UNIVERSIDAD CARLOS III

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

PROYECTO FIN DE CARRERA

RECONOCIMIENTO DE OBJETOS MEDIANTE SENSOR 3D KINECT

Autor: Javier Nuño Simón

Tutor: Alejandro Lumbier Álvarez

GRADO INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

LEGANÉS, MADRID

SEPTIEMBRE 2012

A mi familia

A mis amigos por los "mocasines"

A mis amigos estelares por sus ideas

Índice general

Índice de figuras

Resumen

Actualmente en el mundo de la robótica se están realizando grandes avances en el ámbito del tratamiento de imágenes y de la inteligencia artificial.

Algo que a las personas nos parece tan trivial como distinguir entre diferentes objetos, es más complicado de realizar mediante un ordenador. Este no tiene la capacidad de nuestro cerebro de unir los puntos que pertencen al mismo objeto y descartar aquellos que pertenecen al entorno.

Siguiendo esta línea de trabajo, este proyecto de fin de carrera se centra en el análisis de un entorno en el que podemos encontrar un objeto de forma que el ordenador sea capaz de identificarlo.

Palabras clave: visión, reconocimiento, objetos, kinect.

Abstract

Nowadays there are being great advances in robotics, specially in the field of image processing and artificial intelligence.

Something that humans find so trivial like distinguish between different objects, is much more complex to perform by a computer. It does not have our brain ability to unite the points belonging to the same object and discard those which belong to the enviroment.

Following this line of argument, the goal of our dissertation is to analize the enviroment, where is located an object, in order to make the computer identify that object.

KEYWORDS: vision, recognition, objects, kinect.

Índice general

Índice de figuras

Capítulo 1

Introducción

Como humanos percibimos las estructura tridimensional del mundo que nos rodea con facilidad. Prueba de ello lo constatamos al mirar a nuestro alrededor. Somos capaces de identificar todos los objetos que nos rodean y distinguir los puntos que pertenecen a un objeto de los que corresponden al entorno. Psicólogos han dedicado décadas a intentar entender como funciona nuestro sistema visual para poder aplicarlo a las máquinas. La solución al puzle sigue incompleta.

Los investigadores de la visión por computador han estado trabajando en diferentes técnicas matemáticas para encontrar la forma tridimensional de los objetos. Actualmente se ha conseguido una imagen bastante precisa en 3D de un objeto cualquiera a partir de una serie de fotografías solapadas. Incluso podemos intentar que el ordenador nos reconozca el número de personas de una fotografía e intentar nombrar a cada una de esas personas [20]. Sin embargo, a pesar de todos estos avances, el sueño de que un ordenador sea capaz de interpretar una imagen como nosotros, sigue siendo algo difícil.

¿Por qué la visión es algo tan complicado? Debemos distinguir entre visión y percepción. Cualquier cámara del mundo es capaz de ver el mundo exactamente como nosotros, pero no es capaz de interpretarlo. En parte, es debido a que la visión es un problema inverso. Intentamos encontrar las incógnitas que nos den la información para solucionar el

problema por completo y por ello recurrimos a modelos físicos y de probabilidad para diferenciar entre las posibles soluciones.

En visión por ordenador queremos conseguir describir el entorno que vemos en una o más imágenes de modo que reconstruyamos sus propiedades tales como forma, brillo y distribución de color. Es increíble que los humanos y los animales hagamos esto sin esfuerzo, mientras que los algoritmos de visión por ordenador tienen infinidad de debilidades.

1.1. Un breve recorrido por la historia

La historia de la visión por ordenador comienza a principios de los años 60. Era una de las partes que componían el avance en inteligencia artificial. Por aquel entonces se pensaba que resolver el problema de la visión sería mucho más sencillo. Como ejemplo de este pensamiento podemos contar una anécdota que ocurrió en 1966 con el profesor Marvin Minsky fundador del laboratorio de inteligencia artificial del MIT. Éste propuso a uno de sus alumnos más aventajados que en el verano de 1966 fuera capaz de crear un algoritmo que consiguiera que el ordenador describiera todo lo que veía a su alrededor. El alumno intentó el reto pero fue imposible llevarlo a cabo. Ahora sabemos que el problema es algo más complicado [24].

Lo que diferencia la visión por ordenador de otros campos existentes de procesamiento de imágenes digitales es el deseo de encontrar la estructura tridimensional del mundo a través de imágenes, y usarlo para comprender completamente cualquier escena [32].

Lo primero que se debía hacer para reconocer una escena era encontrar los bordes de los objetos que la componían, para así poder encontrar la estructura del objeto. En los 60 se empezaron a desarrollar algunos algoritmos en este aspecto [18, 7].

También se empezó a investigar sobre la modelización de los objetos no poliédricos

[3]. La aproximación más popular era la utilización de modelos basados en cilindros, algo que se sigue usando hoy en día [1]. Otros muchos avances de esta primera época de la visión por ordenador son, por ejemplo, los primeros de muchos algoritmos basados en la visión estereoscópica, es decir, la creación de un modelo tridimensional a partir de fotografías del mismo objeto desde distintos puntos de vista [26].

Un importante aporte a como se creía que funciona la visión fue resumida por David Marr en su libro *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information* (1982). En su libro, Marr trataba a la visión como un sistema de procesamiento de información que dividía en tres niveles:

- Nivel computacional: describe qué es lo que hace el sistema (qué problemas tiene que resolver y superar) y por qué lo hace.
- Nivel de representación o de algoritmo: describe cómo se puede implementar esta teoría computacional, cuál es la representación del input/output y cuál es el algoritmo para la transformada.
- Nivel físico: describe cómo puede realizarse físicamente el algoritmo y la representación.

Este acercamiento de Marr sigue siendo válido para los problemas actuales.

En los 80 siguieron centrándose en desarrollar técnicas matemáticas más sofisticadas para mejorar el análisis de imágenes y escenas. Se empezó a extender el uso de la pirámides de imágenes para mejorar la búsqueda del contorno de las imágenes procesadas [31]. Paralelamente también se introdujo otro concepto para la detección del contorno y margen de la imagen, *snake* [22].

Los investigadores se dieron cuenta de que muchos de los algoritmos desarrollados se podían unificar, o por lo menos describir, en el mismo marco matemático, si se planteaban como problemas de optimización variable, lo que los haría mucho más robustos [36].

En la misma época, Geman y Geman [12] propuso que esos mismos problemas se podrían formular usando un campo aleatorio de Markov (MRF de sus siglas en inglés, Markov Random Field) que permitiría el uso de mejores algoritmos de búsqueda y optimización.

En los años 90 se empezaron ciertos programas de investigación entorno al reconocimiento de objetos y a los problemas relacionados con el movimiento. Todos estos avances fueron posibles gracias a los continuos progresos en la tecnología, como son las cámaras digitales (cada vez más precisas y con mayor resolución) y a los ordenadores (cada vez más rápidos y con mayor capacidad de procesamiento).

Otro tema muy activo durante esta década fue el perfeccionamiento de los algoritmos para la creación de modelos en 3D a partir de distintas fotografías tomadas desde varios puntos de vista, así como técnicas de descripción de volúmenes tridimensionales a partir de siluetas [34].

Se mejoraron enormemente los algoritmos de seguimiento de contornos entre los que encontramos el de contornos activos (*snake*) desarrollado en los 80. Estos se comenzaron a aplicar para el seguimiento de caras y de cuerpos de personas.

También empezaron a aparecer las primeras técnicas de aprendizaje estadístico cuya principal aplicación es la de reconocimiento de caras [37].

A partir del año 2000 tomó más importancia el desarrollo y la mejora de los algoritmos de reconocimiento de objetos, así como los de interpretación de escenas [30] y [44]. Se siguió por ese camino pues es el principal objetivo del campo de la visión por ordenador dentro de la inteligencia artificial. La máquina tiene que ser capaz de analizar lo que la rodea para poder determinar las acciones correctas.

Capítulo 2

Objetivos

Antes de establecer los objetivos principales de este proyecto es necesario conocer el campo del conocimiento en que está englobado. En este caso, el proyecto trata sobre la visión por ordenador, es decir, captación de imágenes y su posterior tratamiento con el fin de reconocer ciertos objetos.

Queda claro que el principal propósito es la obtención de un algoritmo capaz de identificar ciertos objetos que se muestren a la cámara. Para poder llegar a alcanzarlo, se tienen que dar unos pasos previos, que serán los objetivos secundarios que habrá que cumplir.

1. Comprender el funcionamiento del sensor Kinect de Microsoft para poder utilizarlo en la captura de una escena.

2. Capturar la escena en la que encontramos el objeto a identificar, mediante el sensor Kinect previamente mencionado.

3. Procesar la imagen para quedarnos únicamente con los datos útiles. En este paso será necesario filtrar ruido, remover información innecesaria y hacer la segmentación para el posterior análisis de la imagen.

4. Analizar la imagen procesada para poder dar la solución a nuestro problema inicial, identificar el objeto que tenemos delante de la cámara, a partir de una lista de cinco objetos: una lata de refresco común, un rotulador, una pelota de tenis, un balón de fútbol y una caja.

5. Por último, reslizar una interfaz amigable para exponer los resultados del reconocimiento al usuario.

Así entonces, se busca diseñar un algoritmo capaz de resolver todos estos puntos, de forma eficaz, robusta y en tiempo real.

Capítulo 3

Estado del arte

El problema de capturar y reconstruir los objetos tridimensionales del mundo real para poder reconocerlos se puede dividir en: adquisición de datos, filtrado, segmentación y reconocimiento.

3.1. Adquisición de datos

Las tecnologías para obtener datos con profundidad se pueden separar en dos categorías: aquellas que precisan contacto y las que no lo requieren [9]. Los de contacto siempre han sido bastante más precisos pero lentos a la hora de obtener la información. Por otro lado, los que no requieren contacto han mejorado significativamente la precisión en los últimos años.

Dentro de las tecnologías que no precisan contacto podríamos hablar de: [41]

- Activas: obtienen la información de la forma del objeto emitiendo y recibiendo cierto tipo de radiación, como puede ser ultrasonidos, rayos X o luz.
- Pasivas: no emiten ningún tipo de radiación, pero si que detectan la radiación ambiental.

Como el objetivo de este proyecto no está relacionado con la adquisición de datos mediante métodos de contacto, pasamos a estudiar los procedimientos más importantes que no necesitan contacto.

3.1.1. Método RGB

Los métodos basados en RGB estarían clasificados dentro de los pasivos. Estos procedimientos confían en la existencia de luz visible para determinar la forma del objeto. Existen numerosos métodos que se pueden incluir en esta categoría.

Vouzounaras propuso un método para determinar la geometría tridimensional de un objeto a partir de una única imagen RGB usando la detección de los puntos de fuga [39]. Este método cuenta con el conocimiento que se tiene de las estructuras geométricas tales como los planos ortogonales. Requiere la presencia del suelo, del techo y de las paredes para que el algoritmo funcione. Son capaces de reconstruir objetos de geometría sencilla, pero sin embargo, no pueden hacer lo mismo con objetos esféricos.

Saxena llevó a cabo una investigación sobre la reconstrucción de escenas en 3D a partir de una o muy pocas imágenes tomadas con una cámara de RGB [33]. Lo primero que hacían era sobresegmentar la imagen en pequeños parches llamados superpíxeles. Simultáneamente intentan encontrar la posición y orientación tridimensional de cada uno de esos superpíxeles. Para hacer esto utilizan el campo aleatorio de Markov (MRF).

