del funcionamiento de todas las características de la aplicación.
- Finalmente, en el quinto capítulo se han desarrollado las **conclusiones finales** del proyecto.

1.1.3 Estado del arte

Actualmente se han desarrollado una gran cantidad de medios encargados de facilitar la vida a las personas con discapacidad visual. La mayoría de estos sistemas pueden ser clasificados en dos grandes grupos: los encargados de la detección de obstáculos y los encargados de la localización de objetos en entornos altamente poblados.

En cuanto a los primeros, existen desde métodos tradicionales, y quizás, más usados como el bastón blanco cuyo principal inconveniente es el de no permitir una localización en alturas superiores a la cadera, o el perro guía que supone un gran coste y, además, requiere de un gran entrenamiento y coordinación, hasta la existencia de otros métodos más novedosos y con resultados muy satisfactorios, como la ecolocalización [9] que requiere de un gran entrenamiento.

La tecnología actual ha logrado desarrollar numerosos dispositivos para sustituir, mejorar o complementar a los métodos anteriores. Por ejemplo, cabe destacar una nueva versión mejorada del bastón blanco [10] que incorpora sensores para la detección de objetos en alturas superiores a la cintura y la transmisión de los mismos mediante una pulsera vibratoria. El ejemplo citado anteriormente resulta más comercial, pero presenta un elevado coste. Por otro lado, existen otros métodos desarrollados más asequibles basados en la detección mediante sensores ultrasonidos [11].

En cuanto a los medios desarrollados para la localización de objetos en entornos poblados. Clasificación donde se engloba nuestro proyecto, destacan los siguientes.

La cámara "OrCam" [12], diseñada exclusivamente para personas de baja visión. Se trata de un acople a las gafas convencionales que permite diversas funciones de ayuda: como el reconocimiento de objetos en diferentes entornos o la identificación de personas. Además incluye un sistema de transmisión de los resultados mediante comandos de voz. Bien es cierto, que los comandos de voz puede resultar un inconveniente al caminar por la calle. Debido a ello muchos diseñadores optan por otros medios de transmisión menos invasivos como sensores vibratorios, ya sean situados por el cuerpo o en prendas especiales.

Otro ejemplo que se engloba en este marco es el prototipo diseñado por *Gaëtan* Parseihian [13]. Realiza una localización de objetos en el espacio mediante el uso de dos cámaras y una técnica de puntos de disparidad para la obtención de su profundidad. La transmisión de la posición de estos se realiza mediante técnicas auditivas HRTF (Head-Related Transfer Function). Este sistema tiene la misma base que nuestro algoritmo desarrollado. Sin embargo existen diferencias importantes ya que en nuestro sistema se utilizan cámaras RGB-D, que incorporan un sistema de percepción de profundidad más preciso que las cámaras estéreo utilizadas en [13], el empleo de una técnica de localización del sonido diferente y la posibilidad de realizar un entrenamiento en tiempo real sobre cualquier tipo de objeto.

También existen otros métodos mucho más baratos con la única función de reconocimiento de objetos. Se trata de aplicaciones para smartphones capaces de diferenciar diferentes tipos de productos a partir de su forma, sin un funcionamiento en tiempo real [14] o mediante un análisis del código de barras del producto [15].

Nuestro prototipo presenta varias novedades: su precio no es elevado pues sus componentes presentan un coste relativamente bajo. Nos permite un funcionamiento en tiempo real tanto en el

entrenamiento de un objeto, como en la detección de uno o incluso varios objetos en la misma imagen, permitiendo una posterior localización mediante una técnica binaural de uno de ellos.

1.2 Estudio teórico

El estudio teórico llevado a cabo para la realización del presente proyecto, se puede dividir en cinco apartados según su contenido.

- En el primer apartado 1.2.1 se procede a la explicación del funcionamiento de las cámaras RGB-D.
- En el apartado 1.2.2 se describe el método que nos permite el reconocimiento de los objetos y se detallan sus principales características.
- En el apartado 1.2.3 se describen las principales características del framework ROS sobre el que se encuentra desarrollado el sistema de detección utilizado.
- En el penúltimo apartado 1.2.4 se detalla la técnica de audición binaural empleada para la localización del objeto en el espacio.
- Por último, en el apartado 1.2.5 se explica el funcionamiento del filtro empleado para mejorar del sistema original.

1.2.1 Principio de funcionamiento de las cámaras RGB-D

Un gran avance tecnológico en estos últimos años han sido las denominadas cámaras de profundidad, dispositivos capaces de obtener la distancia a la que se encuentra un objeto de una forma extremadamente precisa y rápida. A raíz de ellas surge un nuevo modelo que combina esta tecnología junto con una cámara convencional de color; son las denominadas cámaras RGB-D. Se trata de un dispositivo capaz de obtener simultáneamente y en tiempo real imágenes en color junto con las coordenadas cartesianas para cada uno de sus píxeles. Una de las más destacables gracias a su bajo coste es conocida como “Kinect” [16], desarrollado por la compañía Microsoft con la finalidad de su uso como interfaz de videojuegos en el sistema Xbox.

1.2.1.1 Descripción de Hardware

De forma muy general el sensor Kinect (véase Figura 1.1) está constituido principalmente por una cámara RGB, y una cámara de profundidad. Además de ello, se encuentra constituido por otros elementos como un conjunto de altavoces y un motor de inclinación, pero estos los obviaremos por no ser relevantes en el proyecto.

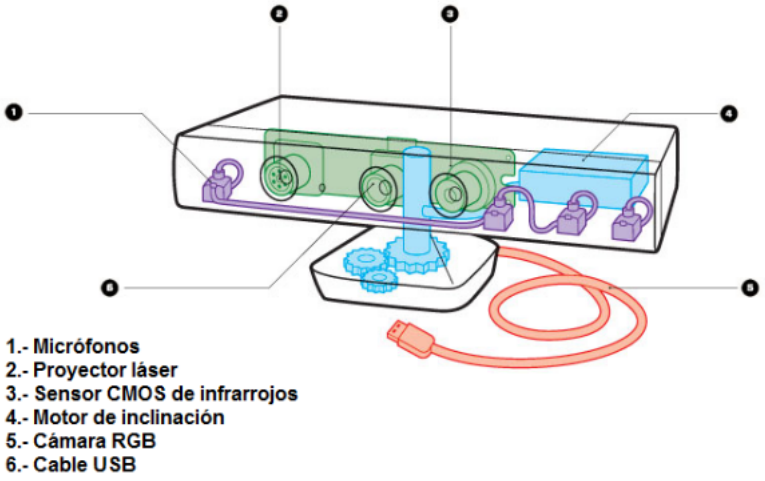


Figura 1.1: Modelo de los elementos que componen el dispositivo Kinect v1. [16]

Proyector láser. Se trata de un láser con la capacidad de emitir un único rayo, a un nivel constante (no produce pulsos) con una longitud de 830nm, el cual es subdividido mediante un sistema de difracción que proyecta un patron de luz estructurada sobre la escena. Incorpora un medidor de temperatura para garantizar esta constante y que no afecte así a la longitud de onda de salida.

Sensor de infrarrojos. Se trata de un sensor CMOS monocrono de 1/2' MT9M001C12STM, de marca Micron. Con un tamaño de píxel de 5,2um, 30 fps programables, un rango de temperatura entre 0-70ºC, un campo de visión horizontal de 57º y vertical de 43,5º.

Cámara RGB. Se trata de un sensor CMOS MT9M112. Con un tamaño de píxel de 2,8umx2,8um, 1,3 Megapíxeles, filtro de color Bayer, una resolución de 30fps a 640x512 y un rango de temperaturas de -30ºC a +70ºC.

Sobre los dos últimos componentes no se conoce su modelo real ya que la compañía no ha proporcionado esta información. Éstos han sido identificados mediante ingeniería inversa. [16]

1.2.1.2 Cálculo de la profundidad

El sistema de percepción de Kinect v1 nos proporciona información del tipo 2.5D, ya que se encuentra capacitado para proporcionar medidas de profundidad sin permitir una observación de los objetos en su plano posterior. Para esta tarea consta de los siguientes elementos: un proyector láser infrarrojos, un sensor CMOS y un microchip para el procesamiento de la información. Kinect utiliza un **procedimiento de luz estructurada** para la extracción de la información de profundidad. Es importante decir que esta tecnología de sensado es completamente diferente a un sensor basado en tiempo de vuelo (ToF). Las cámaras (ToF) admiten dos técnicas para la adquisición de datos de profundidad: el primer método, utilizado en la nueva versión de Kinect, se basa en la medición de las diferencias de fase entre la luz emitida y recibida y es el método más extendido por su relación calidad y precio. El segundo método se basa en la medición del tiempo que emplea la luz emitida por el sensor en impactar la escena y ser capturada de nuevo por el sensor.

El láser se encarga de la emisión de un rayo único, el cual se difracta para formar el patrón que es proyectado sobre el entorno. Mediante el sensor CMOS se realiza una captura de las variaciones

de éste patrón en el espacio y mediante un **proceso de triangulación** se obtiene la información de profundidad de cada elemento del patrón, que se extiende por interpolación a cada píxel.

Proceso de triangulación.

El sistema Kinect tiene incorporado en su memoria un patrón de referencia de la luz emitida, a partir del cual realiza el proceso de triangulación. En función de la posición que ocupe un objeto respecto a ese plano de referencia, el patrón se distorsiona hacia una determinada dirección. Para medir esos desplazamientos que nos permitan realizar la triangulación, y por tanto obtener así sus medidas, es necesario establecer una correlación entre los puntos de la imagen y el patrón de referencia.

El plano de referencia (véase Figura 1.2) se obtiene mediante la difracción del láser que produce un patrón de puntos aparentemente aleatorio. El patrón de referencia se obtiene a partir de la medición de un plano del que se conoce la profundidad, y permanece almacenado en la memoria de la cámara desde su proceso de fabricación.

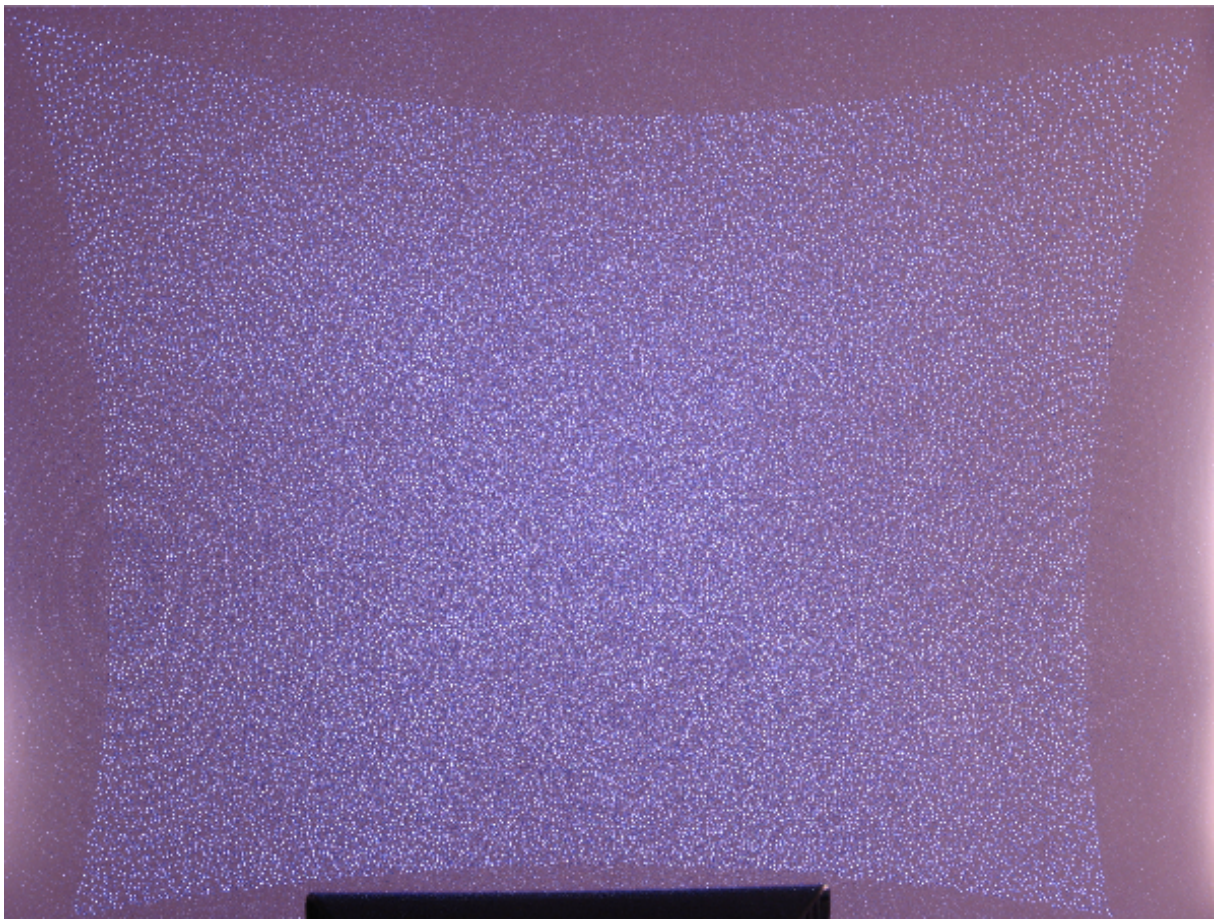


Figura 1.2: Plano de referencia de Kinect v1 capturado a partir de una cámara de infrarrojos.[16]

El proceso de triangulación se detalla en la figura 1.3. En primer lugar uno de los rayos infrarrojos provenientes el proyector láser "L" pasará por el objeto en cuestión "k", que se encuentra situado sobre el plano objeto. Ese rayo tendrá una proyección del objeto sobre el plano de referencia "o". Visto desde la cámara de infrarrojos "C", punto de origen de nuestro sistema, el punto "o", ocupará otra posición distinta (en este caso más a la izquierda del objeto real) en el plano objeto. Ésta distancia "D"

sobre el plano del objeto, produce una distancia de disparidad "d" sobre el espacio de la imagen, y que será utilizada para la obtención de la distancia Z_k a la que se encuentra el plano objeto, a partir de las siguientes expresiones obtenidas mediante relaciones trigonométricas.

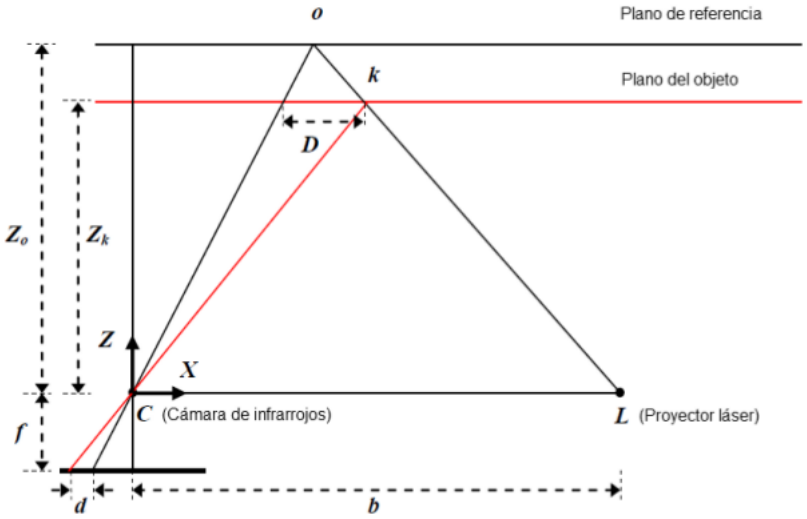


Figura 1.3: Ejemplo del proceso de triangulación para la obtención de la profundidad [16].

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o}$$

$$\frac{d}{f} = \frac{D}{Z_k}$$

$$Zk = \frac{Z_o}{1 + \frac{Z_o}{fb} d}$$

Los parámetros del plano de referencia "Z_o", la distancia focal del sensor infrarrojo "f" y la distancia entre el sensor y el proyector "d" se determinan mediante el proceso de calibración. A partir de ello podemos obtener las coordenadas cartesianas del objeto.

$$\begin{cases} X_k = -\frac{Z_k}{f}(x_k - x_o + \delta_x) \\ Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta_y) \\ Z_k = Z_k \end{cases}$$

Siendo (x_k, y_k) las coordenadas en la imagen del punto, (x_o, y_o) el offset de la imagen y (δ_x,δ_y) las correcciones de la distorsión de la lente.