Weiss estudió la posibilidad del reconocimiento de objetos a partir de una imagen en 2D usando invariantes geométricas [40]. Esas invariantes las encontraban al hacer ciertas suposiciones para modelos tridimensionales de los objetos. Esas hipótesis podían ser para un modelo particular o para uno más general. Después, a través de ciertas relaciones algebraicas eran capaces de describir modelos invariantes en 3D.

Zheng ideó un método para construir modelos en 3D a partir de múltiples imágenes RGB sin la necesidad de calibrar la cámaras antes de tomar las imágenes [46]. Combinaba técnicas como la detección de esquinas de Harris (Harris corner detection) y la detección

de líneas para la obtención del objeto. Este método, sin embargo, requiere el uso de 4 cámaras y una mesa que rote para poder captar al objeto en toda su integridad.

Analizando la documentación sobre los métodos basados en RGB podemos comprobar que existe una gran desventaja. Todas estos procedimientos requieren un paso computacional adicional para la detección de la profundidad. También pueden necesitar ayuda física extra, como las cuatro cámaras y la mesa rotatoria. Además, la gran mayoría de estos métodos sólo detectan ciertos tipos de objetos, normalmente los que son planos, mientras que los esféricos los pasan por alto.

3.1.2. Método estereográfico

Los métodos estereográficos están categorizados en el mismo grupo que los de RGB, no necesitan contacto y son pasivos. De todos modos, existe una gran diferencia entre los dos. El estereográfico necesita dos cámaras de RGB posicionadas de cierta manera y previamente calibradas. Debido a la facilidad de adquirir este tipo de cámaras en la actualidad, existen multitud de investigaciones acerca de la construcción de un mapa de profundidad a partir de las mismas.

Huang propone un método para capturar la imagen tridimensional usando una cámara estereográfica y utilizando el teorema que permite obtener la forma del objeto a partir de su sombra (shape-from-shading) [16]. Como ya usara Zheng [46], se usa una mesa rotatoria para capturar al objeto desde todas las perspectivas. El núcleo de su investigación se basa en la detección de los puntos clave del objeto, como las esquinas, lo que resulta ineficaz para los objetos esféricos.

Jiajun Zhu usa cuatro pares de cámaras para conseguir el efecto estereográfico. Las coloca de tal forma que consigue cubrir los 360 grados del objeto [47]. En esta solución, cada una de las ocho cámaras (cuatro pares) capturan 18 imágenes de alta resolución en color con diferentes patrones de luz (producidas con el flash) para capturar la posición del objeto. Para conseguir ver toda una escena correctamente, a veces se necesitan capturas desde diferentes posiciones. Este método nos permite la obtención de un mapa de

profundidad de gran calidad así como las texturas de la imagen. El inconveniente es que para conseguirlo los tiempos de procesado de imagen son bastante elevados, llegando a 50 minutos por captura de una posición.

Moro propone un método de adquisición de la profundidad de una imagen mediante cámaras estereográficas. También va acompañado de varios algoritmos de las regiones de interés para extraer ciertos modelos de los objetos de la escena previamente entrenados [28]. La necesidad de enseñar los modelos de los objetos antes de la captura de la imagen limita la utilidad de esta aproximación.

Hub hizo un buen trabajo sobre el reconocimiento y el rastreo de objetos móviles para ayudar a los invidentes [17]. Utilizan una cámara estereográfica acoplada a la cabeza del usuario de forma que el sistema va informando de las existencia de objetos, y qué clase de objetos se tratan, si se pueden mover o por el contrario hay que evitarlos.

3.1.3. Método Time-of-flight

Los métodos basados en time-of-flight (tiempo de vuelo) calculan la distancia del objeto a partir del tiempo que tarda un pulso de radiación emitida de luz infrarroja, en ir y volver. Por este motivo, estos procedimientos se incluyen en el grupo de sin contacto activo.

Pheat describe un método en el que usa dos diodos láser y una cámara digital. También usan una mesa rotatoria para poder ver al objeto desde todos los ángulos. Normalmente necesitan entre 200 y 400 escaneos para reconstruir un simple objeto [29].

Hagebeuker nos introduce una nueva generación de cámaras de tipo time-of-flight en [13]. Este nuevo tipo de cámaras, llamadas PMD (Photonic Mixer Device), tienen las ventajas de ser más rápidas que sus predecesoras y tener mayor resolución lateral además de seguir aportando información fiable de profundidad.

Los métodos basados en time-of-flight son los más antiguos, así como los más conocidos para medir profundidad. Además nos proporcionan gran precisión en los datos. Sin embargo, el principal inconveniente es su alto precio.

3.1.4. Método de luz estructurada

Las cámaras de profundidad que usan el método de luz estructurada normalmente contienen una cámara RGB normal junto a un emisor y receptor de luz infrarroja. A este tipo de cámaras se las llama habitualmente RGB-D y cada vez son más fáciles de conseguir para los consumidores. Tenemos un gran ejemplo con el dispositivo Kinect de Microsoft, el núcleo de nuestro proyecto. Las cámaras RGB-D estarán incluidas dentro del grupo de sin contacto activo.

Existe numerosa documentación acerca de la adquisición de datos a través de la Kinect. En el artículo publicado por Cruz [8] podemos aprender sobre los últimos avances llevados a cabo con dicho sensor. También podemos leer acerca de los retos que se presentan así como la mutitud de aplicaciones que se pueden llevar a cabo con este dispositivo.

Izadi [19] propone un sistema para capturar ciertos objetos dentro de una escena a través del sensor Kinect. En este caso, primero se reconstruye la imagen por completo, después el usuario mueve el objeto que queremos identificar, lo que consigue separar el objeto del resto de la escena.

Otro ejemplo de captura de datos a través de una cámara RGB-D lo encontramos en [14]. Hacen un estudio acerca del uso que se puede dar de estas cámaras en el ámbito de la robótica, especialmente para hacer mapas tridimensionales de entornos interiores. Con esos mapas podemos desarrollar aplicaciones de navegación de robots.

3.2. Métodos actuales de segmentación

La segmentación siempre ha sido una parte esencial en el proceso de modelado a partir de los datos sobre los puntos de la imagen [38]. La segmentación es el proceso por el cual somos capaces de dividir una nube de puntos en distintas regiones de interés o extraer las características más importantes de los puntos a partir de los datos de cada uno de ellos. La gran mayoría de los métodos de segmentación se pueden clasificar en tres categorías: métodos de detección de bordes, métodos de crecimiento de regiones y métodos híbridos.

Los métodos de detección de bordes intentan detectar discontinuidades en las superficies que forman los límites de los componentes. Fan [10] usa las propiedades de las superficies curvas para identificar esos límites del objeto. Para evitar la pérdida de localización se usa un sistema de seguimiento que pasa todos los datos por máscaras gaussianas con diferentes parámetros de dispersión.

Por otro lado, Chen y Liu [6] segmenta los datos mediante el corte de los mismos y su posterior aproximación a las curvas NURBS (acrónimo inglés de non-uniform rational B-spline) en dos dimensiones. Los puntos que pertenecen al límite los detecta calculando la curvatura máxima del NURBS.

Milroy [27] usa una aproximación semiautomática basada en la detección de los bordes para los modelos de sección ortogonal cruzada. Las propiedades diferenciales de la superficie son estimadas en cada punto del modelo, y los extremos de la curvatura son identificados como posibles puntos del borde. Después se usa un minimizador del contorno activo para unir a los puntos del borde.

Yang y Lee [42] identifican los puntos del borde como extremos mediante la estimación de la curvatura de la superficie. Después de encontrarlos, se usa un algoritmo de detección de vecinos para la formación de las curvas del borde.

Por otro lado tenemos los métodos de crecimiento de regiones. Estos realizan la segmentación mediante la detección de superficies continuas homogéneas o que tengan propiedades geométricas similares. Hoffman y Jain [15] segmentan la imagen en diferentes parches de superficie y los clasifican como planos, cóncavos o convexos, a partir de test estadísticos de tendencia, valores de curvatura y el análisis de los autovectores.

Besl y Jain [4] desarrollaron un método de segmentación basado en el ajuste del orden de variable de una superficie. Usaron polinomios de orden inferior de dos variables para aproximarlo a un rango de datos. A través de la estimación de la curvatura gaussiana y la curvatura media fueron capaces de identificar ciertas regiones centrales. Fue entonces cuando aplicaron el método de crecimiento de regiones para encontrar todos los bordes.

La mayoría de los métodos basados en discontinuidades son bastante rápidos cuando una parte del borde muestra pequeñas diferencias o cuando las regiones son homogéneas. Los métodos de segmentación basados en regiones permiten una homogeneidad bastante correcta en una región pero suelen fallar al localizar los puntos que se encuentran fuera de dicha región de forma precisa. Para resolver estas limitaciones se desarrollaron los métodos híbridos donde se combinan ambos procedimientos.

El método propuesto por Yokoya y Levine [43] divide los datos tridimensionales en las superficies primitivas usando un ajuste bicuadrático para superficies. Los datos segmentados son homogéneos en cuanto a diferentes propiedades geométricas y no contenían discontinuidades. Se calculan las curvaturas gaussiana y media para realizar la segmentación por regiones inicial. A continuación, tras emplear dos segmentaciones basadas en bordes a partir de las derivadas parciales y de los valores de profundidad, aplicamos la segmentación final a los datos iniciales.

Checchin [5] usa una aproximación híbrida que combina la detección de los bordes basada en las normales a la superficie y el método de crecimiento de regiones para unir regiones sobresegmentadas.

Zhao y Zhang [45] emplean un método basado en triangulación y agrupamiento de

regiones que usa los bordes, los puntos críticos y las normales de la superficie. En su algoritmo, los bordes iniciales y los puntos críticos se detectan usando los residuos morfológicos, y se triangulan los pares del borde. Para la segmentación final, los pequeños componentes de la red triangular que forman la superficie se expanden y son segmentados de acuerdo con la información de las normales de la superficie.

3.3. Métodos de reconocimiento de objetos

Una de las principales aplicaciones de la visión por computador es la de reconocer los objetos que nos rodean. Cuando seamos capaces de enseñar a los robots a interpretar el entorno y reconocer cada uno de los objetos que componen una escena, habremos dotado a la máquina de una gran independencia para poder tomar decisiones por sí misma.

Por ese motivo hay gran cantidad de estudios entorno al reconocimiento de objetos. Una de las líneas de investigación que más se está siguiendo actualmente en este aspecto está relacionado con las bases de datos. Podemos ver un ejemplo de una gran base de datos en la publicación de Lai [23]. Esta gran base de datos tiene 300 objetos organizados en 51 categorías diferentes.

Tang describe en su investigación como crea una gran base de datos con multitud de datos y posiciones de diferentes objetos cotidianos [35]. Gracias a la enorme cantidad de datos de cada objeto consigue una detección altamente eficaz. El sistema es capaz de manejar correctamente obstrucciones, cambios de iluminación, multiples objetos y diferentes casos dentro del mismo objeto.

El método propuesto por Kasper habla de la creación de una gran base de objetos para su posterior utilización en reconocimiento y manipulación de objetos [21].

Ahn usa otro enfoque para el reconocimiento de objetos cotidianos dentro de una escena [2]. Utiliza un sistema de visión para robots conocido como SLAM (Simultaneous localization and mapping). En este caso lo utiliza para el reconocimiento de objetos.

Para aplicar bien el método SLAM debe seguir tres pasos. Primero debe extraer las características invariantes. Después aplica un agrupamiento RANSAC (RANdom SAmple Consensus) para el reconocimiento de objetos en presencia de puntos externos. Por último calcula información métrica precisa para actualizar el SLAM.

Capítulo 4

Conceptos teóricos

Si se quiere tener éxito a la hora de diseñar el algoritmo, se deben estudiar antes los elementos que van a estar involucrados en el proyecto. Hay que familiarizarse con sus características principales, lo cual facilitará el trabajo a la hora de la implementación.

4.1. Fisiología del ojo humano

Para entrar en el tema de visión se hace imprescindible tener unas nociones básicas de la fisiología del ojo humano, ver su morfología y cómo funciona.

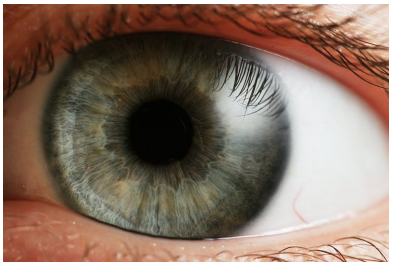

Figura 4.1: *Ojo humano*

4.1.1. Funcionamiento

El ojo humano tiene unos 25 mm de diámetro y es el responsable de captar la luz que se refleja en los objetos, la cual se trasmite desde la córnea hasta la retina atravesando una cámara rellena de un líquido acuoso. En este camino nos encontramos con el cristalino que, junto con la córnea, actúa a modo de lente para focalizar la luz sobre la retina.

La retina se sitúa al final de todo el globo ocular y en ella se encuentran una serie de células fotosensibles encargadas de transformar la luz en impulsos eléctricos, los cuales son transmitidos al cerebro a través del nervio óptico.

4.1.2. Morfología del ojo humano

Los principales elementos que lo componen son:

1. Córnea: elemento transparente que recubre el globo ocular.
2. Iris: es la membrana donde está situada la pupila.
3. Pupila: actúa a modo de diafragma adecuándose a la cantidad de luz del entorno, permitiendo que se transmita más o menos luz.
4. Cristalino: su función es la de permitir enfocar los objetos, lo que consigue modificando la curvatura.
5. Humor acuoso: es un medio gelatinoso cuya función es el transporte de los elementos necesarios para nutrir a la córnea y el cristalino.
6. Humor vítreo: similar al humor acuoso, pero algo más denso y que es el encargado de dar su forma esférica al globo ocular.
7. Retina: se encuentra en la zona más interna del globo ocular y en ella se sitúan millones de células fotosensibles (los conos y bastones) encargadas de captar toda la luz que penetra en el ojo y transformarla en impulsos eléctricos.
8. Conos: estas células son muy sensibles al color. Hay en torno a 7 millones, se encuentran localizadas en la fóvea y tienen su propia terminación nerviosa que las

conecta directamente al cerebro. Existen 3 tipos distintos, cada una de ellas especializadas en captar una longitud de onda en concreta, correspondientes al rojo, verde y azul.

9. Bastones: estas células son sensibles a la intensidad lumínica. Su número es muy superior a los conos, situándose entre los 70 y los 140 millones, que se distribuyen por toda la retina.

10. Fóvea central: situada en la retina, es el punto donde la agudeza visual es máxima ya que es aquí donde se enfocan los rayos de luz.

11. Músculo ciliar: es el encargado de modificar la forma del cristalino.

12. Nervio óptico: a través de él se transmiten las señales eléctricas generadas en la retina al cortex cerebral.

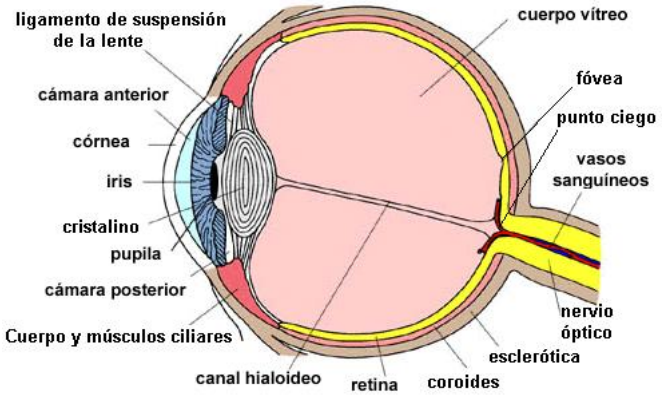

Figura 4.2: *Diagrama del globo ocular humano*

4.2. El color

La longitud de onda que los humanos son capaces de distinguir es una porción muy pequeña dentro de todo el espectro electromagético. En concreto, está acotado entre los 380 nm y los 780 nm tal y como se ve en la Figura 4.3.

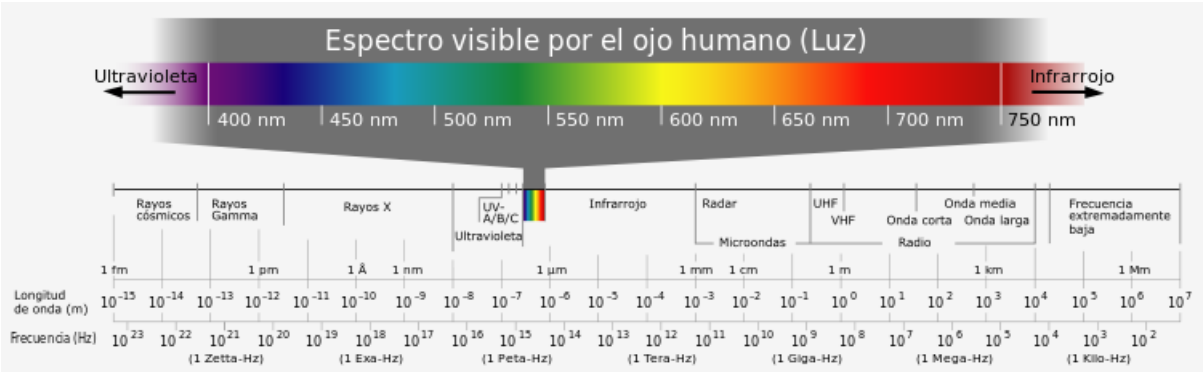

Figura 4.3: *Espectro de la luz*

En este proyecto utilizamos dos intervalos del espectro de la luz, el espectro visible y el infrarrojo. El rango de espectro visible es el que maneja la cámara de RGB que monta Kinect para capturar la escena. Por otro lado, también dispone de un sensor de profundidad que, a través de un haz de luz infrarroja, es capaz de obtener el dato de profundidad en todos los puntos.

4.3. Filtrado

Los nuevos dispositivos de adquisición de datos en tres dimensiones producen unos ficheros de datos de gran densidad debido a la precisión milimétrica. Los algoritmos de reconstrucción de superficies fallan cuando intentan manejar tal cantidad de datos. Para facilitar el procesamiento de los mismos se requiere una simplificación para ahorrar tiempo y memoria.

Al simplificar la nube de puntos aceleramos de forma considerable el problema de la reconstrucción y evitamos algunos de los errores que surgen por la gran cantidad de los datos.

Unos de los filtros que utilizamos en nuestro proyecto es el que llamamos de densidad. Lo definimos así porque quita densidad en la nube de puntos puntos, de forma que no quita ningún tipo de información relevante.

El funcionamiento de este filtro que aplicamos se basa en la búsqueda del centroide dentro de la nube de puntos. El algoritmo divide toda la nube de puntos en pequeños cubos con las dimensiones que indiquemos al programa. Una vez creados dichos cubos, calcula el centroide de cada uno a partir de la información de los puntos que se encuentren dentro.

La dimensión de los cubos es importante pues será el indicador que haga que el filtro nos suprima más o menos datos. Cuanto más grandes sean dicho cubos, mayor cantidad de puntos habrá dentro de cada uno y por tanto más información eliminaremos. Por el contrario cuanto más pequeños sean los cubos, menor será la suma de puntos que filtraremos.

En geometría el centroide de un objeto *X* perteneciente a un espacio *n*-dimensional es la intersección de todos los hiperplanos que dividen a *X* en dos partes de igual *n*-volumen con respecto al plano. Informalmente, es el promedio de todos los puntos de *X*.

Para calcular el centroide se puede hacer de varias maneras, a partir de la fórmula integral o mediante la suma de un conjunto finito de puntos. Mediante la fórmula integral se tendría que aplicar para un espacio R^n :

$$C = \frac{\int xg(x)dx}{\int g(x)dx} \tag{4.1}$$

Por otro lado también se podría hallar el centroide a partir de un conjunto finito de puntos. Esta fórmula es una media. Podemos ver como quedarían las coordenadas del centroide en las siguientes ecuaciones:

$$C_x = \frac{x_1 + x_2 + x_3 + ... + x_k}{k} \tag{4.2}$$

$$C_y = \frac{y_1 + y_2 + y_3 + ... + y_k}{k} \tag{4.3}$$

$$C_z = \frac{z_1 + z_2 + z_3 + ... + z_k}{k} \tag{4.4}$$

4.4. Cálculo de normales

Para llevar a cabo nuestro objetivo tendremos que recurrir al cálculo de las normales de la superficie. Mediante la información que conseguimos con las normales, seremos luego capaces de llevar a cabo una correcta segmentación.

En toda superficie, ya sea plana o curva, podemos calcular los vectores normales en cada uno de sus puntos. Un vector es normal a una superficie en un punto si es perpendicular al plano tangente en dicho punto de la superficie. Esa propiedad nos dice que un vector normal es perpendicular a cualquier otro vector contenido en el plano tangente.

Si tomamos dos vectores diferentes y tangentes a la superficie en un punto su producto vectorial será perpendicular a ambos y por tanto perpendicular a cualquier combinación lineal de ambos, es decir, perpendicular a todo el plano generado por estos dos vectores. Podemos aprovechar esa propiedad para calcular el vector normal simplemente como el producto vectorial de los dos vectores linealmente independientes dados por la parametrización de la superficie. Así el vector normal puede calcularse como:

$$n = \frac{\frac{\partial r(u_0,v_0)}{\partial u} \times \frac{\partial r(u_0,v_0)}{\partial v}}{\left\| \frac{\partial r(u_0,v_0)}{\partial u} \times \frac{\partial r(u_0,v_0)}{\partial v} \right\|} = \frac{r'_u(u_0,v_0) \times r'_v(u_0,v_0)}{\|r'_u(u_0,v_0) \times r'_v(u_0,v_0)\|} \tag{4.5}$$

Si se conoce en cambio la ecuación de la superficie $f(x,y,z) = 0$ entonces el vector unitario normal se calcula simplemente como:

$$n = \frac{\bigtriangledown f}{\|\bigtriangledown f\|} = \frac{(f'_x, f'_y, f'_z)}{\sqrt{f'^2_x + f'^2_y + f'^2_z}} \tag{4.6}$$

Para calcular las normales a partir de una nube de puntos, como es nuestro caso, primero tenemos que identificar los puntos vecinos de nuestro punto P en el que queremos hallar el vector normal. Queda establecido según geometría diferencial que la forma

geométrica local de un punto se puede aproximar a su superficie cuádrica.