1.2.1.3 Sistema de captura RGB

La cámara integrada en Kinect utiliza un **único sensor CMOS** para la captura de información RGB. La función del **sensor CMOS** es acumular carga eléctrica en cada uno de los píxeles en función de la luz que recibe. La característica principal de este tipo de sensores es que cada celda es totalmente independiente del resto, y cada celda tiene su propio procesamiento analógico-digital, lo cual reduce costes y hace al equipo menos robusto y más rápido respecto a sensores basados en otras tecnologías. Cada píxel del sensor se especializa en la captura de uno de los tres canales de color (rojo, verde y azul) en una disposición de patrón de Bayer (ver figura 1.4). Este patrón alterna filas de los colores

primarios de manera no equitativa, pues repartirá tantos píxeles verdes como azules y rojos juntos. Esto es debido a que el ojo humano no es sensible por igual a estos tres colores y es necesario incluir mayor información de píxeles verdes para una mejor percepción del color. La imagen de color, donde cada píxel se asocia a los tres valores RGB, se obtiene mediante una interpolación que se realiza tras la captura.

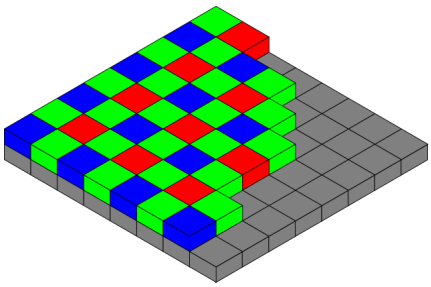


Figura 1.4: Sensor CMOS.

1.2.2 Detección de objetos

La detección de objetos utilizada en este proyecto se basa en el sistema **"Real-time Learning and Detection of 3D texture-less objects: A Scalabe Approach"** desarrollado por Walterio Mayol-Cuevas, Dima Damen, Pished Bunnun y Andrew Calway [2]. Se trata de un método escalable con la finalidad del aprendizaje y la detección de múltiples objetos en una secuencia de vídeo en tiempo real. Primero se realiza un entrenamiento de diferentes objetos, para posteriormente detectarlos en ambientes más poblados.
La principal característica de este método es que permite la detección de objetos cuya apariencia tiene una textura pobre y que se contrastan con la escena gracias a la presencia de bordes en la imagen. Este tipo de objetos se encuentran con facilidad en entornos creados por el ser humano y ofrecen una dificultad añadida para muchas alternativas existentes para la detección de objetos en imágenes.

1.2.2.1 Estado del arte

Para caracterizar el estado del arte en el que se enmarca este algoritmo se propone un análisis en tres fases: en primer lugar se describen los métodos existentes de detección de un único objeto o clase. Posteriormente, se realiza una recopilación de información referente a los métodos que permitan una mejora escalar del algoritmo para la detección de múltiples objetos y, por último, se presentan aquellos métodos que permiten la detección en tiempo real de objetos en una secuencia de vídeo.

Existen numerosos métodos basados en la **búsqueda de un objeto a partir de su apariencia**. Los algoritmos desarrollados que modelan la apariencia del objeto a partir de la textura tienen una gran eficacia tanto para objetos planos como para objetos 3D. Sin embargo estos métodos presentan problemas cuando se aplican a objetos bajos en texturas. Este tipo de objetos cuentan con métodos propios que hacen uso de **la forma o formas del objeto** a partir de los segmentos de borde en la imagen. En estos métodos se realiza una descripción y posterior detección de la forma del objeto a partir de: fragmentos de contornos de los objetos ([17], [18]), una búsqueda de bordes por radios [19] o a partir de agrupaciones de "edgelets" (segmentos de bordes) no adyacentes. Este último principio es en el que se basa nuestro método, la búsqueda de constelaciones de "edgelets" no adyacentes en objetos. Dos propuestas similares a la utilizada en este proyecto son las siguientes:

[20] Este método aprende, de entre todos los pares de "edgelets" que surgen del modelo, el par más consistente que posteriormente es comparado con todos los pares en las imágenes test.

[21] Este método aprende agrupaciones completas de "edgelets" que son generadas a partir de una gran cantidad de descriptores. Después en la fase de detección se buscan agrupaciones de la misma manera y, se comparan con las agrupaciones ya aprendidas.

Uno de los principales objetivos en los métodos de reconocimiento de objetos en imágenes es conseguir que el método propuesto resulte invariante a transformaciones (escala, rotación y translación) y, que además sea escalable según el número de objetos aumenta. Los primeros trabajos desarrollados se enfocaban en la indexación y búsqueda de geometrías a partir de, en primer lugar, puntos de la imagen [22] y superficies [23]. Posteriormente se desarrollaron estos métodos para bordes, los cuales son agrupados y representados como descriptores ([24],[25]) que resultan invariantes. Estos descriptores se agrupan para todas las vistas entrenadas y, en la fase de detección, se generan de la misma manera para que luego sean comparados con los ya aprendidos.

A continuación, se procede a un análisis de los métodos que permitan una **detección de múltiples objetos** de una manera que no empeore el funcionamiento del algoritmo. Los métodos más empleados en visión se basan en establecer jerarquías de las vistas o de los objetos, con la finalidad de no tener que testear todos los objetos almacenados, que son agrupados desde un nivel global a un nivel particular. Para objetos 3D, cuyas vistas pueden llegar a ser muy diversas, existen modelos que proponen jerarquías basadas en las formas [26]. También existen otros modelos que emplean otro tipo de jerarquías basadas en detectores [27]. Estos métodos caracterizan un objeto mediante el uso de diferentes sistemas de detección que actúan simultáneamente y, por tanto, pueden almacenar diferentes características de una misma vista.

Por último, en cuanto a métodos empleados para la **detección de objetos a partir de entradas de vídeo**, se referencia a dos propuestas. En [28] se propone una mejora basada en las transformaciones en las distancias 3D y direcciones de las imágenes. Por otro lado, en el trabajo descrito en [29] se proponen parches formados por histogramas de orientaciones dominantes, que permite un funcionamiento en tiempo real tanto para aprendizaje como para detección. El inconveniente, es que ambos métodos al ser utilizados con más de un objeto, empeoran sus tiempos de detección notablemente y no son invariantes ni a cambios de escala y rotación.

El método que se propone en este proyecto, y que se basa en el detector de objetos propuesto en [2] es altamente escalable con el número de objetos, admite funcionamiento en tiempo real y la detección de objetos con una textura pobre. A continuación se describe dicho algoritmo con detalle.

1.2.2.2 Descripción del algoritmo

Este método está **basado en describir la forma o formas del objeto**, las cuales son representaciones generadas a partir de bordes o segmentos de bordes, también conocidos como "edgelets".

A partir de un mapa de bordes (w) generado sobre una imagen tenemos un conjunto de "edgelets" $\{e_i\}$, fragmentos de borde representados por un punto central y una orientación. Los conjuntos de ellos nos permite definir partes locales o globales de las diferentes formas que se encuentran en la imagen. Por cada imagen existe una gran cantidad de "edgelets", con lo que se debe utilizar un método que permita analizarlos de una manera rápida.

"Path tracing" es el método empleado, tanto en el entrenamiento como en la detección. Este método hace uso de "paths" predefinidos (véase Figura 1.5), los cuales son una secuencia de ángulos $\Theta = (\theta_0, ..., \theta_{n-2})$ utilizada para la búsqueda de conjuntos de "n-edgelets" $c = (e_1, ..., e_n)$ sobre la imagen. Estos "paths" se han predefinido según su habilidad para la búsqueda de constelaciones de "edgelets" en una secuencia de entrenamiento con multiples objetos.

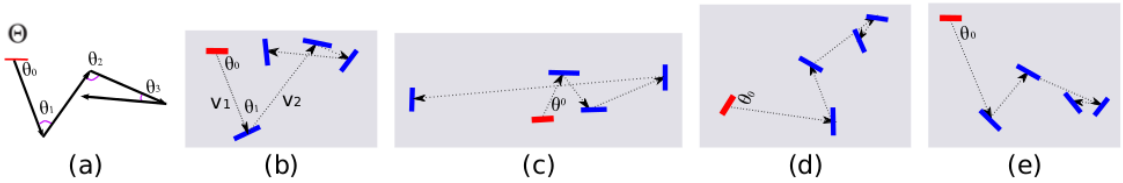


Figura 1.5: Con un mismo path (a), se pueden generar diferentes constelaciones como (b), (c), (d) y (e) [2].

El método para definir una constelación de "edgelets" a partir de un path es la siguiente: se parte de un "edgelet" cualquiera e_1, y en función del primer ángulo θ_0 del path empleado (se seleccionan de una manera aleatoria) generamos un vector v_1 "infinito" con la orientación respecto al "edgelet" seleccionado. Si este vector se cruza en algún momento con otro "edgelet" e_2, este formará parte del conjunto de "edgelets" buscado. A continuación, se vuelve a generar otro vector v_2 según el siguiente ángulo del path e_2, pero con una orientación relativa al vector anterior v_1 (no respecto al "edgelet"). Si este vector se vuelve a cruzar con otro nuevo, se añadirá este también a la constelación siguiendo el proceso hasta que se obtengan los "n-edgelets" que formen la constelación.

Una vez tenemos la constelación c_1 generada se obtiene un descriptor $f(c_i) = (\phi_0, ..., \phi_{n-1}, \delta_1, ..., \delta_{n-2})$. El cual contiene la información relativa entre edgelets consecutivos sobre: su orientación ($\phi_i = \widehat{e_i, e_{i+1}}$) y sobre su distancia ($\delta_i = |v_{i+1}|/|v_i|$). Debido a almacenar datos relativos entre "edgelets", estos descriptores resultan invariantes a translación, escala y rotación. Empleando un mismo path, pueden surgir gran cantidad de descriptores sobre la misma vista, pero es suficiente con encontrar una constelación en el objeto para encontrar un posible candidato, que es verificado usando el resto de "edgelets" (véase Figura 1.6). En la fase de entrenamiento estos descriptores se van almacenando en una librería en función del path que se ha empleado de una forma jerárquica, ya que en la fase de detección el descriptor se va comparando de forma gradual con los almacenados en la librería.

Cuando se encuentra una coincidencia, se calcula su transformación afín H a partir de los "n-edgelets" que forman la constelación de la imagen test. A partir de H se transforman todas las vistas de los "edgelets" almacenados de la correspondencia a la imagen test y se redefinen con el resto de "edgelets" cercanos, donde la distancia entre los dos "edgelets" $d(e_i, e_j)$ depende de tanto la orientación (ori) como la posición (pos) según la siguiente expresión 1.1.

$$d(e_i, e_j) = |e_i.pos - e_j.pos|_2 + \lambda |e_i.ori - e_j.ori| \tag{1.1}$$

El coste de la detección (E) es la media escalar entre las distancias entre "edgelets", definida por la expresión 1.2. Donde $H(e_i)$ es la transformación del "edgelet" e_i bajo su transformación afín H, $\tau(e_i, H)$ es el "edgelet" más cercano a la transformación del edgelet e_i. El término "R" es el radio de la

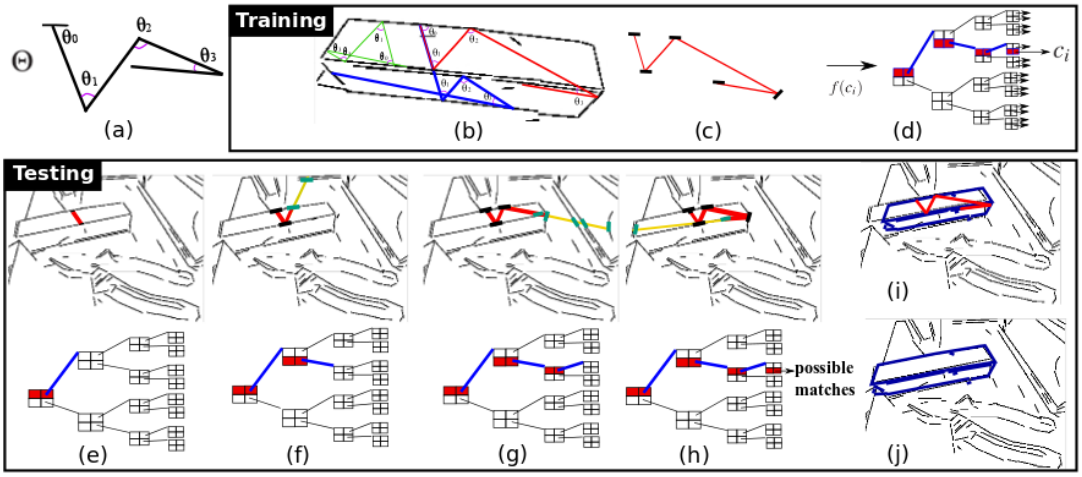


Figura 1.6: Con un mismo path (a), se pueden generar diferentes constelaciones sobre una misma vista (b). A partir de la constelación se obtiene un descriptor (c) que es almacenado en la librería (d). En la fase de detección según se generan los descriptores van siendo comparados con los aprendidos (e), (f), (g), hasta encontrar una coincidencia (h). Se realiza una homografía de las vistas de los "edgelets" aprendidos a la imagen test (i). Que es posteriormente es redefinida con los "edgelets" cercanos (h) [2]

longitud de las vistas de los "edgelets" para la imagen test.

$$E(w,H) = \frac{\Sigma_i \min(d(H(e_i),\tau(e_i,H)),\beta)}{|w|} \frac{|\tau(e_i,H) : d(H(e_i),\tau(e_i,H)) \leq \beta|}{|e_i : d(H(e_i),\tau(e_i,H)) \leq \beta|} R \qquad (1.2)$$

1.2.2.3 Estrategia para implementación eficiente

El método emplea durante el proceso de detección cuatro técnicas para que su funcionamiento no se vea ralentizado, permitiendo así que su funcionamiento sea en tiempo real.

- 1. Se precalcula la orientación relativa, dirección y posición entre todos los pares de "edgelets" en la imagen test. Todos los pares que satisfagan el ángulo base (θ_0) del path dentro de una tolerancia, se tienen en cuenta para la comprobación del resto de pares.
- 2. De todas las constelaciones que pueden surgir utilizando un determinado path a partir de un par edgelets, sólo una es completada. Sabiendo que la cantidad de constelaciones que se encuentran almacenadas para un mismo path es grande, el riesgo de saltarse un par que sea coincidente es pequeño.
- 3. Según se va generando el descriptor detectado, éste va siendo comparando con la librería de los descriptores almacenados. Cuando no se encuentra ninguna coincidencia, esta búsqueda se detiene y se continua con otro par.
- 4. Cuando se encuentra una coincidencia (con un error dentro de tolerancia) los bordes son eliminados para proceder a la búsqueda de nuevos objetos sobre la misma imagen.

Teniendo en cuenta que este método trata cada frame de forma independiente, actúa de la siguiente manera: cuando todo los pares de "edgelets" han sido testados en un path y no hay coincidencias, se pasa a otro path hasta encontrar una coincidencia. Esto sucede hasta que la búsqueda alcanza un tiempo máximo, momento en el que se detiene al ver difícil encontrar una coincidencia y procede al análisis del siguiente frame.

1.2.2.4 Características Principales

Este método de detección de objetos basado en las formas posee numerosas características que deben ser destacadas:

- Funcionamiento en tiempo real tanto para la fase de entrenamiento como para la fase de detección.
- Permite el entrenamiento de hasta un total de 30 objetos diferentes.
- Método invariante a rotación, escala y translación.
- Detección robusta ante pequeñas oclusiones.
- Permite detectar varios objetos de distinta clase en la misma imagen

Con la finalidad de comprender mejor su funcionamiento se procede al análisis por separado de las dos fases de las que consta el algoritmo. En cada una de ellas se detallarán las características antes mencionadas según la fase en la que tenga presencia su efecto.

Fase de entrenamiento

El entrenamiento de los objetos se realiza en tiempo real y nos permite entrenar hasta un máximo de 30 objetos sin afectar el funcionamiento del algoritmo. En la figura 1.7 se muestra un ejemplo de aprendizaje en tiempo real.

(a) Frame 1 (b) Frame 2 (c) Frame 3

Figura 1.7: Fase de entrenamiento.