Por ese motivo algunos investigadores aproximan los puntos vecinos seleccionados a la superficie cuádrica para estimar el vector normal. Esta superficie se puede expresar como:

$$Ax^2 + By^2 + Cz^2 + D + Exy + Fxz + Hyz + Jy + Kz = 0 \tag{4.7}$$

Entonces el vector normal es:

$$n = \frac{(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1)}{\left|(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1)\right|} \tag{4.8}$$

donde

$$\frac{\partial z}{\partial x} = -\frac{2Ax + Ey + Fz + G}{2Cz + Fx + Hy + K} \tag{4.9}$$

$$\frac{\partial z}{\partial y} = -\frac{2By + Ex + Hz + J}{2Cz + Fx + Hy + K} \tag{4.10}$$

Otros investigadores simplemente ajustan los puntos vecinos a un plano y usan la normal ajustada del plano como estimación del vector normal real. La normal también es estimada haciendo mínima la varianza de los productos de los puntos entre el propio vector normal y los vectores de P a sus vecinos más cercanos.

Otro tipo de método para calcular el vector normal en P es mediante el cálculo de la media ponderada de los vectores normales de la malla local. Para una malla local compuesta por los K triangulos, el vector normal es:

$$\mathbf{n} = \frac{\sum_{(i=1)}^{K} w_i \mathbf{n}_i}{\sum_{(i=1)}^{K} w_i} \tag{4.11}$$

donde $\mathbf{n}_i$ y w_i son el vector normal y el peso de la red triangular i de forma $P_i P P_{i+1}$, respectivamente. Para obtener w_i debemos aplicar lo siguiente:

$$w_i = \frac{\left| \overrightarrow{PP_i} \times \overrightarrow{PP_{i+1}} \right|}{2} \tag{4.12}$$

4.5. Segmentación

Probablemente esta sea la parte más importante de nuestro proyecto, pues es la que hace la gran distinción entre los objetos que tenemos para su reconocimiento. Tendremos que segmentar tres tipos de superficies, cilíndrica, esférica y plana.

La superficie plana nos servirá para comprobar si en la escena que está captando nuestro dispositivo se encuentra la superficie de apoyo sobre la que descansan nuestros objetos. Si tuvieramos esa superficie habría que quitarla para que no influya en los cálculos posteriores de reconocimiento.

También será interesante para conseguir identificar uno de los cinco objetos que nos hemos marcado. En nuestro caso obtenemos la ecuación del plano a partir de la información suministrada por los vectores normales. La ecuación de un plano a partir de la normal en un punto perteneciente a dicho plano se puede calcular de la siguiente manera:

Supongamos que tenemos un punto $P_0 = (x_0, y_0, z_0)$ que pertenece al plano π y sea $\vec{n} = (a, b, c)$ un vector normal a π. Entonces, para cualquier punto $P = (x, y, z)$ del plano

π, el vector $\overrightarrow{P_0P}$ es perpendicular a $\vec{n}$ de manera que

$$\vec{n} \cdot \overrightarrow{P_0P} = 0 \tag{4.13}$$

expresión que recibe el nombre de ecuación normal del plano. A partir de la ecuación normal del plano se puede obtener muy fácilmente su ecuación general:

$$\vec{n} \cdot \overrightarrow{P_0P} = a\Delta(x - x_0) + b(y - y_0) + c(z - z_0) = 0 \rightarrow ax + by + cz + d = 0 \tag{4.14}$$

donde $d = -ax_0 - by_0 - cz_0$.

Otro de los tipos de objetos que vamos a identificar son cilíndricos. Dichos objetos son una lata de refresco común y un rotulador. En geometría se dice que un cilindro es la superficie formada por todos los puntos situados a una distancia fija de una línea recta dada, el eje del cilindro.

La ecuación del cilindro circular recto es:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{4.15}$$

También se puede tener la ecuación del cilindro en forma paramétrica:

$$x = a + r\cos(u) \tag{4.16}$$

$$y = b + r\sin(u) \tag{4.17}$$

$$z = c + v \tag{4.18}$$

Por último tenemos los objetos que son esféricos como la pelota de tenis y el balón de fútbol. La esfera es un cuerpo geométrico limitada por una superficie curva cerrada cuyos puntos equidistan de otro interior llamado centro de la esfera. La esfera, como sólido de revolución, se genera haciendo girar una superficie semicircular alrededor de su diámetro.

La ecuación cartesiana de una esfera unitaria (de radio 1) con centro en el origen es:

$$x^2 + y^2 + z^2 = 1 \tag{4.19}$$

Esta ecuación se obtiene considerando que en el punto $M(x, y, z)$ de la esfera, el vector normal $\overrightarrow{OM}$ es igual a 1. Generalizando, la esfera de radio r, de centro $\Omega(a, b, c)$ tiene como ecuación:

$$(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2 \tag{4.20}$$

También se puede escribir la ecuación de una esfera en forma paramétrica. Esta ecuación quedaría:

$$x = x_0 + r\cos\theta\sin\varphi \tag{4.21}$$

$$y = y_0 + r\sin\theta\sin\varphi \tag{4.22}$$

$$z = z_0 + r\cos\varphi \tag{4.23}$$

$$(0 \leq \theta \leq 2\pi, 0 \leq \varphi \leq \pi) \tag{4.24}$$

donde r es el radio, $(x0, y0, z0)$ son las coordenadas del centro y (θ, φ) son los parámetros angulares de la ecuación.

Capítulo 5

Herramientas usadas

5.1. Dispositivos hardware

5.1.1. Kinect

El sensor Kinect es nuestra principal herramienta del proyecto, es en la que nos basamos para llevarlo a cabo.

Este dispositivo fue creado por la compañía Microsoft para su videoconsola Xbox 360. Kinect permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, y objetos e imágenes.

El sensor Kinect estaba destinado únicamente a su utilización junto a la videoconsola. Para extraer la información tan valiosa que obtiene, hubo que recurrir a ingeniería inversa. Con este proceso se pudo empezar a utilizar el dispositivo con ordenadores para diferentes aplicaciones como es el reconocimiento de objetos.

5.1.1.1. Partes

Las partes básicas de Kinect, como podemos ver en la figura 5.1 y en la figura 5.2 son:

1. Cámara RGB.
2. Sensor de profundidad.
3. Micrófono multiarray.
4. Base motorizada.

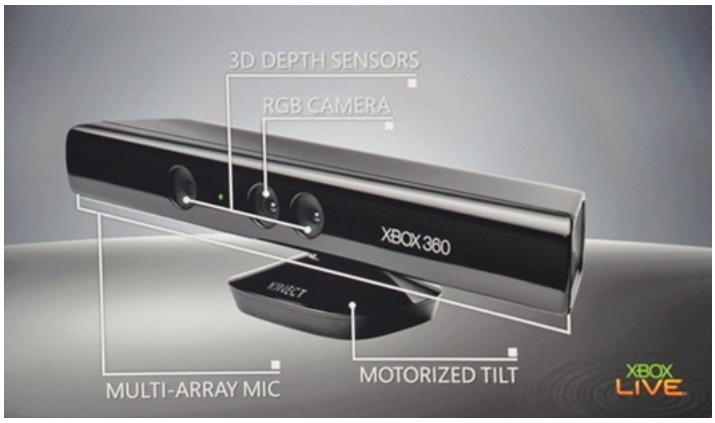

Figura 5.1: *Partes que componen la Kinect*

Figura 5.2: *Kinect desmontada en la que se ven todas sus piezas*

5.1.1.2. Funcionamiento

El sensor Kinect es capaz de capturar el mundo que lo rodea en 3D mediante la combinación de la información obtenida de la cámara RGB y del sensor de profundidad. El resultado de esta combinación es una imagen RGB-D (color + profundidad) con una resolución de 320x240, donde a cada uno de los píxeles se le asigna una información de

color y de profundidad.

Cámara RGB

El funcionamiento de la cámara RGB que monta la Kinect es como el de una cámara digital estándar. La luz atraviesa una lente que la dirige a un filtro encargado de separarla en los colores primarios, los cuales son proyectados sobre un sensor fotosensible. Este sensor genera una señal eléctrica en función de la intensidad de la señal que incide sobre él. Posteriormente, esta señal es convertida a digital mediante un ADC (Analog Digital Convert), que más tarde es analizada y reconstruida para su almacenamiento. Esto se consigue gracias a la interpolación, que permite rellenar aquellos espacios en los que falta información.

El problema del sensor es que no distingue los colores, sino variaciones de intensidad, por tanto para obtener una imagen en color es necesario descomponer la imagen en los colores primarios (rojo, verde y azul). Estos son proyectados sobre distintas zonas del sensor, el cual reconoce la cantidad de intensidad de cada uno de ellos por separado, como se puede observar en la Figura 5.3.

Estos sensores según qué tecnología empleen se clasifican en CCD (Charge Couple Device) y CMOS (Complementary Metal Oxide Semiconductor). La diferencia viene dada por la forma en la que es transmitida la información.

En el caso de los CCD, esta se envía a los extremos del sensor digital y de ahí al ADC, mientras que en el CMOS los valores se conducen directamente en formato digital, por lo que no precisa de un ADC.

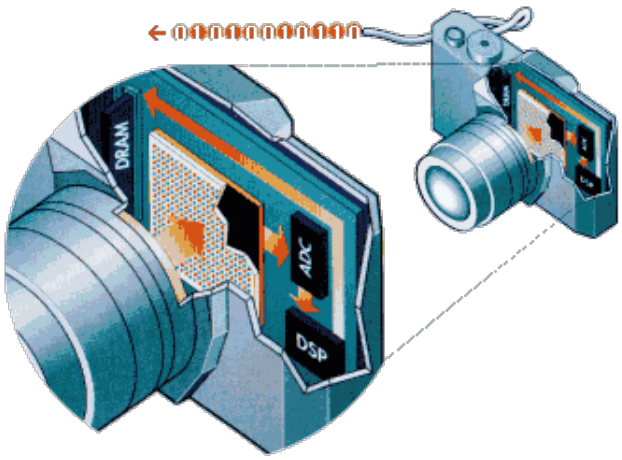

Figura 5.3: *Diagrama de funcionamiento interno de la cámara digital*

Sensor de profundidad

Por su parte, el sensor de profundidad está formado por dos componentes: un proyector de luz infrarroja (IR) y un sensor CMOS monocromo estándar. Ambos se encuentran alineados a lo largo del eje X del dispositivo, a una distancia (denominada "línea base") de 75mm, con ejes ópticos paralelos (ver figura 5.1). Esta disposición dentro de Kinect facilita los cálculos de profundidad, que se basan en un principio similiar al de triangulación activa entre emisor y cámara, esto es, entre los rayos visuales de los puntos proyectados y sus correspondientes proyecciones en la imagen.

A diferencia de los métodos tradicionales de triangulación activa, los desarrolladores de la tecnología de la cámara de rango presente en Kinect proponen una técnica ingeniosa para la obtención de información 3D, denominada Codificación de Luz (Light Coding) [11]. La idea principal consiste en un proceso en dos fases, una primera de calibración, y otra de funcionamiento.

En la fase de calibración, se emplea el proyector de luz infrarroja para proyectar un patrón de puntos (ver figura 5.4) sobre un plano de la escena, variando su distancia entre posiciones conocidas. A su vez, la cámara captura una imagen del patrón proyectado sobre el plano para cada una de estas distancias. Las imágenes obtenidas se denominan imágenes de referencia y se almacenan en el sensor.

Figura 5.4: *Ejemplo de patrón de puntos proyectado*

En la fase de funcionamiento se emplean las imágenes de referencia para sustituir "virtualmente" al emisor del patrón IR, de tal manera que para cada nueva imagen capturada por el sensor, el cálculo de profundidad se resume a un problema de visión estéreo con configuración ideal: cámaras idénticas, ejes alineados y separados una distancia base de 75 mm En cuanto al error cometido por las mediciones, de acuerdo con [**?**], éste es menor a los 10cm a distancias superiores a los 4m, y menor a los 2cm en mediciones inferiores a los 2.5m.

Micrófono multiarray

El sensor Kinect también dispone, como ya hemos mencionado, de un micrófono multiarray que permite situar la procedencia de los sonidos. Este dispositivo no será utilizado pues no tenemos que trabajar con ningún tipo de sonido.

Base motorizada

Por último disponemos de la base motorizada, que es la encargada de ajustar la inclinación en caso de que la cámara no detecte bien lo que tenga delante. En nuestro caso

tampoco será necesaria la utilización de este elemento pues el objeto siempre estará colocado en una inclinación adecuada y a una distancia correcta de modo que la cámara no tendrá que realizar ningún ajuste.

5.1.1.3. Características principales

Campo de visión

- Campo de visión horizontal: 57 grados.
- Campo de visión vertical: 43 grados.
- Rango de inclinación física: +/- 27 grados.
- Rango de profundidad del sensor: 1,2 - 3,5 metros.

Data Streams (Flujo de datos)

- Sensor profundidad: 320 x 240 a 16 bits de profundidad y 30fps.
- Cámara de color: 640 x 480 32-bit de color y 30fps.
- Micrófono: audio de 16-bit a 16 kHz.

Sistema de Seguimiento

- Rastrea hasta 6 personas.
- Rastrea 20 articulaciones por persona.
- Capacidad para mapear.

5.1.2. PC

Otra de las herramientas de hardware fundamentales es un ordenador. Este es el encargado de recibir la información captada por el sensor Kinect. También se encarga de tratar esa información para conseguir llegar a nuestro objetivo, el reconocimiento de objetos.

Debido a la gran cantidad de información con la que tiene que trabajar, se necesita un ordenador que disponga de un procesador rápido. Acelerará nuestros cálculos y nos permitirá manejar nubes de puntos mayores, es decir, que contengan más información. Cuanta más información tengamos, más precisos seremos, pero nos llevará más tiempo. Dependiendo del procesador que dispongamos, ese tiempo se verá afectado en mayor o menor medida.

Por otro lado, también es importante una tarjeta gráfica potente. Esta se ocupará de la visualización de los datos. Podremos ver la nube de puntos desde distintos ángulos, e incluso acercarnos y alejarnos. Al igual que con el procesador, cuantos más puntos tengamos, mayor será el tiempo que tardará el sistema en responder a nuestras interacciones. Por tanto, cuanto mejor sea la tarjeta gráfica, menos tiempo tardará en responder.

Figura 5.5: *Ordenador utilizado en el proyecto*

Las características principales del ordenador utilizado en este proyecto son las siguientes:

- Procesador Intel Core i7 a 2.2GHz.
- Memoria RAM de 4GB.
- Tarjeta gráfica Nvidia Geforce GT540M con 2GB dedicados.

5.2. Software

5.2.1. Linux

El entorno elegido ha sido Linux por tratarse de un sistema operativo de gran estabilidad y compatible con todas las aplicaciones necesarias para elaborar el proyecto.

En concreto se ha optado por Ubuntu (figura 5.6), una distribución GNU/Linux, cuya licencia es libre y cuenta de código abierto.

Las ventajas de este sistema operativo son:

1. Libre
2. Multitarea, multiplataforma, multiusuario y multiprocesador.
3. Protección de la memoria para hacerlo más estable frente a caídas del sistema.
4. Carga selectiva de programas según la necesidad.
5. Uso de bibliotecas enlazadas tanto estáticamente como dinámicamente.

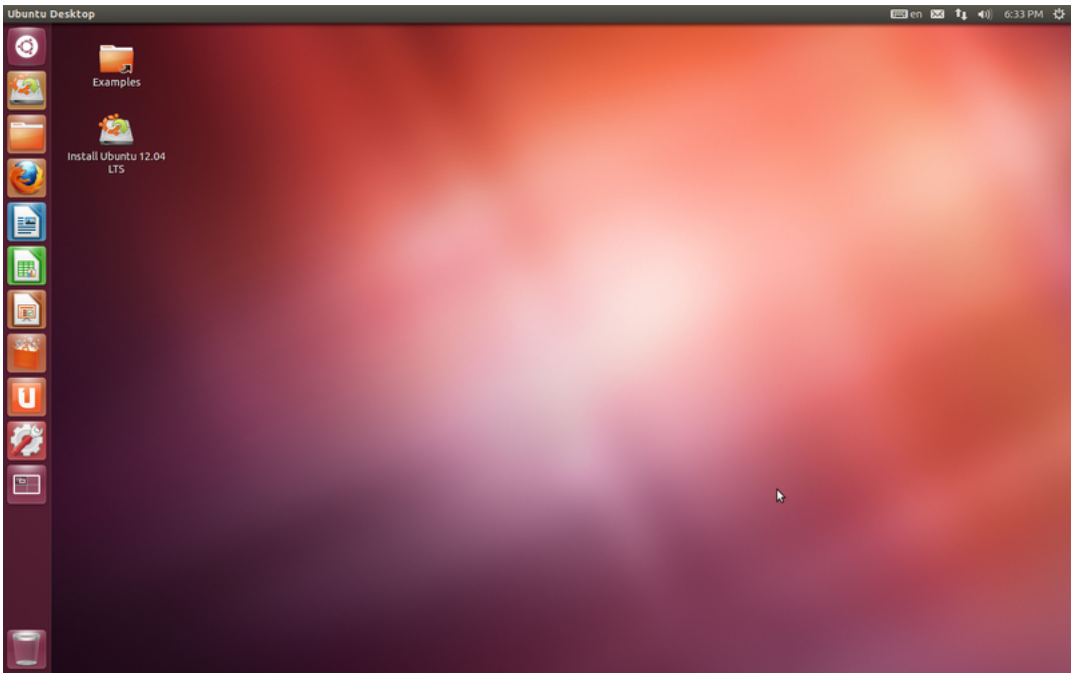

Figura 5.6: *Captura de pantalla de Ubuntu versión 12.04 LTS*

5.2.2. PCL

PCL del inglés Point Cloud Library (biblioteca nube de puntos) es un framework independiente y de código abierto que incluye numerosos algoritmos para manejar nubes de puntos en N dimensiones, y procesamiento tridimensional de las mismas.

La nube de puntos es una estructura de datos que representa una colección de puntos en varias dimensiones. Comúnmente se utiliza para representar datos tridimensionales. En una nube en 3D, cada punto tiene tres coordenadas X, Y, Z. Cuando además disponemos de información del color, la nube de puntos se convierte en 4D. Podemos ver un ejemplo en la figura 5.7.

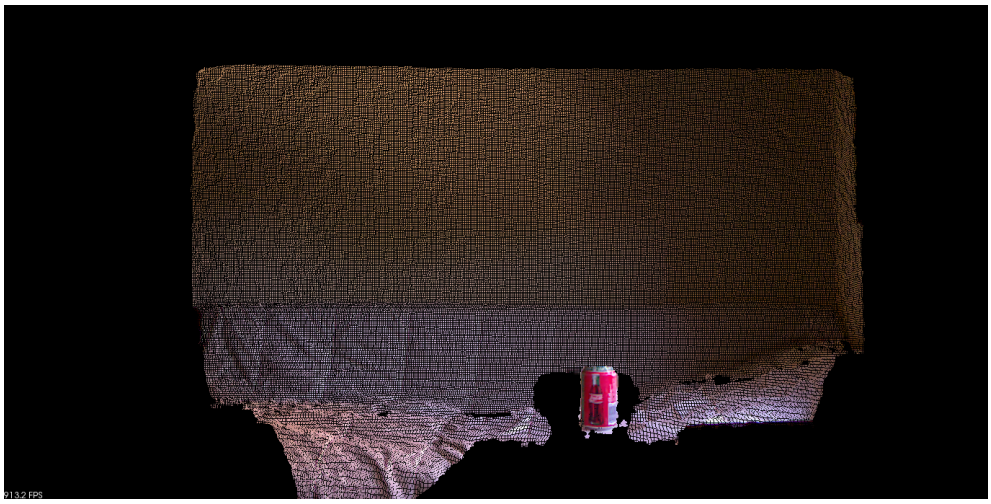

Figura 5.7: *Ejemplo de nube de puntos*

Esta nube de puntos se transmite al ordenador mediante un archivo .pcd, en el que se indica toda la información necesaria para su procesamiento. Este archivo incluye los datos de las coordenadas espaciales, así como el dato que indica el color de cada píxel. Podemos ver un ejemplo de este tipo de archivo en la figura 5.8.

```
1 # .PCD v0.7 - Point Cloud Data file format
2 VERSION 0.7
3 FIELDS x y z rgb
4 SIZE 4 4 4 4
5 TYPE F F F F
6 COUNT 1 1 1 1
7 WIDTH 640
8 HEIGHT 480
9 VIEWPOINT 0 0 0 0 1 0 0
10 POINTS 307200
11 DATA ascii
12 0.66745907 -0.7817753 1.9360001 7.9151125e-39
13 0.66733336 -0.77733338 1.9250001 8.1909763e-39
14 0.671 -0.77733338 1.9250001 8.4664856e-39
15 0.6746667 -0.77733338 1.9250001 8.2828174e-39
16 0.67833334 -0.77733338 1.9250001 8.1906133e-39
17 0.68200004 -0.77733338 1.9250001 8.2824376e-39
18 0.68566668 -0.77733338 1.9250001 8.3742801e-39
19 0.68933338 -0.77733338 1.9250001 8.0073039e-39
20 0.69300002 -0.77733338 1.9250001 7.6399661e-39
21 0.69666672 -0.77733338 1.9250001 7.6403304e-39
22 0.70033336 -0.77733338 1.9250001 7.6406822e-39
23 0.704 -0.77733338 1.9250001 7.3655274e-39
```

Figura 5.8: *Ejemplo de archivo tipo .pcd*

Las nubes de puntos se pueden obtener gracias a cierto tipo de sensores como son las cámaras estereográficas, cámaras 3D o cámaras de time-of-flight. En nuestro caso la nube

de puntos la obtenemos gracias al sensor Kinect.

La biblioteca PCL está desarrollada por un gran consorcio de ingenieros e investigadores de todo el mundo. Está escrito en C++ y es distribuida bajo una licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

5.2.3. Qt

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y ayudar al usuario a utilizar programas que no poseen una interfaz gráfica, como herramientas de la consola y servidores.

Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Utiliza el lenguaje de programación C++ de forma nativa, aunque también puede ser utilizado en otros lenguajes de programación a través de bindings.

Funciona con las principales plataformas. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y muchos de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Está distribuida bajo los términos de GNU Lesser General Public License, siendo un software libre y de código abierto.

Capítulo 6

Método de la solución diseñada

El objetivo final es la identificación de una serie de objetos, que previamente han sido captados por nuestra cámara 3D Kinect. Después haremos una pequeña interfaz para presentar los resultados por pantalla de forma clara.