Como se mencionó en el apartado anterior 1.2.2.2, el detector se encarga de generar los descriptores a partir de las constelaciones encontradas. Estos son aprendidos para posteriormente en la fase de detección comparar los obtenidos con los que se encuentran almacenados. Pueden surgir para una misma vista del objeto diferentes descriptores, ya que estos dependen del path empleado, que es aleatorio, y del edgelet sobre el que comience la búsqueda (véase Figura 1.8).

(a) Path 1 (b) Path 2 (c) Path 3

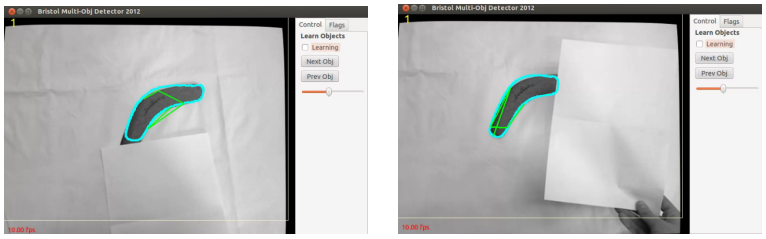
Figura 1.8: Generación de diferentes constelaciones (líneas verdes) para un mismo frame.

Fase de detección

La detección también tiene un funcionamiento en tiempo real y resulta invariante a rotación (véase Figura 1.9), translación y escala. Además, también se permite la detección de objetos parcialmente ocluidos (véase Figura 1.10).

(a) Rotación 1 (b) Rotación 2 (c) Rotación 3

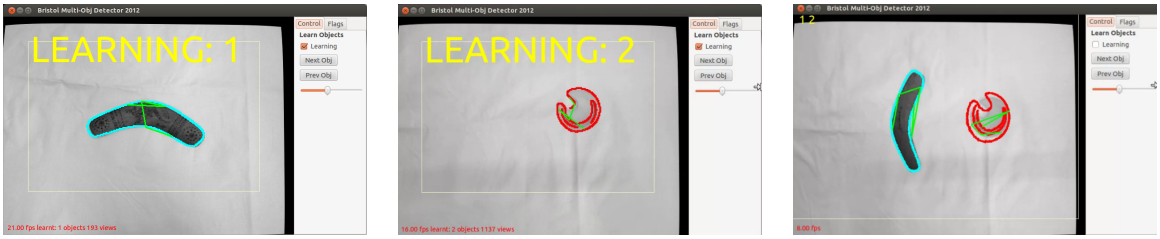
Figura 1.9: Ejemplo de demostración de que el algoritmo resulta invariante a rotación.



(a) Oclusión 1 (b) Oclusión 2

Figura 1.10: Ejemplo de demostración de que el algoritmo resulta invariante a oclusiones.

Debido a que se puede realizar el entrenamiento de más de un objeto, se pueden dar situaciones en las que en un mismo frame estén situados varios de ellos. En estas situaciones, también se permite la detección de varios objetos de manera simultánea (véase Figura 1.11).

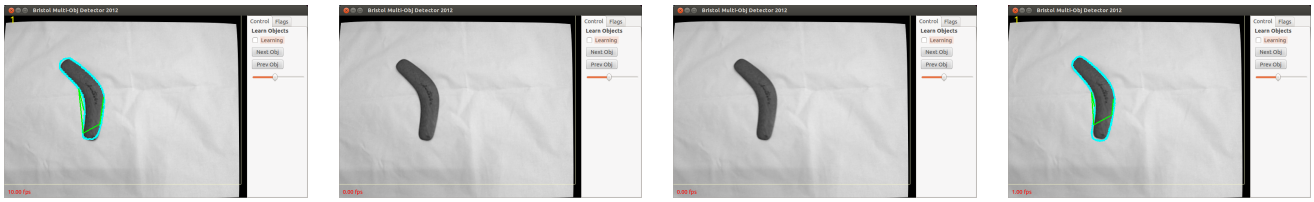


(a) Entrenamiento objeto 1 (b) Entrenamiento objeto 2 (c) Múltiple detección

Figura 1.11: Ejemplo de demostración de que el algoritmo esta capacitado para múltiples detecciones simultáneas.

Los principales inconvenientes que se han observado en el funcionamiento de este algoritmo son los siguientes:

- La detección de objetos resulta de forma intermitente. Como podemos apreciar (véase Figura 1.12) entre dos detecciones seguidas tenemos dos frames en los que la detección ha fallado. Una de las razones que lleva a esta situación, es la manera en la que se encuentra desarrollada el algoritmo, pues trata cada frame de manera independiente.



(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

Figura 1.12: Fase testeo.

- Debido a que la extracción de bordes se realiza a partir de imágenes color se dan diversas situaciones en las que no existe un claro contraste de color entre determinados objetos de la imagen. Es en estas situaciones cuando no se detectan determinados bordes, y hace que se pierda gran cantidad de información tanto en la fase de entrenamiento como detección.
- Otro inconveniente, resulta la escasa detección de objetos cuando se encuentran en entornos más poblados, esto se analizará posteriormente en la sección 1.3.3.1.

1.2.3 ROS

Robot Operating System [30], más conocido por sus siglas "ROS", se trata de un **framework** distribuido en código abierto utilizado para el desarrollo de software. Como cualquier framework no es más que un conjunto de librerías, las cuales nos permiten el desarrollo de aplicaciones para ciertos ámbitos como el control robótico o la visión artificial, siendo éste último el ámbito utilizado en este proyecto.

ROS se desarrolló en 2007 bajo el nombre de "switchyard" por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del robot STAIR2. Sin embargo, desde 2008 su desarrollo continúa por el instituto robótico "Willow Garage". La versión de ROS que ha sido utilizada es conocida como "ROS fuerte".

Podemos decir que ROS consta de varios elementos principales: Paquetes, nodos y stacks o pilas.

1.2.3.1 Paquetes

Se trata de la unidad principal de ROS que nos permite la agrupación en serie de las aplicaciones, facilitando la portabilidad e instalación entre diferentes equipos. Los elementos que componen un paquete son los siguientes.

CMakeList.txt Contiene las instrucciones necesarias para construir el paquete: Nombre del paquete, rutas de las librerías, generación de ejecutables, etc.

Makefile Encargado de hacer construir el paquete.

manifest.xml Incluye las licencias, identificación del autor y breves descripciones puntuales. Además su principal función es incluir las dependencias necesarias para construir el paquete. En este caso unos ejemplos representativos de algunas dependencias utilizadas son:

- PCL1.7 [31].
- CVD.
- Soundplay.
- OpenCV [32].

1.2.3.2 Stack o pilas

Colección de paquetes que comparten una misma finalidad. Dentro de ellos podemos encontrar los ficheros que contienen los metadatos encargados de proporcionar la información para que estos funcionen correctamente.

1.2.3.3 Nodos

Unidades de computación encargadas de realizar una tarea simple, es decir, se trata de las aplicaciones desarrolladas por el usuario para llevar a cabo una acción. Deben estar definidas en lenguaje C++ o Python, aunque ROS permite la implementación también del lenguaje Java. Una de las mayores ventajas de los nodos es el hecho de proporcionarnos una tolerancia adicional a los errores, de manera que los incidentes que ocurran se pueden identificar de forma aislada en nodos individuales en vez de en un programa más complejo.

Destacar el **nodo máster** (véase Figura 1.13) el cual realiza diversas funciones principales:

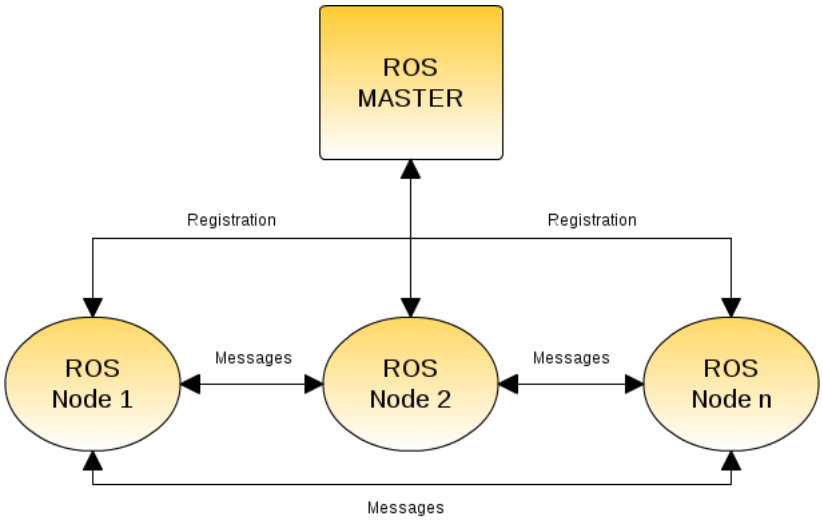


Figura 1.13: Comunicación de ROS entre los diferentes nodos y el nodo máster.

- Llevar a cabo el registro de los nodos que se están ejecutando.
- Permitir la comunicación entre nodos. Sin el nodo máster, el resto de los nodos no son capaces de encontrar los mensajes o invocar los servicios. Por tanto imposibilita el intercambio de información.

Para iniciar el nodo máster, es necesario ejecutar la siguiente orden en el terminal.

```
$ roscore
```

Si durante la ejecución de un programa se desea ver los nodos activos, basta con ejecutar la siguiente orden.

```
$ rosnode list
```

Comunicación entre nodos.

La comunicación entre nodos es algo fundamental en ROS pues nos **permite la creación de una aplicación mas compleja**. La comunicación se realiza de una forma muy sencilla, y puede

desde transmitirse a través de estructuras primitivas (floats, characters...) hasta mas complejas como combinaciones de las anteriores. Existen dos formas de comunicación:

Topic. Son como buses a través de los cuales se intercambian los mensajes entre nodos. Se enrutan en dos formas: Publicadores o Subscriptores. Cada nodo puede publicar y subscribirse a uno o varios topics de una manera periódica, con el único impedimento que un nodo publicador nunca tendrá información del subscriptor o los subscriptores (véase Figura 1.14) ,es decir, se encuentran diseñados para trabajar de una manera unidireccional. Durante la ejecución de un programa se puede obtener una lista sobre los topics activos con el siguiente comando.

```
$ rostopic list
```

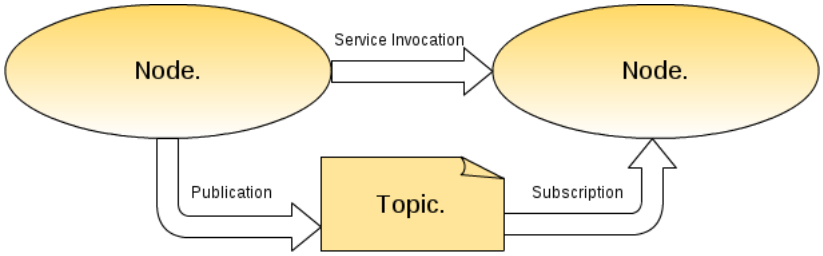


Figura 1.14: Método de comunicación entre nodos a partir de publicaciones y subscripciones.

Servicios. Similares a los "Topic" pero estos sí tienen información de cuando el subscriptor recogió la información, además permite un retorno de la información, es decir, permiten un intercambio de información bidireccional. Al igual que con los "Topics, también podemos ver la lista de servicios activos ejecutando el siguiente comando en el terminal.

```
$ roservice list
```

1.2.4 Localización binaural

La localización del sonido se refiere a la capacidad del oyente para poder identificar el origen de un sonido en dirección y distancia. También se utiliza para definir los métodos que se usan en la ingeniería acústica para la simulación del posicionamiento de señales auditivas en espacios tridimensionales.

El sistema auditivo de los mamíferos necesita de varias señales sonoras para poder identificar el posicionamiento de una fuente, es incapaz de identificarla mediante una única señal. Éstas señales deben incluir diferencias en relación a su intensidad, tiempo o información espectral entre la llegada a ambos oídos, excepto si la fuente se encuentra en una posición equidistante a ambos. Por ello se denomina **audición biaural**, pues necesitamos de los dos oídos para que se establezcan estas diferencias. El cerebro procesa las señales auditivas que recibe y tiene la capacidad de identificar la fuente en cuanto a su dirección, ya sea en el plano horizontal o de azimut o en el plano medio o de elevación y, además, permite el reconocimiento en función de la profundidad.

En el presente proyecto asumimos la siguiente referencia de planos (véase Figura 1.15), los cuales se encuentran referenciados sobre la cámara RGB de nuestro sensor Kinect.

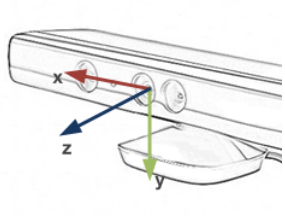


Figura 1.15: Sistema de referencia de Kinect.

- Plano horizontal queda definido por los ejes X-Z.
- Plano medio queda definido por los ejes Y-Z.
- Plano frontal queda definido por los ejes X-Y.

Además, como simplificación, asumiremos la fuente del sonido como ideal y siempre situada en campo lejano, donde se cumple la ley de disminución de la amplitud sonora con la inversa del cuadrado de la distancia entre la fuente y el receptor. En el sistema propuesto se utilizan auriculares para la escucha de las señales sonoras, ya que el hecho de reproducirse mediante un altavoz puede provocar el distorsionamiento de las señales antes de llegar al oído y, por tanto, interpretar el sonido de forma errónea.

1.2.4.1 Localización en plano horizontal

Para la ubicación en el plano horizontal de una fuente el cerebro utiliza pistas provenientes principalmente entre las **diferencias de intensidad y de tiempo** que se dan entre las señales. En función de la frecuencia del sonido una u otra resulta más efectiva. Nuestra referencia en este plano vendrá definida de la siguiente manera. (véase Figura 1.16).

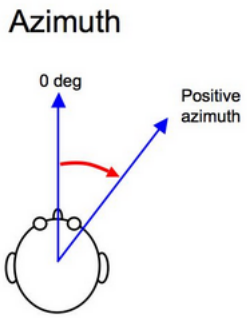


Figura 1.16: Plano de referencia horizontal.

Diferencia interaural de tiempo (DIT).

Se trata de la diferencia de tiempos de llegada de un sonido respecto a ambos oídos. Al tratarse de una diferencia de tiempo, su unidad de medida es en segundos. Su valor puede ser calculado a partir de la diferencia que se establece entre las distancias que deben recorrer ambas ondas hasta llegar a los oídos junto con la velocidad del propio sonido.

La diferencia en las distancias que deben recorrer ambas ondas se obtiene mediante razones trigonométricas básicas (véase Figura 1.17). Esta diferencia viene determinada por la expresión: $\phi_{head}(sin\theta + \theta)$,

que junto con la velocidad del sonido nos define la fórmula que caracteriza este fenómeno (Ecuación 1.3) en la que "a" se define como el radio de la cabeza y "c" como la velocidad del sonido. [1]

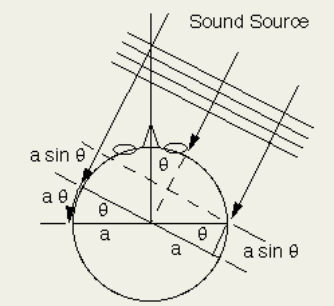


Figura 1.17: Diferencia de la distancia recorrida por un sonido entre ambos oídos.

$$ITD = \frac{a}{c}(\theta + sin\theta) \tag{1.3}$$

A partir de la fórmula mencionada en el párrafo anterior se define la siguiente gráfica (Gráfica 1.18) para cada uno de los ángulos posibles. Como se puede observar su valor llega a variar desde 0 segundos para fuentes sonoras con un ángulo de 0º, es decir, situadas justo frente al individuo en la cual no se puede establecer ninguna diferencia en tiempos de llegada, hasta aproximadamente un tiempo máximo de 0.7ms, en función de la persona y las condiciones, para fuentes sonoras situadas a $\pm 90^o$, punto en el que la diferencia resulta máxima puesto que la onda más tardía debe recorrer todo el diámetro de la cabeza (Ecuación 1.4).

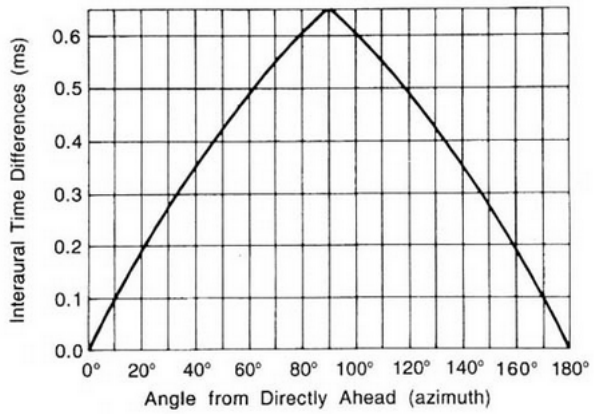


Figura 1.18: Gráfica de la diferencia interaural de tiempos (ITD) [33].

$$ITD_{\max} = \frac{\phi_{\text{head}}}{V_{\text{sound}}} = 0{,}695ms \tag{1.4}$$

Sin embargo, en el presente proyecto hemos empleado la fórmula definida por Kuhn en 1977. Definida tras su análisis del ITD dependiente de la frecuencia en función de la difracción de la onda plana, la cual supone una simplificación de la ecuación que define el ITD a la siguiente expresión (Ecuación 1.5), para frecuencias bajas (inferiores a 500Hz) como las empleadas.

$$ITD = \frac{3a}{c}sin(\theta) \tag{1.5}$$

Existe una ambigüedad para sonidos senoidales en los que la longitud de onda del sonido es menor o

[1] Siempre se ha supuesto un promedio de diámetro de la cabeza de 22cm y una velocidad del sonido constante de 345m/s.

comparable con la distancia entre ambos oídos, es decir para frecuencias superiores a aproximadamente 1.6kHz (Ecuación 1.6).

$$f = \frac{V_{\text{sound}}}{\lambda} = \frac{V_{\text{sound}}}{\phi_{\text{head}}} = 1568{,}18Hz \tag{1.6}$$

Esto es debido a que las diferencias de fases pueden ser tales que existan ciclos enteros de diferencia entre ambos oídos, lo que puede hacer que nuestro sistema auditivo no pueda determinar a que ciclo corresponde cada uno. Además también se generan confusiones en frecuencias cuya mitad de longitud de onda es comparable con la dimensión del diámetro de la cabeza, es decir para frecuencias entorno a 800Hz, o múltiplos de ella.

Existen dos modelos que nos permiten conocer mejor la forma con la que el cerebro interpreta las diferencias entre tiempos para la localización de la fuente.

Modelo de Jeffress. Modelo hipotético propuesto por Jeffress que indica como las neuronas actúan en el cerebro para pequeñas diferencias de tiempo (véase Figura 1.19). Para visualizar un vídeo explicativo sobre el funcionamiento del modelo, consultar la siguiente referencia [34].

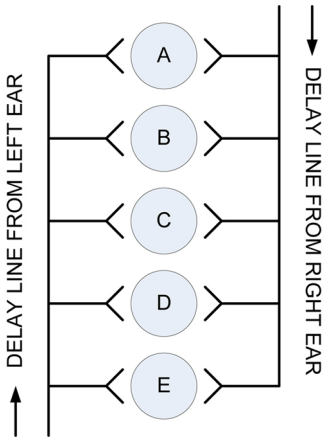


Figura 1.19: Modelo de Jeffress.

Efecto Haas. Descrito por Helmut Haas, quién da nombre a este curioso efecto, establece que si dos sonidos llegan a ambos oídos con un retardo inferior a 50 ms, el cerebro los interpretará como un único sonido [35]. De aquí surgen dos posibles interpretaciones.

- Si la diferencia entre los sonidos es inferior a 5ms, el cerebro localiza la dirección en función del primer estímulo.
- Si la diferencia se establece entre 50ms y 5ms, el cerebro los sumará como uno único de doble intensidad y localiza la fuente a medio camino entre todas.

Diferencia interaural de intensidad (DII)

Las diferencias interaurales de intensidad se dan fundamentalmente por dos hechos: La distancia que deben recorrer las ondas para llegar de un oído a otro y por la sombra acústica producida por la cabeza del oyente debido a la difracción que se provoca alrededor de esta hasta llegar al oído alejado.

Cuando la longitud de la onda es suficientemente pequeña respecto al obstáculo con el que se encuentra, se produce una mínima difracción de ésta y, por lo tanto, se genera mayor sombra acústica.

Mientras que si la longitud de onda es suficientemente grande respecto al obstáculo, la onda se difracta, generando así poca sombra acústica (véase Figura 1.20). Por tanto para el DII, es fundamental trabajar con altas frecuencias ya que se genera mayor sombra acústica, y por tanto, permite una mejor diferenciación de las señales y así permite que la localización resulte más efectiva. La diferencia del nivel

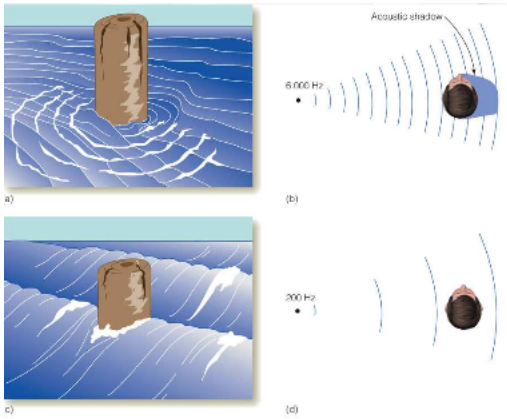


Figura 1.20: Sombra acústica generada por la cabeza en función de la frecuencia del sonido. A mayor frecuencia, mayor sombra acústica.

de intensidad sonoro que hay que establecer entre ambas señales, como ya ha sido mencionado depende de la frecuencia. A partir de la gráfica IID (Gráfica 1.21), podemos establecer a que ángulo queremos simular el posicionamiento virtual de nuestra fuente.

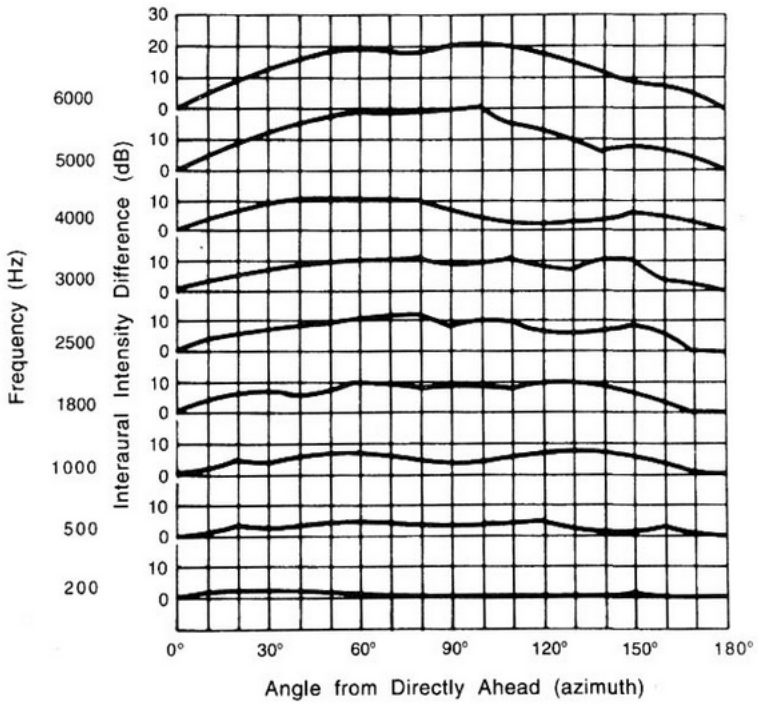


Figura 1.21: Gráfica de la diferencia interaural de intensidad (IID) [33].

El nivel de intensidad de una onda depende de su potencia y superficie (Ecuación 1.7). Sabiendo que la potencia de una onda depende de la amplitud al cuadrado, y que la superficie resulta igual para ambas señales, podemos establecer que la intensidad será en únicamente dependiente del término de la

amplitud.

$$I = \frac{P}{S} \tag{1.7}$$

Una vez obtenida la intensidad del sonido podremos obtener su nivel de intensidad (Ecuación 1.8). Posteriormente, deberemos establecer la diferencia entre los niveles de intensidad de ambas ondas a partir de la gráfica anteriormente mencionada y, a continuación, realizar el proceso inverso para así obtener el valor de la amplitud de la segunda onda.

$$B_I = 10log(I/I_0) \tag{1.8}$$

Teoría del dúplex

Para concluir la localización en el plano horizontal, comentar la teoría del dúplex la cual fue enunciada por Lord Rayleigh en 1907. Esta teoría nos explica que en altas frecuencias las pistas que nos proporcionan información sobre la localización de la fuente vienen dadas por diferencias de intensidad, mientras que en bajas frecuencias estas pistas provienen de diferencias en cuanto a las diferencias de tiempo. La teoría únicamente explica este fundamento en ondas senoidales.

1.2.4.2 Localización en plano medio

En el plano medio, suponiendo el ángulo azimut nulo, las ondas sonoras recorrerán el mismo espacio y emplearán el mismo tiempo para llegar a ambos oídos, por lo tanto el valor ITD e IID resulta nulo.

La localización en este plano es totalmente independiente de ITD e IID, como dijimos anteriormente, por lo que debemos emplear otro sistema para ello. De forma natural la localización de ondas en el plano medio, depende de factores acústicos como la frecuencia del sonido y de factores físicos como los hombros, la cabeza y el pabellón auditivo, siendo este último el más importante ya que en función de como rebotan las ondas nos indicará la posición de dicha fuente.

Para lograr esta técnica se emplean cabezas artificiales a las que se les implementa unos pequeños micrófonos en las orejas. Después se ejecutan sonidos que son captados mediante estos micrófonos. Las diferencias espectrales entre el sonido original y el sonido captado es lo que se conoce como HRTF.

La localización en el plano medio es muy individual, enormemente influenciada por las características físicas de la persona, debido a ello decidimos no realizar la localización en este plano.

1.2.4.3 Profundidad de la fuente

Estudio de las técnicas de profundidad de sonidos, para la diferenciación de la distancia a la que se encuentra el objeto detectado. La interpretación de la profundidad de un sonido viene determinada principalmente por tres factores:

Motion Parallax

Referido a los movimientos de la cabeza. El hecho de mover la cabeza hace que los parámetros de azimut varíen, y estos son dependientes de la lejanía de la fuente.

Reverberación

Influye en las reflexiones del sonido con las paredes de una habitación. La reverberación es mayor, cuanto menor sea la distancia de la fuente.

La intensidad del sonido

La intensidad de un sonido decae según el cuadrado de la distancia la fuente (Ecuación 1.7) ya que siempre asumimos estar en campo lejano, como comentamos al principio del punto 1.2.4. Por ello, la amplitud se vera disminuida en relación a la distancia de la fuente. Debido a que esta sensación de profundidad requiere conocer previamente un sonido base, para ver si se encuentra más cercano o alejado, decidimos crear cuatro conjuntos de sonidos bien diferenciados en cuanto a intensidad para indicar a que distancia se encuentra.

En el presente proyecto, para la identificación de la distancia a la que se encuentra la fuente se tendrá en cuenta el factor de la intensidad del sonido. La reverberación no debería tenerse en cuenta al suponer que siempre se usarán auriculares para la escucha de los sonidos, lo cual hace que no exista reflexión con los objetos de la sala. Por otro lado, los movimientos de la cabeza (motion parallax) son simplemente un sistema de ayuda en caso de no estar reconociendo la posición de la fuente.

1.2.5 Filtro de Kalman

Con numerosas aplicaciones en sistemas de control, seguimiento y navegación, el Filtro de Kalman, se trata de un algoritmo para la estimación de estados no observados directamente en sistemas dinámicos lineales contaminados con ruido gaussiano. El Filtro de Kalman nos permite obtener el estado en cada momento del tiempo con base a la información disponible en el momento $k-1$, y actualizar ésta con la información adicional disponible en el momento k.

El modelo del proceso que describe la transformación que sigue el estado, viene definido por la siguiente expresión: 1.9, la cual es dependiente del vector de estados en el instante anterior (X_k), la entrada del sistema (U_k) y una perturbación aleatoria del proceso (W_k).

$$X_k = AX_{k-1} + BU_k + W_k \tag{1.9}$$

- **Matriz de transición de estados. (A).** Es la matriz que relaciona el estado (K-1) con el estado siguiente (K). Puede ser variante con el tiempo, pero lo asumimos constante.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Matriz de transición de la entrada. (B)** Matriz que refleja la influencia de la entrada del sistema.
- **Ruido del proceso. (W).** Se trata de un ruido blanco con una distribución de probabilidad normal y de media cero $E(W_k) = 0$. Se asumirá constante. $E(W_k W_k^t) = \mathrm{Q}$

En nuestro modelo se asumirá un **modelo de velocidad constante**, ya que en principio los movimientos que se deben realizar no deben ser muy bruscos para un buen funcionamiento. La variable de entrada al sistema (u) será nula, quedando por tanto la ecuación de estado reducida a expresión 1.10.

$$X_k = AX_{k-1} + W_k \tag{1.10}$$

El modelo de medición describe la relación entre el proceso y las medidas, según la expresión 1.11. Dependiente además de la matriz de covarianza de la perturbación de la medida (R).

$$Y_k = HX_k + V_k \tag{1.11}$$

- **Matriz de medición. (H).** Establece la relación entre las mediciones y el vector de estado en el momento (k), en el supuesto ideal de que no hubiera ruido en las mediciones.

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- **Ruido de la medida. (V).** Al igual que el ruido del proceso (W), se trata de un ruido blanco con una distribución de probabilidad normal y de media cero $E(V_k) = 0$. Se asumirá constante. $E(V_k V_k^t) = \mathrm{R}$

El filtro es un procedimiento matemático que opera por medio de un **mecanismo de predicción y de corrección** (véase Figura 1.22). En esencia este algoritmo pronostica el nuevo estado a partir de su estimación previa añadiendo un término de corrección proporcional al error de predicción, de tal forma que este último es minimizado estadísticamente (véase Figura 1.23).

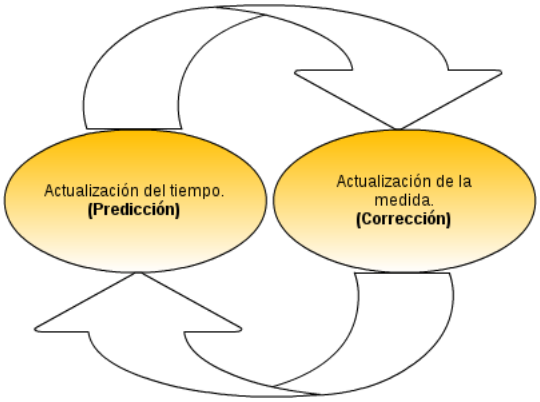


Figura 1.22: Etapas del Filtro de Kalman.

Estimación Se obtiene el valor del vector de estados estimado[2] (X_k^-) empleando para ello el modelo matemático del sistema. Además se obtiene la matriz de covarianza del error de estimación a priori (P_k^-), que incluye la matriz de covarianza del error del sistema (Q), incorporando así el error de modelado.

Corrección La primera tarea consiste en la obtención de la ganancia de Kalman (K_k) a partir de las matrices (P_k^-, H y R), éste parámetro es seleccionado de tal forma que se minimice la covarianza del

[2]El superíndice "-" indica que proviene de un proceso de estimación.