Para conseguir esto hay que ir abordando uno por uno una serie de objetivos, e ir encontrando las soluciones a los distintos problemas que van surgiendo.

6.1. Consideraciones previas

Lo primero que debemos plantearnos es realizar un estudio de los objetos que vamos a intentar reconocer. Tenemos que analizar sus características y hallar un método rápido y eficaz que cumpla el objetivo deseado.

Una vez se tienen claras las características de los objetos hay que ponerse en la piel del ordenador, analizar como trabaja este y qué es lo que extrae de una captura de la cámara. Es aquí donde comienzan los problemas, ya que un ordenador no gestiona la información de la imagen de igual forma que nosotros, y por tanto tenemos que enfocarlo desde el punto de vista de la computadora.

Lo que el ordenador obtiene de cada captura es un conjunto de píxeles independientes. No los agrupa de forma intrínseca, tal y como lo hace nuestro cerebro, y este es uno de los cometidos: ayudarle a que reconozca de alguna forma los distintos elementos que componen la imagen.

La primera aproximación que intentamos para resolver el problema de la identificación de objetos era la utilización de bases de datos. Tendríamos que crear una base de datos con los objetos que quisieramos identificar, de forma que mediante la comparación de las imágenes captadas por la cámara y las que se incluyen en la base de datos, fueramos capaces de dar con la solución.

Para la creación de dicha base de datos teníamos que tomar distintas capturas del objeto en diferentes posiciones y orientaciones. Cada objeto tendría que tener muchas imágenes para que luego el ordenador pudiera reconocer el objeto de una forma rápida.

Un incoveniente de este sistema de detección es la rigidez. Sólo será capaz de detectar los objetos que nosotros le hayamos enseñado previamente en la base de datos, y en caso de querer ampliar el reconocimiento a más objetos, tendríamos que añadir nuevas bases de datos para cada uno de los nuevos objetos. Nosotros queríamos tener más flexibilidad, de forma que aunque enseñaramos un objeto de mayor tamaño, también fuera capaz de indicarnos qué tipo de objeto es así como sus dimensiones.

Otra desventaja que pudimos apreciar era el excesivo tiempo que se tardaba en completar cada una de las bases de datos que teníamos que hacer para los distintos objetos. Esto se debía a la obligación de tomar muchas imágenes con pequeñas diferencias, porque de no hacerlo, el sistema perdería precisión, cosa que queremos que tuviera nuestro algoritmo.

Debido a los motivos mencionados anteriormente decidimos descartar esta opción para llevar a cabo el objetivo de la detección de los objetos.

Posteriormente intentamos atajar el problema de los objetos como una composición

de planos. Esto significa que cada objeto está compuesto por una serie de planos. Si conseguimos encontrarlos, podemos crear el objeto a partir de esos planos para poder reconocerlo. Para llevar a cabo esta solución era necesario encontrar los puntos extremos.

Esta opción resultaba realmente complicada puesto que, como ya hemos dicho antes, el ordenador no ve la imagen como nosotros, segmentada. Por tanto para encontrar esos puntos había que realizar una serie de operaciones que resultaban ciertamente complicadas. Había que trabajar con las coordenadas de cada uno de los píxeles de la imagen, lo que daba lugar a una gran cantidad de operaciones, y también de errores, porque cuando seleccionaba algún punto que fuera incorrecto, toda la operación quedaba anulada.

Descartamos también esta alternativa porque además no trabaja correctamente con planos cilíndricos o esféricos, cosa que necesitamos en este proyecto.

Por último, la solución por la que optamos para la ejecución del proyeto también se orientaba a la localización de los planos que forman los objetos. A diferencia del método anterior, en este utilizamos la ayuda de los vectores normales a la superficie.

Como ya explicamos en el capítulo de conceptos teóricos, se calculan las normales en cada punto a partir de sus vecinos. Una vez calculadas todas las normales somos capaces de identificar que superficie forman. Gracias a esta aproximación, podremos dividir los objetos que queremos identificar en 3 tipos de superficies: cilíndrica, esférica y plana.

6.2. Captura de la imagen

La captura se hace directamente desde la Kinect. Esta se envía en tiempo real al ordenador donde será necesario tratarla con la biblioteca PCL. Esta captura se hace en 4 dimensiones, las tres espaciales (x, y, z) y la dimensión del color.

La resolución que nos permite la cámara de color para esta adquisición de datos es de

640 x 480. Esto significa que tenemos información de 307200 píxeles. Toda esta información habrá que filtrarla para conseguir reducir tiempos de computo y posibles errores.

Podemos ver la captura de la cámara antes de ser filtrada en la figura 6.1.

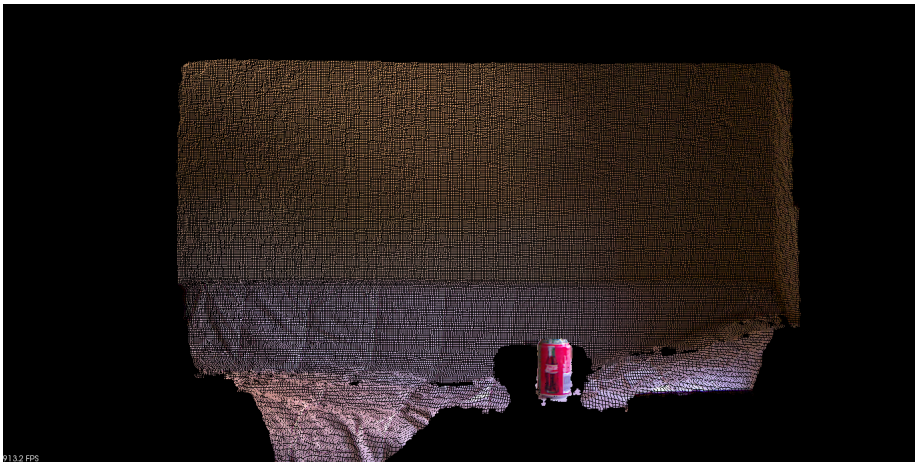

Figura 6.1: *Nube de puntos previa al filtrado*

6.3. Filtrado de la imagen

Como ya hemos dicho anteriormente, la imagen resultante de la adquisición de datos tiene gran cantidad de información y no toda ella es válida. Lo único que provoca esa información inútil es un retraso en los cálculos y un aumento de la posibilidad de incurrir en errores.

Para filtrar correctamente la imagen, deberemos llevar a cabo tres operaciones diferentes, cada una de ellas nos eliminará parte de la información innecesaria.

6.3.1. Filtrado de profundidad

Esta operación que hemos llamado filtrado de profundidad tiene que ver con la posición relativa del objeto que queremos identificar. El objeto deberá estar delante de la cámara y a una distancia entre medio metro (la cámara no detecta bien los puntos por

delante de esa distancia) y un metro.

Debemos quitar los puntos que estén situados por encima del metro de distancia a la cámara. Así evitamos confundir los puntos del fondo de la escena con los del objeto.

Para conseguir filtrar eso puntos utilizamos un algoritmo que compara el valor de profundidad de cada punto con el límite que le hayamos impuesto. Si dicho valor excede nuestro límite, el algoritmo no incluirá ese punto en el nuevo fichero que creamos con la nube de puntos filtrada en profundidad. La imagen resultante una vez pasada por este primer filtro la podemos ver en la figura 6.2.

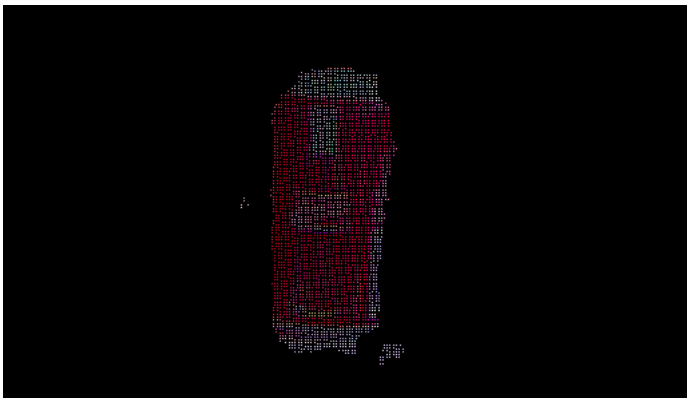

Figura 6.2: *Imagen pasada por el filtro de posición*

6.3.2. Filtrado de densidad

En este segundo paso de filtrado lo que queremos es eliminar cierta densidad de puntos para acelerar el proceso.

Utilizamos un algoritmo que divide toda la nube de puntos en una red de cubos tridimensionales. Todos los puntos que estén dentro de cada cubo se aproximan a su centroide. Esta aproximación al centroide indica que prevalecerá el segmento del cubo en el que haya mayor densidad aumentando la precisión. Dentro del algoritmo podemos seleccionar el tamaño de esos cubos para filtrar en mayor o menor medida pudiendo ver el efecto de ese parámetro de filtrado en las siguientes imágenes.

En la figura 6.3 vemos la imagen ya filtrada por posición. Sólo tenemos los puntos del objeto de donde queremos quitar cierta densidad para acelerar los cálculos.

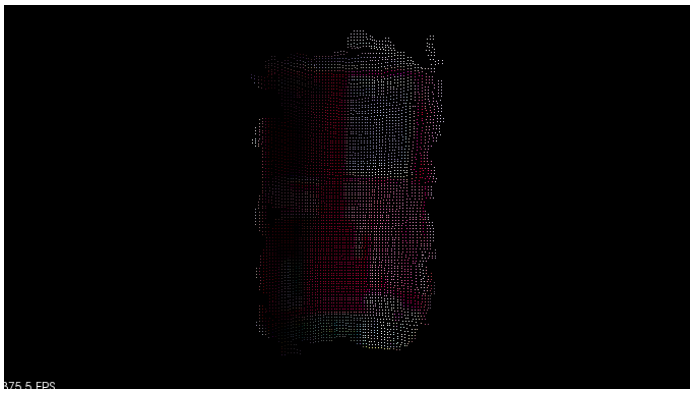

Figura 6.3: *Captura que viene del filtro de posición*

En la figura 6.4 hemos aplicado el filtro de densidad en un valor de 0.25, lo que suprime el 25 % de los datos. En la figura 6.5 podemos ver la aplicación del filtro al 0.5 y apreciándose una diferencia sustancial entre uno y otro.