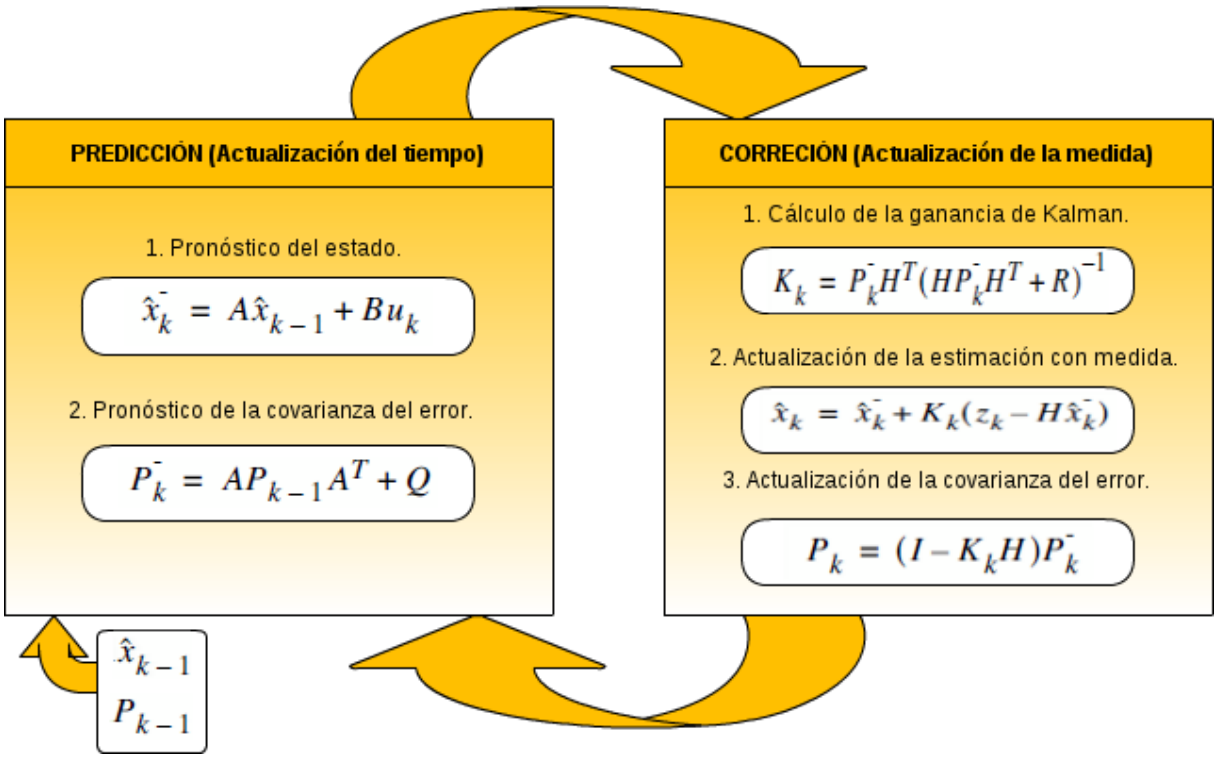


Figura 1.23: Fase de estimación y correción del Filtro de Kalman.

error de la nueva estimación del estado. Después debemos generar una nueva estimación del estado (X_k) incorporando la nueva medición externa (Z_k). Por último, debemos actualizar la matriz de covarianza del error de estimación (P_k) para la siguiente etapa de predicción. El proceso, se trata de una combinación de la información obtenida de la etapa de predicción con la información externa.

Para la inicialización del filtro, este requiere de dos variables: La ecuación de estados (X_{k-1}), que consiste en la posición del objeto obtenida por medio del detector y sobre el que se va a realizar el proceso del Filtro de Kalman y, además, se requiere de la matriz de covarianza del error de estimación (P_{k-1}), que debido a conocer el valor inicial de la ecuación de estados se asumirá nulo, ya que no presenta ningún error inicialmente.

1.3 Estudio experimental y solución desarrollada

En el siguiente apartado se detalla la solución desarrollada y se realiza un análisis acerca de las mejoras propuestas.

- En primer lugar, en la sección 1.3.1 se detalla la parte hardware que compone nuestro sistema.
- En la siguiente sección 1.3.2 se comenta la parte software del algoritmo desarrollado.
- Para finalizar, en la última sección 1.3.3 se detallan las pruebas realizas sobre el funcionamiento del algoritmo. Se divide en dos partes: la primera, en la que se detallan las pruebas enfocadas en la mejora de la detección de objetos y, la segunda, en la que se realiza el análisis de la localización de objetos mediante sonidos.

1.3.1 Hardware del sistema

El hardware del sistema (véase Figura 1.24) se encuentra constituido por los siguientes elementos.

Figura 1.24: Diagrama Hardware del sistema.

Sensor Kinect. Sistema de visión desarrollado por la compañía "Microsoft". Se encuentra constituido por una cámara RGB y un sensor de profundidad. Permite la captura del entorno en tiempo real, proporcionando al detector imágenes en color junto a una nube de puntos que contiene información de la profundidad de cada píxel de la imagen. Para más información sobre su funcionamiento, consultar el apartado 1.2.1.

CPU. Encargada de ejecutar el algoritmo desarrollado sobre el framework ROS. Se alimenta de los datos que proporciona el sensor Kinect. Como resultado genera una señal acústica para la localización binaural de los objetos detectados.

Dispositivo audio. Permite la escucha de la señal auditiva que es enviada por la aplicación para la poder realizar una localización del objeto en el espacio.

1.3.2 Software del sistema

El Software empleado en nuestro sistema (véase Figura 1.25) se encuentra desarrollado sobre la plataforma ROS. Como se indicó en la sección 1.2.3, una aplicación desarrollada sobre el framework de ROS se encuentra constituida a partir de varias aplicaciones pequeñas creadas sobre nodos, compartiendo entre ellos la información a partir de los llamados topics.

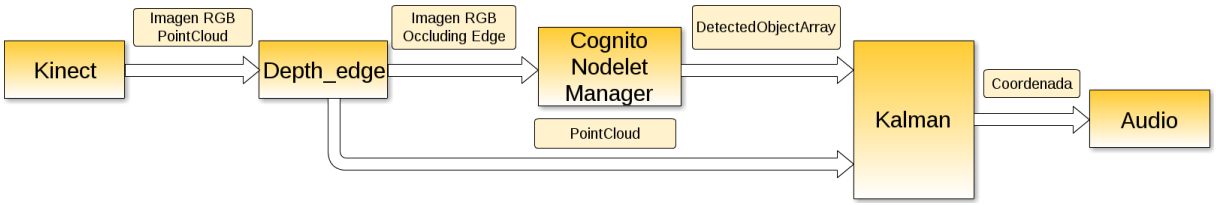
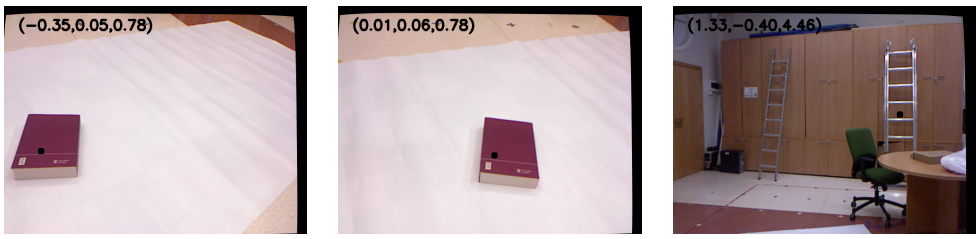


Figura 1.25: Diagrama Software del algoritmo desarrollado sobre el framework de ROS.

El software original del detector [2], se encuentra formado únicamente por dos nodos: uno encargado de la ejecución de su algoritmo y, otro nodo perteneciente a la cámara empleada encargado de alimentar el detector con imágenes color. Como ya se comentó anteriormente, este detector no resulta útil en determinadas situaciones en las que no se puede establecer un contraste entre el color de los objetos y la superficie sobre la que estaban situados. Por esta razón, una de las primeras modificaciones llevadas a cabo ha consistido en aprovechar el sensor de profundidad de Kinect, e incluir los bordes debidos a las discontinuidades de profundidad, que se pueden obtener a partir de la nube de puntos que Kinect proporciona, a los ya detectados por el algoritmo original.

Una vez detectados los objetos sobre la imagen. El detector se encarga de notificar el tipo y la posición del objeto sobre las coordenadas de la imagen. A partir de ello, con la finalidad de solucionar el problema de intermitencia a la hora de detectar objetos, se ha decido realizar un seguimiento del objeto a partir de un Filtro de Kalman. Este seguimiento se realiza sobre las coordenadas de la imagen (u, v) y, nos permite en determinadas situaciones en las que el detector falla, disponer de una estimación acerca de la posición que debería ocupar el objeto.

En función de la posición que proporciona el Filtro de Kalman sobre las coordenadas de la imagen, se ha logrado la obtención de su posición en el coordenadas cartesianas respecto la cámara RGB de Kinect (véase Figura 1.26). Para ello se hace uso de la nube de puntos proporcionada por el nodo de Kinect.



(a) Detección libro 1 (b) Detección libro 2 (c) Detección escalera 3

Figura 1.26: Coordenadas cartesianas de objetos detectados en el espacio. En la esquina superior izquierda de cada imagen se puede comprobar las coordenadas cartesianas del centroide del objeto detectado, que viene identificado en la imagen con un punto de color negro.

Una vez obtenidas las coordenadas cartesianas, el algoritmo se encargará de reproducir la señal auditiva en función del ángulo y la distancia euclídea a la que se encuentre el objeto correspondiente, para así poder identificar el posicionamiento de este mediante una técnica de sonidos binaurales. El dispositivo Kinect limita en cierto modo el rango admisible de detección de objetos, en cuanto a la orientación se permite un margen entre $\pm 30^{o}$ y un rango admisible de profundidad a partir de 40cm, ya que un valor inferior no es detectado por el dispositivo.

A continuación se detallan los nodos y topics que forman la aplicación. De esta manera se podrá comprobar como se ha desarrollado todo lo citado anteriormente.

1.3.2.1 Nodos

Se tratan de archivos ejecutables dentro de un paquete de ROS, que realizan funciones concretas.

“Kinect”

Este paquete contiene los archivos de lanzamiento para el uso de dispositivos compatibles, como el Kinect de Microsoft en ROS. Se crea un nodelet capacitado para la transformación de los datos desde el dispositivo a nubes de puntos, imágenes de disparidad y otros productos adecuados para el procesamiento y visualización.

“Depth Edge”

A partir de la nube de puntos recibida, se realiza la detección de los bordes de profundidad em-

pleando las librerías "PCL". El método empleado obtiene dichos borde en aquellas situaciones en las que exista una diferencia determinada de profundidad entre los puntos vecinos. Además de recibir la nube de puntos sobre la que obtienen los bordes, también se obtiene la imagen color correspondiente sobre la que no realiza ninguna modificación. Esto resulta necesario para la sincronización de las dos informaciones para el resto de procesos, ya que un desajuste provocaría un incorrecto funcionamiento de la aplicación.

"Cognito nodelet manager"

Se trata del detector "Real-time Learning and Detection of 3D texture-less objects: A Scalabe Approach" [2], para más información sobre su funcionamiento consultar el punto 1.2.2. Se ha realizado una mejora sobre el sistema al incluir los bordes detectados mediante la cámara de profundidad, a los ya detectados por el sistema original.

"Kalman"

Teniendo como entrada la nube de puntos y el topic"MultiObjectDetector" , que contiene la información sobre todos los objetos detectados en la imagen. Este nodo se encarga de realizar el seguimiento a partir de un Filtro de Kalman. Una vez realizado el seguimiento del objeto según se detalla en la sección 1.2.5, se hace llegar la información al usuario por medio de la reproducción de sonidos. Para ello, basta con indicar el ángulo y la distancia euclídea a la que se encuentra el objeto respecto a la cámara RGB de Kinect al siguiente nodo llamado "audio". Debido a que el seguimiento del objeto, por parte del Filtro de Kalman, se realiza sobre las coordenadas (u, v) del plano imagen y, por tanto nos proporcionará los puntos predichos sobre el plano imagen, estas debe ser combinadas junto con la nube de puntos para así obtener sus coordenadas cartesianas y así poder obtener la información deseada para su posterior localización binaural.

Para el funcionamiento del Filtro de Kalman se ha desarrollado el siguiente algoritmo:

Data: Inicio: Primera detección, se crea el filtro.
Objeto_detectado: Se ha detectado en el frame correspondiente un objeto.
Exceso: $|P| >$ Valor
begin
  if *Filtro_Activo* **then**
    if *Inicio* **then**
      Iniciar_matrices ();
    if *Objeto_detectado* **then**
      Corrección ();
    Estimación ();
    if *Exceso* **then**
      Filtro_Activo=false;
end

Algorithm 1.1: Algoritmo explicativo sobre el funcionamiento del Filtro de Kalman.

La primera función "Iniciar_matrices ()" se ejecutará siempre que se inicia por primera vez un Filtro de Kalman, en ella se procede a la definición de las matrices que dan inicio a este filtro: X_0 y P_0. La primera, tendrá asignado el valor de las coordenadas en las que ha sido detectado el objeto sobre el plano imagen y sus velocidades, que serán nulas hasta que se establezcan las siguientes detecciones y,

por tanto, se actualicen sus valores de forma automática. Por otro lado, P_0 tendrá un valor nulo ya que al conocer el valor de X_0 no representa incertidumbre en el valor de la estimación inicial.

$$\begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} u \\ v \\ 0 \\ 0 \end{pmatrix}$$

$$P_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

En caso de que el detector nos indique que se ha realizado una detección sobre el objeto del que se está ejecutando el filtro, es decir, se actualicen sus coordenadas en el plano imagen, se procede a realizar un proceso de corrección de las matrices mediante la llamada a la función: "Corrección ()". Después de su ejecución se lleva a cabo un proceso de estimación de las medidas que se ejecuta por cada frame, exista o no proceso de corrección.

Según se ejecuten procesos de estimación el valor de la matriz de covarianza del error de estimación, es decir, P irá aumentando su valor, pues el error generado resulta cada vez mayor. En el instante en que se cumpla la siguiente expresión.

$$P(0,0) * P(1,1) - P(0,1) * P(1,0) > 20$$

Se indica que la estimación ha generado demasiado error como para conocer la verdadera posición que ocupa el objeto y, el Filtro de Kalman dejará de ejecutarse hasta la existencia de una nueva detección que comience de cero. El valor ha superar para la cancelación de filtro ha sido seleccionado por el método prueba-error e indica que se ha la posición estimada ha superado un radio ficticio preestablecido respecto a la última posición de detección del objeto.

"Audio"

A partir de información recibida tras el proceso Filtro de Kalman, el nodo "Audio" se encargará de reproducir el sonido correspondiente que se encuentra almacenado en la base del programa, para así poder realizar una localización en el espacio del objeto mediante la técnica binaural. Cada vez que se detecte un nuevo objeto se notificará mediante un "speech" el número identificador del objeto en cuestión.

1.3.2.2 Topics

Medio de comunicación entre los nodos de un sistema.

Imagen RGB

Contiene la imagen color proporcionada por el sensor Kinect (véase Figura 1.27 a).

PointCloud

Contiene la información nube de puntos sobre el entorno proporcionada por el sensor Kinect. (véase Figura 1.27 b).

Occluding Edge

Se trata de la nube de puntos que contiene toda la información sobre los bordes extraídos de la nube de puntos original generada por Kinect (véase Figura 1.27 c).

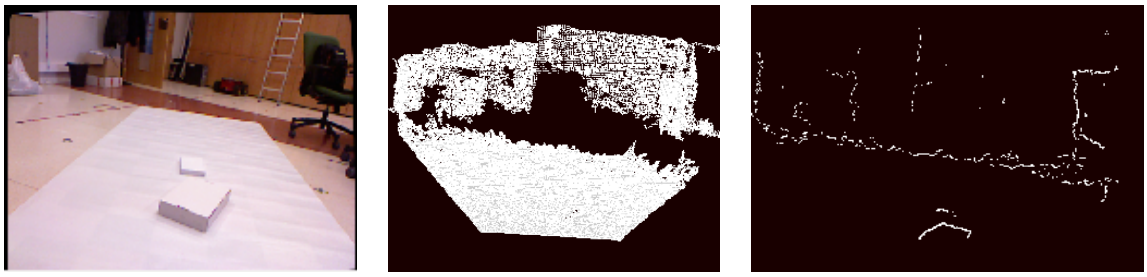


Imagen RGB (a) PointCloud (b) Occluding Edge (c)

Figura 1.27: Ejemplo del contenido de diversos "topics".

DetectedObjectArray

Indica las características sobre los objetos detectados en la imagen (véase Figura 1.28). Se puede encontrar la siguiente información:

- Número de objeto detectado.
- Área de interés del objeto.
- Coordenada de los puntos que conforman los bordes del objeto detectado.

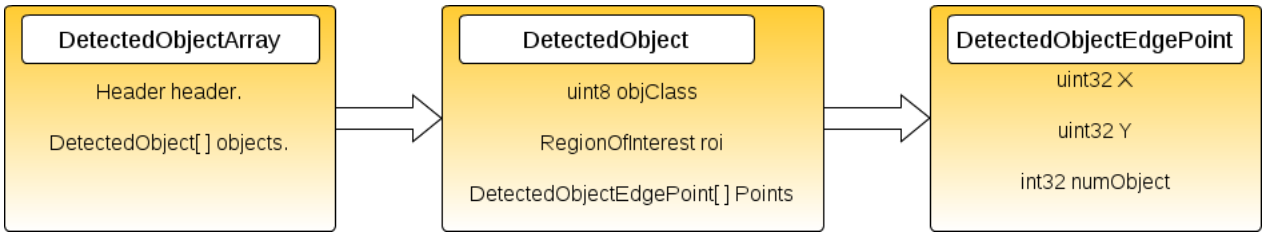


Figura 1.28: Información del contenido del mensaje "DetectedObject".

Coordenada

Contiene información sobre la posición que ocupa el objeto: el ángulo, con un margen admisible entre $\pm 30^{o}$, y la distancia euclídea, que debe ser superior a 40cm, ambas medidas vienen referenciadas respecto a la cámara color del dispositivo Kinect. Además, proporciona el número identificador del objeto que es detectado.

1.3.3 Resultados experimentales

Se procede al análisis de los resultados obtenidos a partir de las diferentes pruebas realizadas con el algoritmo que se ha desarrollado. En primer lugar se detallan los resultados de las pruebas enfocadas a la detección de objetos y, posteriormente los resultados obtenidos en el testeo de los sonidos binaurales con personas.

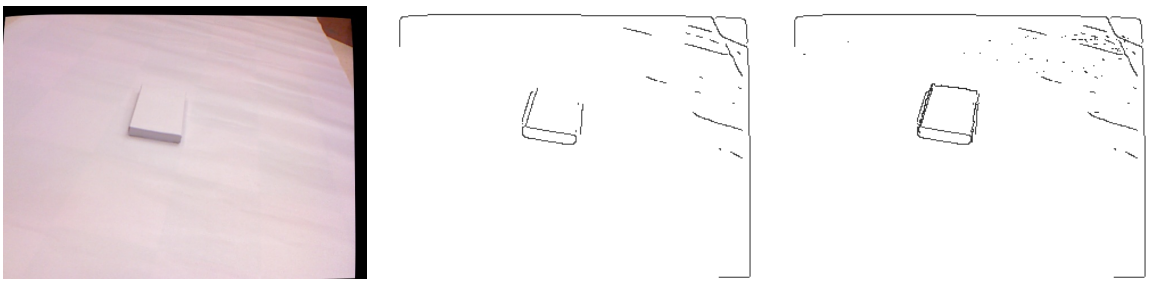
1.3.3.1 Detección de objetos

Este apartado se encuentra divido en tres secciones. En la primera se detallan las pruebas enfocadas en la detección de bordes, en la segunda se comentan las pruebas enfocadas en la detección de los objetos en diferentes entornos y, por último, se detallan las pruebas realizadas para comprobar el funcionamiento del seguimiento de los objetos.

Detección de bordes.
La primera modificación que se ha llevado a cabo consiste en la mejora de los bordes detectados. De forma original, como se indicó en el punto 1.2.2, el detector sólo utilizaba los bordes obtenidos a partir de imágenes color. Debido a la utilización del sensor Kinect en el proyecto, este método no sacaba partido a todo su potencial pues desaprovechaba su sistema de percepción de la profundidad tanto para la fase de entrenamiento como para la fase detección. Por ello se decidió obtener los bordes a partir de la nube de puntos obtenida con la cámara de profundidad y, añadirlos a los obtenidos mediante la imagen color por el detector tanto para la fase de entrenamiento como para la de detección.

Además, se pretende solucionar así uno de los problemas detectados en el algoritmo original: completar la detección de bordes en objetos en los que el detector no es capaz debido, principalmente, a no poder establecer un contraste de color.

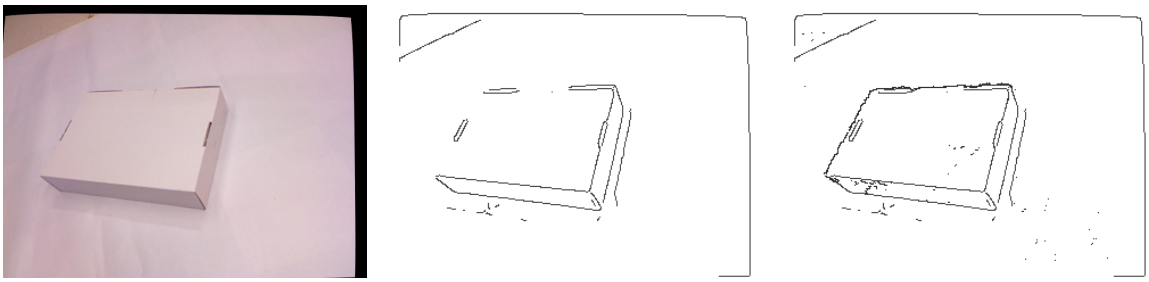
Para testear el funcionamiento de esta primera modificación, hemos probado como es la detección sobre dos objetos: un libro (véase Figura 1.29) y una caja (véase Figura 1.30), ambos de un color similar al de la superficie sobre la que se encuentran. Como se puede observar en los bordes detectados mediante



RGB | Bordes RGB | Bordes RGB + Profundidad

Figura 1.29: Ejemplos de detecciones sobre un libro.

color, no se puede establecer un contraste entre los objetos y el fondo. Por ello, gran cantidad de bordes no son detectados provocando que el contorno de estos no sea completamente identificado. El hecho de que tanto la fase de entrenamiento como la fase de detección no tengan una correcta información sobre el objeto, provocará que este no pueda ser reconocido en una gran cantidad de ocasiones. Si, por otro lado, añadimos nuestra mejora en el algoritmo podemos comprobar como todo el contorno del objeto es detectado y, así, se puede realizar tanto un buen entrenamiento como una buena detección.



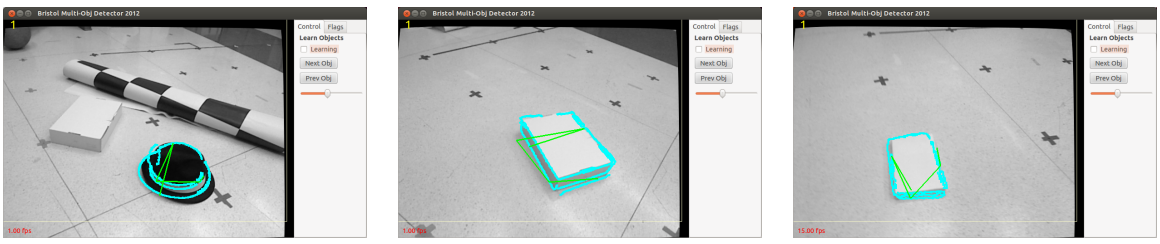
RGB Bordes RGB Bordes RGB + Profundidad

Figura 1.30: Ejemplos de detecciones sobre una caja.

Detección de objetos

Con el fin de evaluar la inclusión de bordes de profundidad para la detección de objetos, hemos sometido el programa a una prueba. Se ha realizado el siguiente experimento. Se han seleccionado tres objetos a testear: un sombrero, una caja y un libro, los dos últimos (cómo en el caso anterior) con un color similar a la superficie sobre la que se ha realizado el entrenamiento. Se ha grabado para uno de los objetos dos secuencias de vídeo: una en la que se realizará un entrenamiento del objeto sobre una superficie lisa, evitando que otros bordes pueda interferir en el entrenamiento y, un segundo vídeo sobre una superficie diferente en la que interfieren mayor cantidad de bordes no pertenecientes al objeto. A partir de estos vídeos, se procede a contrastar el funcionamiento del algoritmo incluyendo bordes de profundidad y sin ellos, para poder comprobar si se mejora la detección como se preveía en un primer momento.

Como se puede observar en los tres casos testeados (véanse las Figuras 1.31) se produce una mejora en las detecciones incorporando los bordes de profundidad (véase Tabla 1.1). Principalmente este efecto es percibido con el segundo objeto (la caja) donde pasamos de no obtener ninguna detección a una media más aceptable de más de diez. Aun así, en todas las secuencias los resultados siguen sin ser tan satisfactorios como se deseaba, ya que la tasa de detección es en general muy baja, siendo el máximo porcentaje de un 6.43 %. Esta situación ya había sido conocida desde el análisis del algoritmo original, por tanto el detector no genera datos muy satisfactorios para ambientes en los que interfieran gran cantidad de bordes.



Sombrero Caja Libro

Figura 1.31: Detecciones sobre entornos con mayor tasa de bordes no pertenecientes al objeto a identificar.

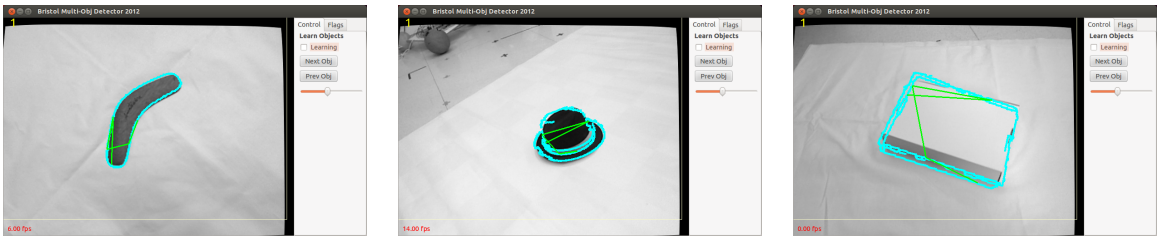
Objeto testeado	Sistema detección	Secuencia (Detecciones)				Media de frames totales	Tasa de detección (%)
		Primera	Segunda	Tercera	Media		
Sombrero	RGB	1	0	1	0.67	150	0.446
	RGB + Depth	**4**	**2**	**3**	**3**	**150**	**2**
Caja	RGB	0	0	0	0	160	0
	RGB + Depth	**11**	**9**	**11**	**10.33**	**160**	**6.43**
Libro	RGB	0	0	0	0	108	0
	RGB + Depth	**3**	3	**3**	**3**	**108**	**2.78**

Tabla 1.1: Prueba de mejora de bordes sobre fondos con mayor tasa de bordes no pertenecientes al objeto a identificar.

La segunda prueba se ha realizado de una manera similar a la anterior. Pero con la diferencia de que en este caso la fase de testeo será también sobre una superficie en la que apenas existan bordes que puedan interferir (véase Tabla 1.2), evitando así uno de los inconvenientes del algoritmo original. En este caso los objetos a testear serán: dos objetos con un elevado contraste con el fondo de entrenamiento y de detección (**un búmeran y un sombrero**) y un objeto que no permita un buen contraste de colores (**un libro**) (véanse las Figuras: 1.32).

Objeto testeado	Sistema detección	Secuencia (Detecciones)				Media de frames totales	Tasa de acierto (%)
		Primera	**Segunda**	**Tercera**	**Media**		
Búmeran	RGB	21	20	15	18.66	30	62
	RGB + Depth	**16**	**22**	**19**	**19**	**30**	**63.3**
Sombrero	**RGB**	**28**	**27**	**24**	**26.3**	**80**	**32.87**
	RGB + Depth	21	20	25	22	80	27.5
Caja	RGB	0	0	0	0	55	0
	RGB + Depth	**11**	10	**11**	**10.67**	**55**	**19.4**

Tabla 1.2: Prueba de mejora de bordes sobre entornos limpios.



Búmeran Sombrero Caja

Figura 1.32: Detecciones sobre entornos limpios.

En esta prueba realizada las conclusiones son mejores y, entran dentro de lo esperado en un primer momento. En todos los casos las tasa de detecciones son mucho mayores que en la prueba anterior, llegando incluso al 63.3 % en el mejor caso. Se procede al análisis individual de cada uno:

- El objeto **búmeran** se trata de un objeto plano. Por consiguiente, que la media en ambas pruebas sea muy similar es lo esperado. Esto es resultado de no poder generar bordes a partir de la cámara de profundidad, debido a que el sistema no es capaz de establecer las diferencias de profundidad para la generación de estos.

- El segundo objeto testeado, **el sombrero**, también nos aporta valores muy similares. En este caso sí se obtienen bordes a partir del sensor de profundidad, pero al ser de un tonalidad oscura y, realizar tan buen contraste con el fondo blanco, los bordes generados por las imágenes color son lo suficientemente buenos para detectar el contorno del objeto en su totalidad.

- En el último caso, **la caja**, es donde se puede apreciar la mejoría del sistema. A partir de imágenes en color el sistema es incapaz de obtener todos los bordes que delimitan el objeto. Con lo cual no se puede realizar un entrenamiento adecuado y, por consiguiente, en la detección, que también sigue siendo insuficiente con la cantidad de bordes obtenidos, no es capaz de detectar al objeto en ninguna ocasión. Una vez añadidos los bordes generados por el sensor de profundidad, sí se obtienen todos los bordes que delimitan al objeto en ambas fases y, el objeto es detectado en una gran cantidad de ocasiones, teniendo una tasa de detección del 19,4 %.

Como **conclusión** final, **se ha logrado mejorar** de una manera muy satisfactoria la: **ineficaz identificación de contornos de objetos en diferentes situaciones**, como por ejemplo en las que no existe un contraste de colores entre el objeto y el entorno que lo rodea. En ambos casos, tanto en los entornos con interferencia de otros bordes como en entornos aislados se ha logrado un incremento en las tasas de detecciones, aunque en los primeros el detector no proporciona una gran eficacia. Esto lleva a la conclusión de preferiblemente utilizar el algoritmo en situaciones en las que no interfiera una gran cantidad de bordes.

Filtrado de Kalman

En la aplicación original la detección de objetos resulta de forma intermitente, es decir, existen entre varias detecciones de un mismo objeto varios frames sin detección alguna, como se indicó en la sección 1.2.2. Este hecho ha provocado la realización de una nueva incorporación en el algoritmo que consiste en realizar un seguimiento del objeto detectado mediante un Filtro de Kalman.

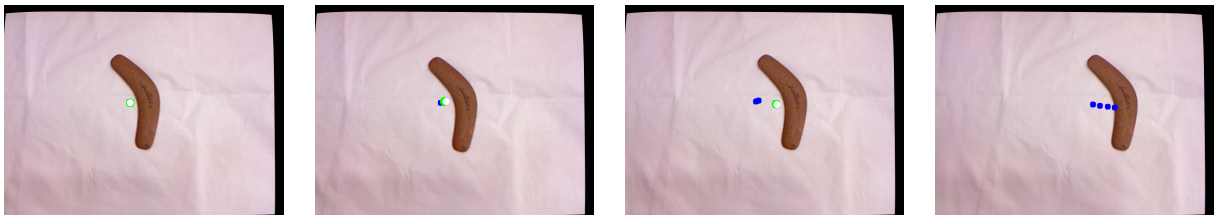
La prueba realizada para observar el proceso ha sido la siguiente. Se realiza el entrenamiento de dos objetos: un libro y un búmeran. A continuación, se realiza una secuencia de vídeo para cada objeto en la que son grabados mientras desplazamos nuestro dispositivo Kinect hacia la izquierda. Con ello se pretende observar si realmente el sistema, con la mejora incluida, es capaz de proporcionar la posición del objeto mientras éste no es detectado. En los ejemplos se debe tener en cuenta lo siguiente.

Punto Blanco. Indica la posición en la que ha sido detectado el objeto.

Punto Verde. Indica la posición proporcionada por el Filtro de Kalman tras aplicar un proceso de corrección.

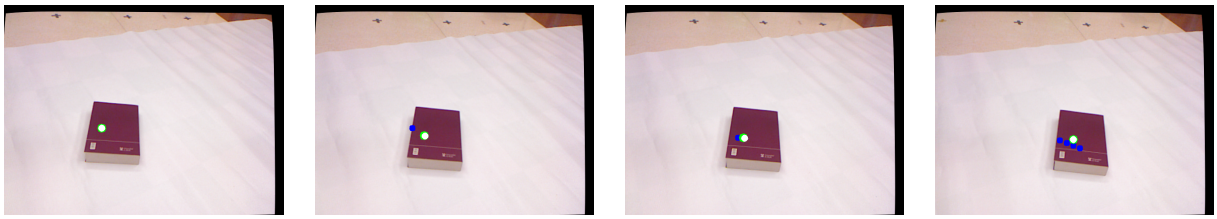
Punto Azul. Indica la posición proporcionada por el Filtro de Kalman tras aplicar únicamente el proceso de estimación. Pueden existir varios puntos sobre la imagen, puesto que se realizan varios procesos de estimación hasta que exista una nueva detección o el Filtro de Kalman queda eliminado.