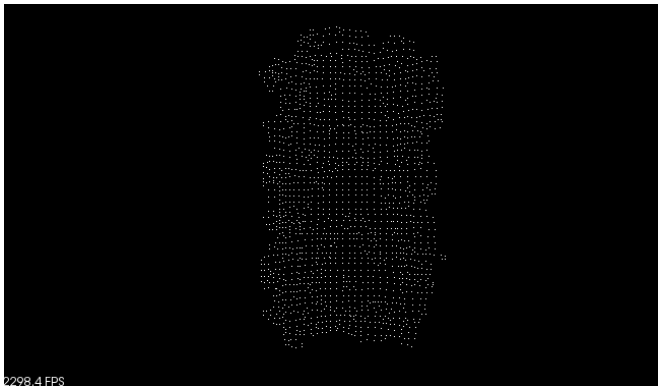

Figura 6.4: *Captura pasada por el filtro de densidad a 0.25*

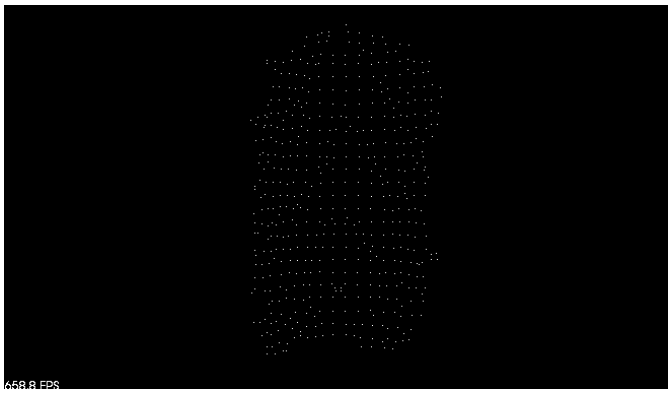

Figura 6.5: *Captura pasada por el filtro de densidad a 0.5*

6.3.3. Filtrado de la mesa

Tras la realización de pruebas para comprobar la eficacia de nuestro algoritmo de reconocimiento, detectamos que había un gran descenso del porcentaje de acierto en caso de que la cámara captara el plano sobre el que está apoyado el objeto que queremos identificar. En nuestro caso la superficie sobre la que apoyabamos los objetos era la mesa, de ahí el nombre de este filtro.

Con el filtro de profundidad, no éramos capaces de eliminar la presencia de los puntos de la mesa, y posteriormente estos datos nos impedían la correcta segmentación del objeto. Por ese motivo decidimos introducir este filtro, pues en el momento que la cámara esté situada en una posición algo elevada respecto al objeto, siempre veremos la mesa, con el consiguiente error en lo cálculos.

El algoritmo que creamos expresamente para la eliminación de la mesa requiere el uso de la segmentación de un plano, así como de la información de las coordenadas.

Lo primero que tenemos que comprobar es que exite un plano en la imagen. Este plano lo detectamos mediante las normales. Sacamos todas las normales de la imagen y las comparamos. En un plano los vectores normales son paralelos. Para comprobarlo comparamos las direcciones de dichos vectores. Si resulta que tenemos muchas direcciones paralelas, significará que hemos encontrado un plano.

El hallazgo de un plano, no significa exclusivamente que tenemos la mesa. Puede darse el caso de que esa superficie plana sea parte de la caja. Para comprobar que no se trata de ese caso, sino que efectivamente tenemos la mesa, pasamos a analizar los puntos pertenecientes a ese plano.

La idea de distinguir entre la mesa y la caja viene a consecuencia del valor de profundidad de los puntos. Aquellos que pertenezcan a la mesa, irán desde el mínimo que detecte la cámara, hasta el límite que nosotros hayamos puesto en el filtro de profundidad. Esa diferencia de profundidad será mayor en la mesa que en la caja, porque en cualquier posición que coloquemos la caja, el diferencial de profundidad nunca excederá los 20 cm, mientras que la diferencia de la mesa si.

Esos 20 cm serán nuestro indicador para descartar que se trata de la caja y por tanto tenemos la presencia de la mesa. Hay que añadir que las cajas que hemos utilizado de prueba tienen unas dimensiones máximas de 20 cm, por tanto este algoritmo lo cumplen perfectamente. En caso de querer reconocer una caja de un tamaño superior, será necesario modificar el valor asignado al algoritmo.

Una vez filtrados los puntos pertenecientes a la mesa y quedándonos con la información realmente necesaria, que es la del objeto, pasaremos a realizar los siguientes cálculos. En caso de que no fuera la mesa y se tratara de la caja, continuamos con el proceso sin eliminar ningún dato.

En las siguientes ilustraciones podemos comprobar el funcionamiento de nuestro algoritmo. En la primera de las imágenes (figura 6.6) vemos el objeto cilíndrico apoyado sobre la mesa. Aplicando el filtro los puntos de la superficie desaparecen y sólo quedan los que que guardan información relevante. Esto se puede apreciar en la figura 6.7.

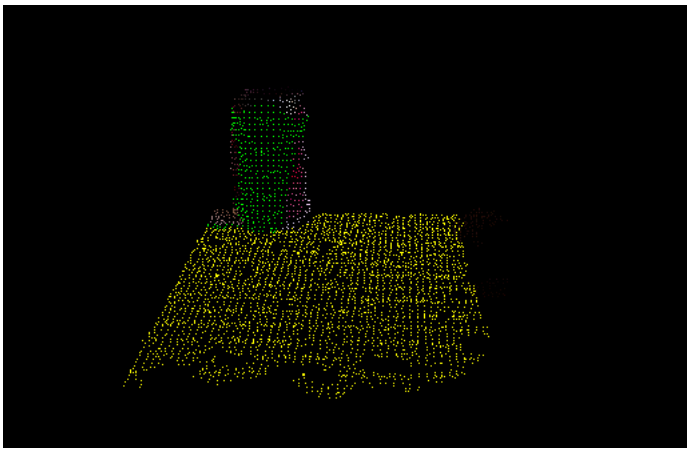

Figura 6.6: *Detalle de la lata apoyada sobre la mesa*

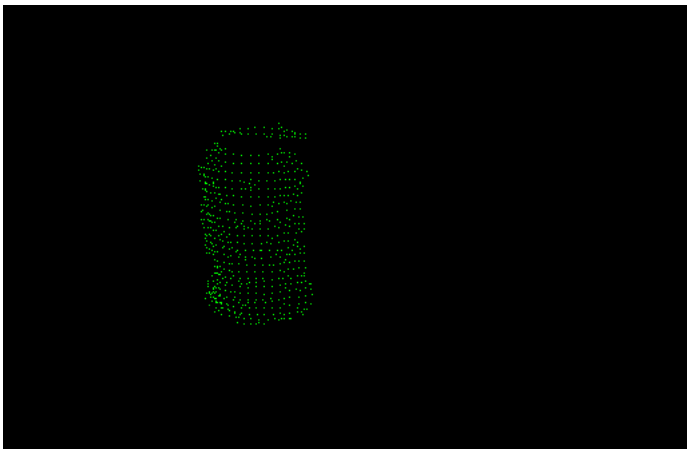

Figura 6.7: *Detalle de la lata suprimida la mesa*

6.4. Segmentación

Para identificar los objetos, primero tendremos que segmentarlos, ver de que forman son para poder clasificarlos en una u otra categoría. Para proceder al segmentado, primero será necesario calcular las normales.

Para conseguir las normales utilizamos un algoritmo que nos da en cada punto de la imagen su normal. Esta normal se calcula a partir de los puntos vecinos. En base a un radio de búsqueda que nosotros indiquemos, el algoritmo coge a los vecinos dentro de ese radio de búsqueda y crea el plano que forman. Una vez creado, obtiene la dirección

perpendicular a ese plano, es decir, el vector normal a ese plano.

La precisión del cálculo de las normales se puede variar simplemente cambiando el radio de búsqueda. Cuanto mayor sea el radio, más puntos entrarán dentro de él y por tanto mejor estará definido el plano que forman. Un aumento en la precisión de los cálculos significa un incremento exponencial en la cantidad de cálculos a realizar por el programa.

Podemos ver una imagen de la pelota de tenis antes de calcular las normales en la figura 6.8. Después podemos apreciar como se presentan las normales sobre la misma pelota de tenis en la figura 6.9.

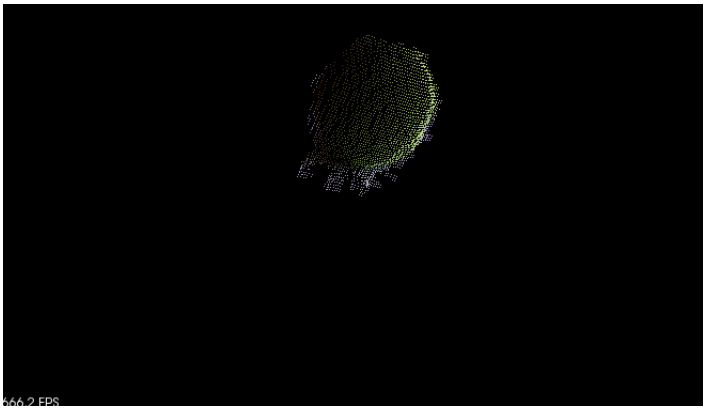

Figura 6.8: *Pelota de tenis sin normales*

Figura 6.9: *Pelota de tenis con normales*

Una vez que tenemos las normales pasamos a segmentar la imagen. Para los objetos que necesitamos identificar utilizaremos tres tipos de formas geométricas: cilíndrica, esférica y plana.

Para determinar si el objeto es cilíndrico comprobamos que las normales se cortan aproximadamente en un eje. El corte con el eje se obtiene a partir de una ecuación que calcula los puntos de corte de las normales. Si esos puntos de corte forman una recta en una misma dirección del espacio, el algoritmo determinará que esa figura es un cilindro y la recta donde se cortan todas las normales es el eje de dicho cilindro. Por lo que conoceremos más adelante el radio del cilindro.

Por otro lado, para comprobar que el objeto es esférico, lo que utilizamos, como en el caso anterior, es el corte de las normales. En el caso de una esfera, este corte se producirá en un punto, que será el centro de la esfera. La información del centro de la esfera nos será de ayuda en el análisis posterior para hallar el radio de la esfera.

Por último, entramos en la segmentación del plano. El procedimiento es el que llevamos a cabo en la sección de filtrado de la mesa. Si las direcciones de las normales son paralelas, estaremos en presencia de un plano.

6.5. Reconocimiento del objeto

Como objetivo para este proyecto nos propusimos reconocer cinco tipos de objetos diferentes. Estos cinco objetos serán: una lata, un rotulador, una pelota de tenis, un balón de fútbol y por último una caja. A estos cinco casos habría que añadir un sexto, innexistencia de objeto. Podemos ver unas capturas de los bancos de prueba con los objetos que queremos identificar en las siguientes imágenes (figuras 6.10, 6.11 y 6.12)