En ambas secuencias (véanse las Figuras 1.33 y 1.34) se puede observar que el Filtro de Kalman requiere de varios frames de detección para poder realizar una buena estimación del objeto. Es por lo que en las primeras imágenes de ambas secuencia, que representan las primeras detecciones que se dan, los puntos estimados se encuentran prácticamente en el mismo lugar del punto en el que se detectó el objeto. Sin embargo, una vez el objeto ya no es vuelto a detectar, como podemos comprobar en la última imagen de ambos casos, si se aprecia como la predicción del objeto se realiza en la dirección correcta. Procedemos al análisis individual de cada secuencia:

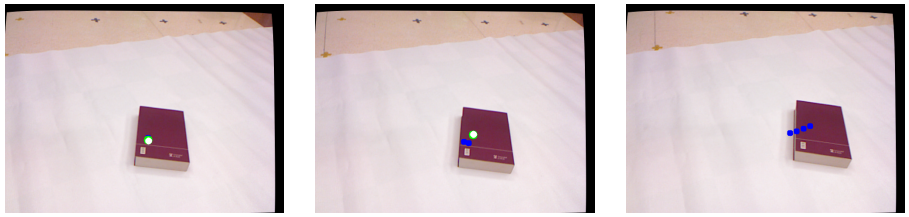


Detección 1 Detección 2 Detección 3 No realiza detección.

Figura 1.33: Ejemplo del Filtro de Kalman sobre un búmeran.



Detección 1 Detección 2 Detección 3 Detección 4.



Detección 5 Detección 6 No realiza detección.

Figura 1.34: Ejemplo del Filtro de Kalman sobre un libro.

- **En la secuencia del búmeran.** Llegamos a tener un total de tres detecciones hasta que el Filtro de Kalman pierde la pista al objeto. Con lo cual de haber podido transmitir un total de tres posiciones del objeto (puntos blancos), si no se hubiera aplicado el Filtro de Kalman. Se ha conseguido incrementar el número de posiciones a transmitir a un total de diez, que corresponde a la suma de los puntos estimados y corregidos generados por el filtro.
- **En la secuencia del libro.** Han surgido un total de seis detecciones hasta que el filtro ha perdido la pista al objeto. Por lo tanto, de un total de seis posiciones de las que transmitir información, se ha logrado incrementar el número de posiciones a un total de diecinueve aplicando los procesos de estimación y corrección del Filtro de Kalman.

Como **conclusión**, suponiendo que el portador del sistema no va a realizar movimientos bruscos y, además, que el objeto a detectar se trata de un objeto inmóvil. Hemos conseguido obtener una manera que permite conseguir más información sobre la posición del objeto que se quiere transmitir a la persona. Con lo cual **la búsqueda del objeto es más eficiente**.

1.3.3.2 Localización binaural

Con la finalidad de comprobar si la técnica empleada en el proyecto para la localización de objetos mediante señales auditivas es efectiva, se ha procedido a realizar un experimento. En este experimento que se detalla a continuación, gracias a la colaboración de la "Organización Nacional de Ciegos Españoles" (ONCE) [36] se ha podido contar con la participación de una persona invidente desde nacimiento (F.

Marfil), una persona invidente no de nacimiento (J.A. Saiz) y, además también se ha podido contar con la colaboración de varias personas sin ningún problema de discapacidad visual.

Para la realización del experimento se han generado cuatro archivos de audio. Tres correspondientes con cada una de las pruebas que se van a realizar y el restante para realizar una primera toma de contacto con la técnica. Cada pista de audio contiene una cantidad determinada de tonos y, cada tono ha sido creado mediante la técnica de localización binaural empleada en el proyecto a través de establecer diferencias de tiempo entre sonidos (ITD) que ha sido detallada en la sección 1.2.4.

Para la realización de las pruebas se ha establecido el siguiente sistema de referencia 1.16. En el que el origen del sistema se situará justo enfrente de la persona que esta realizando la prueba (0^o) y, podrá admitir un rango de -90^o si el sonido se sitúa justo a la izquierda de la persona hasta $+90^o$ si, en este caso, el sonido se sitúa a la derecha.

En primer lugar las personas han podido escuchar el audio de testeo de la técnica. En este audio se puede escuchar tres tonos distintos, cada uno proveniente de una dirección determinada. El primero en la izquierda del todo, es decir -90^o, el segundo situado justo en frente (0^o) y, el tercero en la derecha del todo el rango admisible ($+90^o$). La finalidad de este audio, que ha sido repetido en varias ocasiones para que entiendan bien el funcionamiento, es que el resto de pruebas se realicen sin ninguna repetición.

En la primera prueba realizada (véase Tabla 1.3), se podrá escuchar un total de tres tonos distintos. El primero simulará la posición de la fuente en -45^o, el siguiente situará la fuente justo enfrente del usuario (0^o) y, el último situará la fuente en $+45^o$. Lo que se pretende es que la persona que escuche la grabación sea capaz de decirnos en que ángulo cree que se encuentra la fuente en cada uno de los tonos que escucha (véanse Figuras 1.35).

Nombre	**-45º**	**0º**	**45º**
Jorge P.	-45º	0º	90º
Gonzalo D.	-90º	0º	45º
David V.	-40º	10º	80º
Álvaro G.	-60º	30º	60º
Álvaro M.	-90º	15º	50º
Cristian V.	-90º	60º	90º
F. Marfil	**-35º**	**0º**	**35º**
J.A. Saiz	**-45º**	**0º**	**45º**

Tabla 1.3: Resultados de la primera prueba - Localización mediante sonidos.

La segunda prueba resulta muy similar a la anterior, pero en este caso se realizará con un total de cinco sonidos (véanse Figuras 1.36). Estando cada uno de ellos situados en $-60^o, -30^o, 0^o, 30^o$ y 60^o, respectivamente. Los datos han sido los siguientes (véase Tabla 1.4).

Las dos primeras pruebas resultan muy similares para las conclusiones a obtener de ellas. Como se puede comprobar en todos los casos, esta técnica sí permite una buena percepción de la variación de la fuente. Siendo en unas personas los resultados más exactos que en otras, en todos los participantes se ha podido lograr una **buena diferenciación entre los sonidos** que provenían de la parte izquierda con los provenientes de la parte derecha, e incluso diferenciar perfectamente una variación entre los sonidos provenientes del mismo lado.

Fuente situada en -45º

Fuente situada en 0º

Figura 1.35: Primera prueba de localización binaural.

Fuente situada en 30º

Fuente situada en 0º

Figura 1.36: Segunda prueba localización binaural.

Nombre	**-60º**	**-30º**	**0º**	**30º**	**60º**
Jorge P.	-90º	-45º	0º	45º	90º
Gonzalo D.	-90º	-60º	0º	10º	60º
David V.	-60º	-45º	0º	30º	50º
Álvaro G.	-90º	-45º	0º	45º	90º
Álvaro M.	-70º	-50º	15º	20º	50º
Cristian V.	-90º	-60º	0º	30º	60º
F. Marfil	**-60º**	**-30º**	**0º**	**30º**	**60º**
J.A. Saiz	**-90º**	**-30º**	**0º**	**30º**	**60º**

Tabla 1.4: Resultados de la segunda prueba - Localización mediante sonidos.

Para finalizar, la última prueba consiste en la simulación del movimiento de la fuente. La fuente ficticia comenzará situada en la izquierda del todo, es decir, a -90^o. A partir de ello, realizará un barrido hacia la derecha con saltos de 5^o, hasta acabar situándose en la derecha a $+90^o$. Lo que se pretende es que cuando la persona que esta realizando la prueba perciba que el sonido se encuentra situado a $-45^o, 0^o$ y 45^o lo notifique para poder comprobar si realmente ha sido correcta su apreciación. Los resultados se detallan a continuación (véase Tabla 1.5).

Nombre	**-45º**	**0º**	**+45º**
Jorge P.	+3	-3	-12
Gonzalo D.	-7	-13	-18
David V.	-3	-5	-3
Álvaro G.	+1	-2	-5
Álvaro M.	-2	-2	-1
Cristian V.	0	3	-5
F. Marfil	**0**	**-1**	**-5**
J.A. Saiz	**-5**	**-6**	**-11**

Tabla 1.5: Resultados de la tercera prueba - Localización mediante sonidos. Sobre cada persona vienen indicados los segundos de desviación que han interpretado respecto a los teóricos. La prueba total tiene una duración de 60 segundos, y deben pasar 15 segundos, teóricamente, entre cada una de las posiciones a indicar.

Esta prueba ha sido la más precisa de las realizadas y que menos variación, en general, entre personas ha sufrido. Se puede sacar como conclusión que todas las personas **han sido capaces de identificar las pequeñas variaciones** en la posición de la fuente. Por tanto, esto resulta muy eficaz ya que para la detección de un objeto en tiempo real, la posición que ocupe no debe cambiar mucho entre cada sonido nuevo que nos llega.

Como **conclusión general** de las tres pruebas realizadas, los resultados son interpretados de maneras un tanto diferentes para cada una de las personas. En general todos los sujetos han sido capaces de percibir una buena aproximación de los valores reales, pero existe una gran cantidad de variaciones de uno respecto a otro. Esto ya se asumía, pues la técnica empleada **depende de gran cantidad de factores individuales** como por ejemplo el diámetro de la cabeza, del cual ya se indicó que se ha utilizado un promedio, y hace que esta técnica pueda ser interpretada de una manera ligeramente diferente en cada persona. Pero en general siempre han sido capaces de percatarse en que **zona esta aproximadamente localizada la fuente**. Es importante decir que con esta técnica resulta muy difícil dar una aproximación muy exacta del ángulo del que proviene la fuente, pero si son capaces de identificar la localización dentro de un cierto rango.

Los resultados para personas con discapacidad visual resultan mucho más exactos respecto a las personas que no la padecen. Por lo tanto, al ser diseñado para este tipo de personas la eficacia de esta técnica nos aporta conclusiones aún mejores. Esto es provocado debido al gran desarrollo de la audición que las personas con discapacidad visual han tenido que desarrollar para mejorar su día a día.

Capítulo 2

Pliego de condiciones

Para la correcta utilización de la aplicación desarrollada en el presente proyecto se deben cumplir una serie de requisitos.

2.1 Requisitos de Hardware

- Microsoft Kinect v1.
- Adaptador de corriente para Kinect.
- Acer Aspire v3-571G. (Dispositivo utilizado en el proyecto).
  - Procesador Intel Core i7.
  - Memoria RAM: 8GB.
  - Disco duro 750GB.
  - NVIDIA GeForce GT 640M.
- 1.5GB de memoria disponible en el disco duro para la aplicación y la carpeta contenedora de los archivos de audio.
- Sistema de audición auricular.

2.2 Requisitos Software

- Sistema operativo Ubuntu 12.04.5 LTS.
- ROS - Versión Fuerte.
- Openni drivers - ROS fuerte.
- Librería OpenCV version 3.1.
- PCL-1.7 trunk for ROS-Fuerte.

Capítulo 3

Presupuesto

3.1 Costes de ejecución material

En esta sección se desglosan los costes estimados del proyecto.

3.1.1 Costes de hardware

Concepto	Cantidad	Coste por unidad	Subtotal
Kinect	1	70 €	70 €
Adaptador Kinect	1	10 €	10 €
Acer Aspire v3-571G	1	550 €	550 €
Sistema reproducción auricular	1	20 €	20 €
Subtotal			**640 €**

Tabla 3.1: Coste del Hardware.

3.1.2 Costes de software

Concepto	Cantidad	Coste por unidad	Subtotal
Sistema operativo Ubuntu 12.04 LTS	1	0 €	0 €
ROS	1	0 €	0 €
Librería PCL	1	0 €	0 €
Librería OpenCV	1	0 €	0 €
Matlab r2014	1	2000 €	2000 €
Editor de texto LaTeXTextmaker 4.4.1	1	0 €	0 €
Subtotal			**2000 €**

Tabla 3.2: Coste del Software

3.1.3 Costes mano de obra

Concepto	Horas	Coste por hora	Subtotal
Ingeniería	380	35 €/hora	13300 €
Mecanografía	130	15 €/hora	1950 €
Subtotal			**15250 €**

Tabla 3.3: Costes de la mano de obra.

3.1.4 Coste total

El coste total incluye todos los gastos de software, hardware y mano de obra del proyecto.

Concepto	Subtotal
Coste hardware	640 €
Coste software	2000 €
Coste mano de obra	15250 €
Subtotal	**17890 €**

Tabla 3.4: Coste total

El precio final asciende a un total de:
Diecisiete mil ochocientos noventa Euros.

Capítulo 4

Manual rápido de usuario

En esta sección se detallan todos los aspectos necesarios para un correcto uso y conocimiento de la aplicación desarrollada.

4.1 Instalación de prerequisitos

4.1.1 OpenCv

- Descargar OpenCv de la página mediante Git.

```
$ git clone https://github.com/Itseez/opencv.git
```

- Instalación OpenCV.

```
$ cd ~/opencv
$ mkdir release
$ cd release
$ cmake..
$ cd ..
$ make
$ sudo make install
```

4.1.2 PCL

- Instalar las siguientes dependencias:
  - Boost 1.47.
  - Eigen 3.0.
  - FLANN 1.7.1.
  - VTK 5.6.
- Trunk version.

```
$ git clone https://github.com/PointCloudLibrary/pcl pcl-trunk
$ cd pcl-trunk && mkdir build && cd build
$ cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo ..
$ make -j2
$ sudo make -j2 install
```

- Versión inestable 1.7

```
$ sudo apt-get install libpcl-1.7-all-dev
```

4.1.3 ROS-Fuerte

Para la versión Ubuntu 12.04 (Precise), se debe seguir los siguientes pasos.

- Aceptar software de packages.ros.org

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main"
> /etc/apt/sources.list.d/ros-latest.list'
```

- Configurar las keys.

```
$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

- Actualizar

```
$ sudo apt-get update
```

- Instalación completa de ROS.

```
$ sudo apt-get install ros-fuerte-desktop-full
```

- Instalar dependencias necesarias.

```
$ sudo apt-get install python-rosinstall python-rosdep python-rospkg
\subsection{Openni Drivers}
```

4.1.4 Drivers Kinect

- Openni drivers.

```
$ sudo apt-get install ros-fuerte-openni-camera ros-fuerte-openni-launch
```

4.2 Instalación de la aplicación

- Instalar el algoritmo original. Ir a la carpeta contenedora del detector original [2] y ejecutar el archivo install. (En caso de duda viene incorporado un fichero .txt de como realizar esta instalación).

```
$ ./Install_Script.sh <loc>
```

- Posteriormente sustituir los archivos de nuestro algoritmo sobre el lugar donde se realizó la instalación.

4.3 Ejecución de la aplicación

- En el directorio donde se ha instalado la aplicación ejecutar la siguiente acción.

```
$ source ./~/setup.bash
```

- Compilar el programa al menos una vez, ejecutando la siguiente orden en el terminal.

```
$ rosmake mod
```

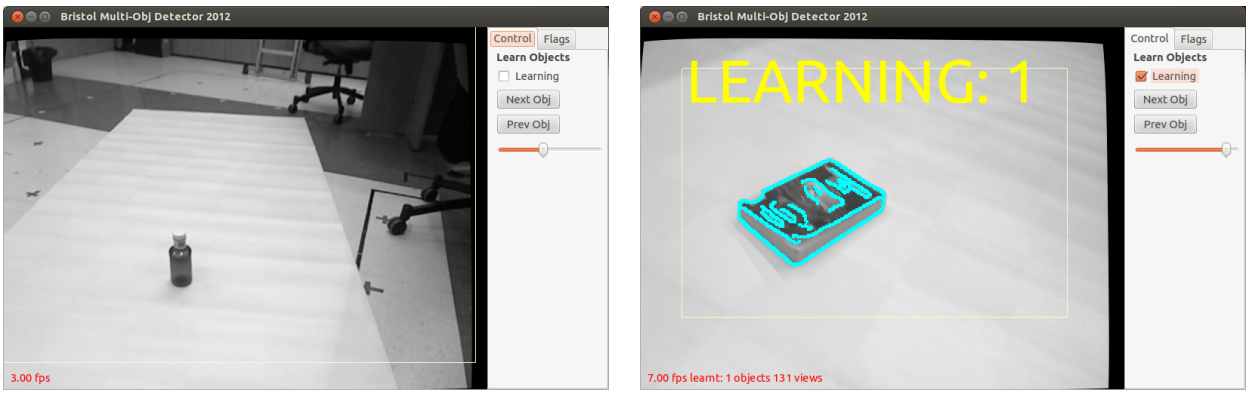
- Por último para lanzar la aplicación, ejecutar la siguiente orden.

```
$ roslaunch mod_base demo_rgbd.launch
```

4.4 Funcionamiento

4.4.1 Comandos de la aplicación.

En el lanzamiento de la aplicación se nos mostrará una pantalla (véase Figura 4.1 a). Si una base de datos se ha cargado, el programa funcionará por si solo, reproduciendo sonidos en función de la posición en el espacio que ocupe el objeto en cuestión. Aun así, a continuación procedemos a explicar todas las opciones de las que dispone la aplicación.