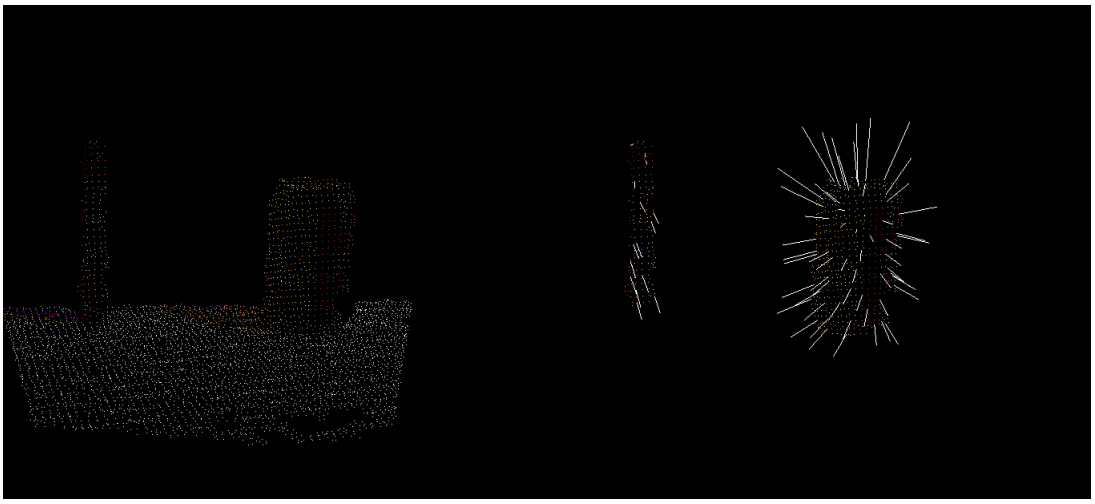

Figura 6.10: *Banco de pruebas con cilindros*

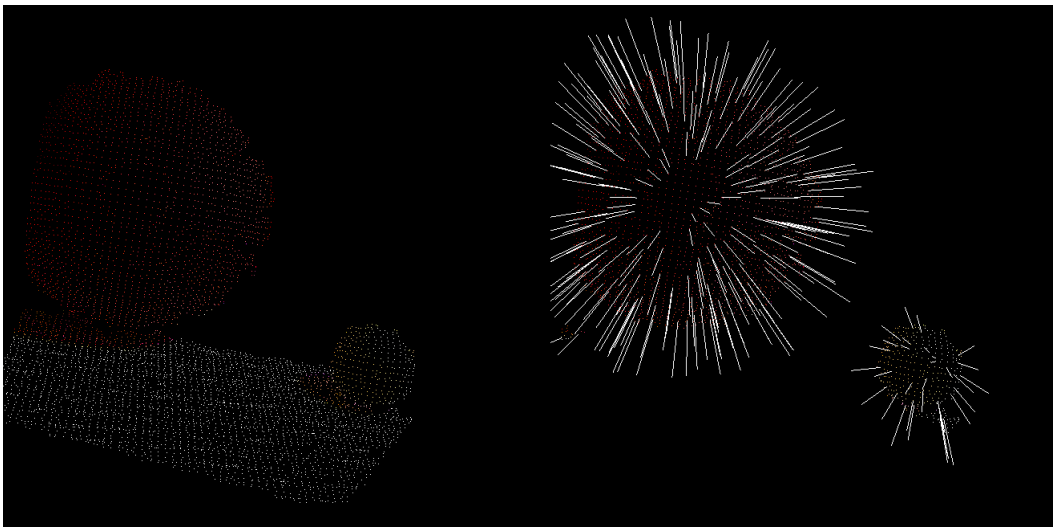

Figura 6.11: *Banco de pruebas con esferas*

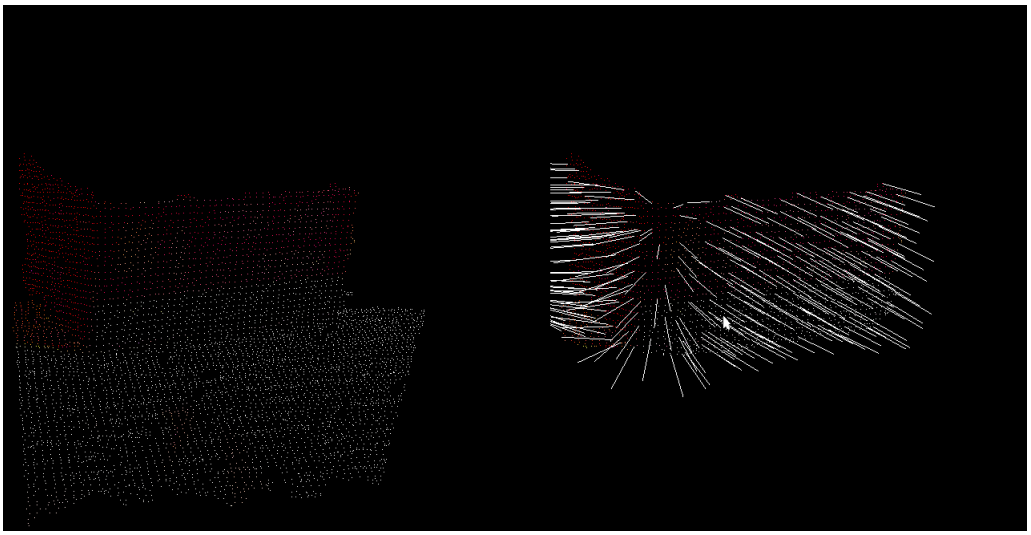

Figura 6.12: *Banco de pruebas con caja*

Hecha la segmentación del objeto y sabiendo que es un objeto cilíndrico se puede tratar de una lata o de un rotulador. Para hacer la distinción entre los mismos obtenemos el valor del radio del cilindro. El radio lo obtenemos a partir de las coordenadas del eje del cilindro. No tenemos más que hallar la distancia que existe desde el eje hasta un punto de la superficie. Para mejorar la precisión, el algoritmo toma más de una medida.

Una vez tenemos el radio, podemos determinar si el objeto es una lata o un rotulador. Si el radio está entre 2 y 4 cm determinamos que se trata de una lata. El radio habitual de una lata de refresco normal es de 3 cm. Se eligió este rango para asegurar la correcta identificación del objeto y se incluyó un porcentaje de seguridad para evitar ciertos errores. Estos fallos se detectaron tras la realización de las pruebas, ya que al tener un intervalo de búsqueda muy pequeño, el mínimo ruido provocaba que el programa no funcionara.

Si por el contrario, el radio del cilindro está entre 0,5 y 2 cm, podremos asegurar que será el rotulador. Propusimos este intervalo para el caso del rotulador porque el radio medio de un rotulador de pizarra normal es de 1 cm. Con la inclusión de este intervalo evitamos ciertos fallos gracias al coeficiente de seguridad que incorporamos.

Por otro lado, si tenemos que el objeto es esférico, se pueden dar dos casos; que sea una pelota de tenis o un balón de fútbol. Para diferenciar estos dos elementos recurrimos

nuevamente al radio. Si el radio de la esfera se encuentra entre 1 y 5 cm, podemos determinar que tenemos una pelota de tenis. Este intervalo lo añadimos tras realizar diversos estudios en los que comprobamos que el radio medio de la pelota de tenis es de 3,2 cm. Puede parecer que el intervalo es excesivo, pero elegimos este de modo que se el límite superior fuera el límite inferior del rango de búsqueda del balón de fútbol.

Si por el contrario, el radio está entre 5 y 15 cm podremos hablar de un balón de fútbol. La circunferencia de un balón de fútbol no debe exceder los 70 cm. Esto resulta en un radio de 11,14 cm. El intervalo seleccionado va desde el límite superior del la pelota de tenis hasta el valor de seguridad estimado según nuestras pruebas para reducir los posibles fallos por ruido.

Los intervalos de búsqueda de los objetos esféricos se podrían modificar para hacer una ampliación en la cantidad de objetos a reconocer. Si consideráramos incluir un objeto esférico menor que la pelota de tenis (una pelota de golf), no tendríamos más que reducir el intervalo para poder introducir el nuevo. También cabe la posibilidad de añadir un objeto de dimensiones intermedias entre la pelota de tenis y el balón de fútbol. No habría ningún problema, únicamente tendríamos que modificar los valores de los rangos.

Por último, entramos en el reconocimiento la caja. En este caso ya sabemos que disponemos de por lo menos un plano, y que no es el de la mesa porque ya lo eliminamos en sus correspondiente filtro.

Para que el algoritmo reconozca que es una caja debemos recurrir a las normales. Tendremos que encontrar dos normales que sean perpendiculares entre sí. Si las hallamos, significará que hay dos superficies perpendiculares. Podremos entonces asegurar que nos encontramos ante una caja, porque todas las caras de la caja son perpendiculares y siempre veremos al menos dos de ellas.

Para comprender el funcionamiento del algoritmo, se representa la forma en que trabaja en el diagrama de flujos de la Figura 6.13.

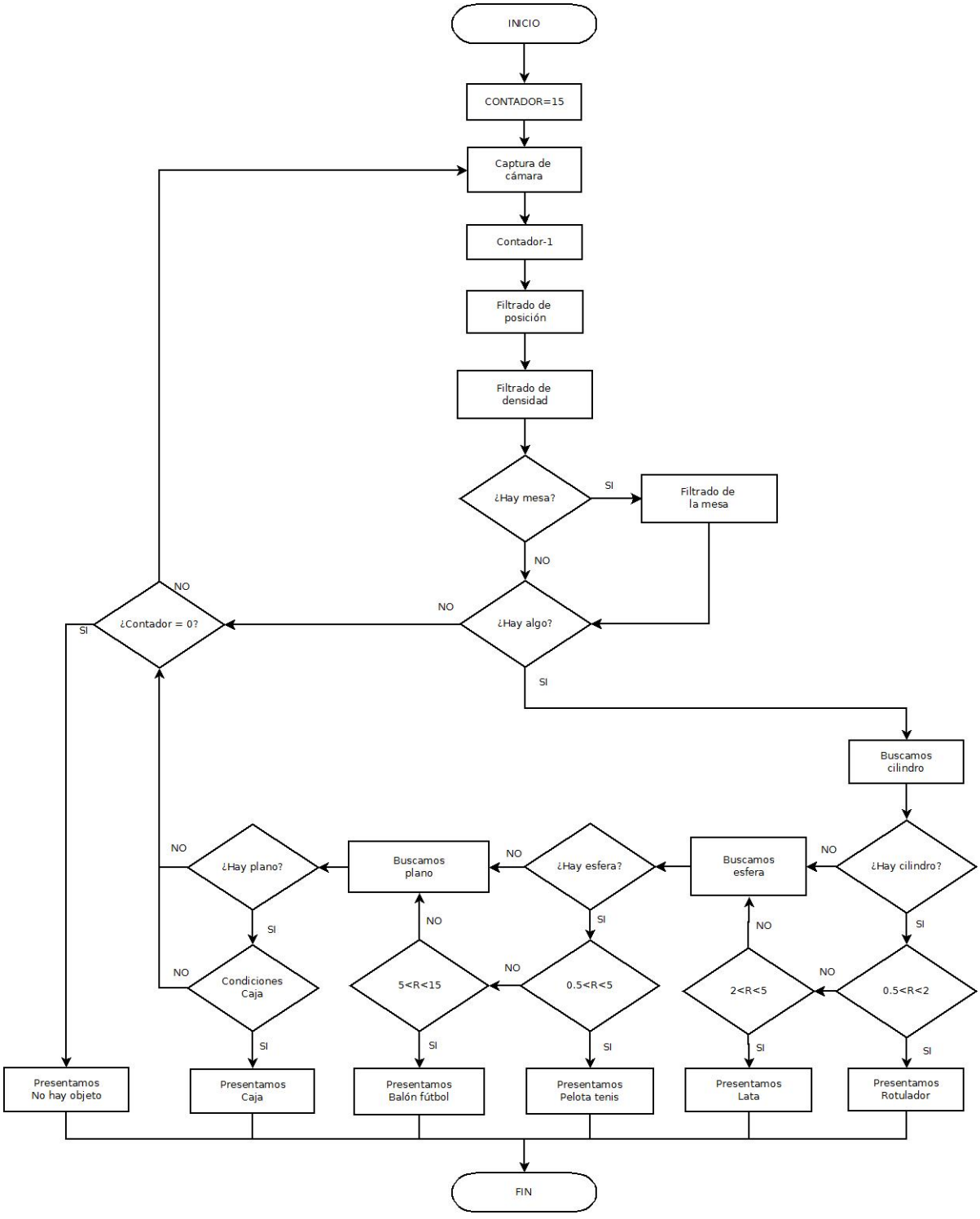

Figura 6.13: *Diagrama de flujo del funcionamiento del