Pantalla inicial (a) Pantalla modo aprendizaje (b)

Figura 4.1: Pantallas en la aplicación

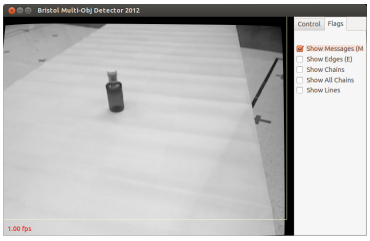
Pestaña Control. En la pestaña control podemos encontrar las siguientes opciones referidas principalmente a la etapa de entrenamiento (véase Figura 4.1 a).

- **Learning.** Permite comenzar la etapa de entrenamiento de objetos. Al activar la opción se mostrará un cartel indicando que se encuentra en la etapa "Learning" seguido del número de objeto sobre el que está almacenando la información. Además, se mostrará un rectángulo sobre la imagen que representará la zona sobre la que está realizando la detección para el entrenamiento (véase Figura 4.1 b). En la esquina inferior izquierda podemos ver el número de objetos que tenemos almacenados en la base junto a las vistas almacenadas del objeto sobre el que se está realizando la detección.
- **Next Obj.** Pasar al siguiente objeto para la fase de entrenamiento. (Permite un máximo de 20 objetos almacenados)
- **Prev Obj.** Pasar al anterior objeto para la fase de entrenamiento.
- **Barra deslizante.** Permite ajustar el tiempo máximo que se empleará en cada imagen hasta que ocurra alguna detección.

Pestaña Flags. En la pestaña "Flags" podemos encontrar las siguientes opciones que nos permitirán observar diferentes características de la aplicación (véase Figura 4.2 a)

- **Show Message.** Activar o desactivar todos los mensajes que aparecen sobre la pantalla (FPS, mensajes de la etapa de aprendizaje e indicador de objeto detectado).
- **Show Edges.** Nos permite observar otra vista que nos muestra los bordes que es capaz de detectar en la imagen (véase Figura 4.2 b).
- **Show Chains.** En detecciones nos muestra la constelación coincidente (véase Figura 4.2 c).
- **Show All Chains.** En detecciones nos muestra todos las constelaciones.

Fps. Nos muestra la velocidad de frames por segundo a la que se esta ejecutando el programa.


Pantalla secundaria


Pantalla de bordes

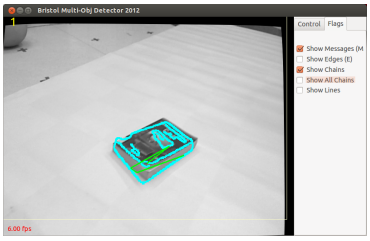

Pantalla show chain

Figura 4.2: Diferentes modos admitidos en la aplicación.

4.4.2 Guardar y cargar archivos contenedores de los diferentes entrenamientos.

Para crear nuevos ficheros contenedores de los entrenamientos realizados de objetos detectados debemos abrir el launcher. Este se encuentra en la siguiente ubicación *.../ObjectRecog/mod/mod_base/launch/demo_rgbd.launch*. Una vez abierto, debemos modificar los siguientes parámetros.

```
<node pkg="nodelet" type="nodelet" name="MultiObjDetector" output="screen"
 args="$(arg method) mod_base/MultiObjDetectorNodelet $(arg manager)">
   <param name="rgb_image" value="/image_rect_color" />
   <param name="CodebookFilename" value="$(find mod_base)/XXXX.eod.bin"/>
   <param name="LoadCodebookFilename"value="$(find mod_base)/XXXX.eod.bin"/>
   <param name="Interface" value="SHOW" />
   <param name="PublishResults" value="PUBLISH" />
   <param name="WriteToFile" value="" />
   <param name="ImageSize" value="VGA" />
   <param name="LearnObjNo" value="obj0" />
   <param name="depthss" value="s" />
</node>
```

El parámetro “CodebookFilename” debe tener como “value” el nombre del archivo que contiene el entrenamiento. Estos tienen una extensión “eod.bin”.

Guardar un nuevo entrenamiento. El parámetro llamado “WriteToFile” debe tener el mismo valor que el del archivo “CodebookFilename”. Con los cambios incluidos, lanzar la aplicación y realizar un entrenamiento del objeto, al cerrar la aplicación el archivo se ha guardado en la ubicación seleccionada con el entrenamiento realizado.

Cargar un nuevo entrenamiento. El parámetro llamado “LoadCodebokFileName” debe tener el mismo valor que el del archivo “CodebookFilename”. Con los cambios incluidos, lanzar la aplicación y desde el inicio si este detecta un objeto entrenado lo notificará.

4.4.3 Impresión por pantalla durante la aplicación

Durante el uso de la aplicación por pantalla del terminal (véase Figura 4.3) podemos ir observando el ángulo y la distancia a la que se encuentra el objeto detectado.

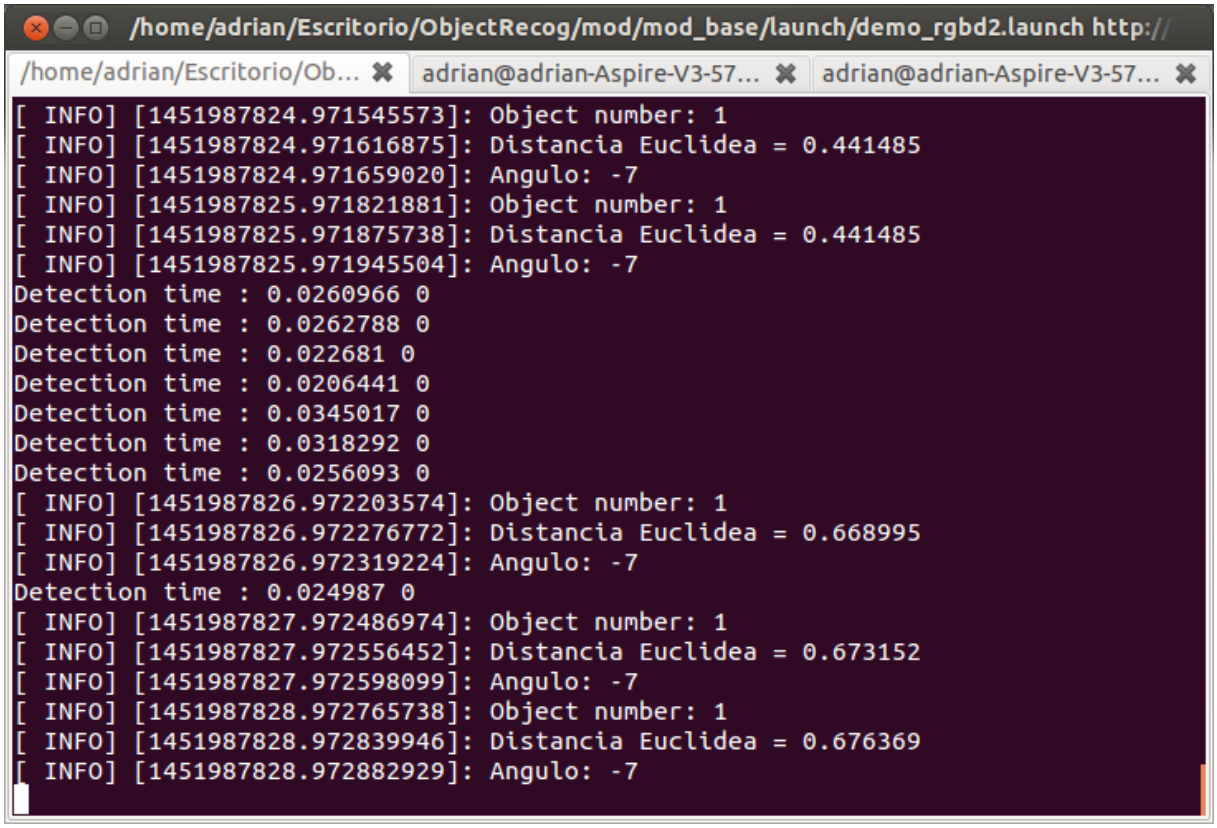


Figura 4.3: Terminal

Capítulo 5

Conclusiones y líneas futuras

"Si al final valió la pena, el camino no importa."

Victor Rutty.

En este apartado se resumen las conclusiones obtenidas y se proponen futuras líneas de investigación que se han derivado del trabajo.

5.1 Conclusiones

En el presente proyecto final de grado se ha elaborado un sistema de localización de objetos para personas invidentes. Para ello se ha hecho uso del dispositivo de captura Kinect V1 y un dispositivo CPU para la ejecución del algoritmo que se ha desarrollado. Este algoritmo se encuentra elaborado sobre el entorno ROS y, ha requerido de varios conocimientos en visión artificial, tales como: detección de bordes a partir de color y profundidad, entrenamiento y detección de objetos en tiempo real y seguimiento de los mismos mediante Filtros de Kalman. Además de otras técnicas empleadas como la localización de fuentes sonoras mediante sonidos binaurales.

El primer enfoque del proyecto ha sido dirigido a la mejora del algoritmo de partida con la finalidad de solucionar aquellos aspectos en los que este no era eficaz. Para ello, se realizo una mejora mediante la inclusión de los bordes obtenidos a partir de la información de profundidad en las fases de entrenamiento y detección, esto permitió la obtención de la totalidad de los contornos de los objetos en las situaciones en las que no se podía establecer un contraste de color, permitiendo tanto un mejor entrenamiento como una mejor posterior detección de los mismos. Aunque la inclusión de estos bordes resultó totalmente satisfactoria, el algoritmo seguía sin ser del todo eficaz en la detección de objetos en entornos altamente poblados.

También con la misma finalidad que la solución anterior, se propuso realizar un seguimiento de los objetos detectados con el fin de evitar aquellas situaciones en las que las detecciones de los objetos era intermitente, y así, poder seguir transmitiendo información al usuario aunque se desconociera la posición exacta del objeto.

Por último, se realizó un estudio de la técnica empleada para la localización de fuentes sonoras a partir de sonidos binaurales. En este experimento se pudo contar con personas invidentes, pues son

las destinadas a hacer uso de este dispositivo. Tras varias pruebas realizadas, se han obtenido unos resultados en la localización binaural muy precisos, incluso sin realizar ningún entrenamiento previo para esta técnica. Además se ha obtenido una muy buena aceptación del prototipo por parte de las personas invidentes con citas como:

"Para una gran cantidad de personas invidentes, sobre todo para gente despistada o que conviva con más gente que puedan mover las cosas de sitio, le vendría de maravilla.", Juan Antonio Saiz.

"Es un dispositivo muy interesante y tendría buena aceptación, te animo a que sigas con ello.", Francisco Marfil.

Como conclusión final de este proyecto, se ha podido estudiar e implementar diversas técnicas empleadas en la visión artificial, tales como: sensores de profundidad, obtención de bordes a partir de diversas técnicas y seguimiento de objetos a partir del Filtro de Kalman. Además de otras técnicas novedosas como la localización mediante sonidos. Pero sin duda, si hay que destacar algo por encima de todo, es la satisfacción de haber podido realizar un prototipo que puede permitir mejorar el día a día de las personas con discapacidad visual, y encima el hecho haber obtenido una buena crítica por su parte, lo hace aún más interesante para haber sido y, seguir siendo, una futura línea de investigación.

5.2 Líneas futuras

Las posibles líneas futuras para la continuación de este presente proyecto son la siguientes.

Ejecución del algoritmo sobre una placa con microcontrolador tal como Arduino, Raspberry Pi o dispositivo similar. Así se puede prescindir de utilizar un ordenador portátil, como el empleado, para la ejecución del algoritmo y hacer así al sistema mucho menos robusto y por tanto de más fácil manejo, transporte y aceptación.

Utilización de un detector de objetos que nos proporcione una mayor tasa de detecciones en entornos altamente poblados, pues es en estos entornos donde las personas invidentes van a hacer uso de este algoritmo.

Mejorar el seguimiento de los objetos, permitiendo la ejecución simultánea de varios Filtros de Kalman debido a que el algoritmo permite almacenar múltiples aprendizajes de objetos y permite una detección de varios objetos sobre la misma imagen.

Mejora de la localización binaural incluyendo una localización de los objetos en el plano vertical, haciendo así el sistema mucho más completo.

Bibliografía

[1] "Kinect | xbox 360," http://www.xbox.com/en-US/xbox-360/accessories/kinect.

[2] D. Damen, P. Bunnun, A. Calway, and W. W. Mayol-Cuevas, "Real-time learning and detection of 3d texture-less objects: A scalable approach." in *BMVC*, 2012, pp. 1–12.

[3] "Página organización mundial de la salud," http://www.who.int/es/.

[4] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–4.

[5] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.

[7] F. Remondino and D. Stoppa, *TOF range-imaging cameras.* Springer, 2013, vol. 68121.

[8] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4–10, 2012.

[9] J. A. M. Rojas, J. A. Hermosilla, R. S. Montero, and P. L. L. Espí, "Physical analysis of several organic signals for human echolocation: oral vacuum pulses," *Acta Acustica united with Acustica*, vol. 95, no. 2, pp. 325–330, 2009.

[10] "Bastón egara," http://www.instead-technologies.com/sample-page-2/.

[11] J. E. Rincón Ruiz, "Diseño y construcción de un dispositivo electrónico para la detección de obstáculos, como ayuda a personas con discapacidad visual," 2010.

[12] "Orcam," http://www.orcam.com/.

[13] G. Parseihian, A. Brilhault, and F. Dramas, "Navig: an object localization system for the blind," in *Workshop on Multimodal Location Based Techniques for Extreme Navigation (Pervasive 2010), Helsinki, Finland*, 2010.

[14] B. Holton, "A review of the taptapsee, camfind, and talking goggles object identification apps for the iphone," 2013.

[15] J. Sudol, O. Dialameh, C. Blanchard, and T. Dorcey, "Looktel-a comprehensive platform for computer-aided visual assistance," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on.* IEEE, 2010, pp. 73–80.

[16] M. S. Magallón, "Sistema interactivo para manejo de electrodomésticos en entornos domésticos, apéndice a," Ph.D. dissertation, Universidad de Zaragoza, 2013.

[17] A. Opelt, A. Pinz, and A. Zisserman, "A boundary-fragment-model for object detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 575–588.

[18] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 503–510.

[19] O. Carmichael and M. Hebert, "Object recognition by a cascade of edge probes," 2002.

[20] O. Danielsson, S. Carlsson, and J. Sullivan, "Automatic learning and extraction of multi-local features," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 917–924.

[21] R. G. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "Lsd: A line segement detector," *IEEE Trans. on Pattern and Machine Intelligence (PAMI)*, vol. 32, no. 4, 2010.

[22] Y. Lamdan and H. J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," 1988.

[23] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of geometric hashing," in *Computer Vision, 1990. Proceedings, Third International Conference on*. IEEE, 1990, pp. 334–338.

[24] J. S. Beis and D. G. Lowe, "Indexing without invariants in 3d object recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 10, pp. 1000–1015, 1999.

[25] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy, "Canonical frames for planar object recognition," in *Computer Vision ECCV92*. Springer, 1992, pp. 757–772.

[26] C. S. C Wiedemann, M Ulrich, "Recognition and tracking of 3d objects. in dagm symposium on pattern recognition," 2008.

[27] S. Fidler, M. Boben, and A. Leonardis, "A coarse-to-fine taxonomy of constellations for fast multi-class object detection," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 687–700.

[28] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1696–1703.

[29] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," 2010.

[30] "Página del framework ros," http://www.ros.org.

[31] "Página de la librería pcl," http://www.pointclouds.org/.

[32] "Página de la librería OpenCv," http://www.opencv.org.

[33] "Lozalización del sonido." http://www.eumus.edu.uy/docentes/maggiolo/acuapu/loc.html.

[34] "The jeffress model - animation," https://auditoryneuroscience.com/topics/jeffress-model-animation.

[35] "Efecto haas," http://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_06_07/io1/public_html/haas.htm.

[36] "Organización nacional de ciegos españoles, once," http://www.once.es/new.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá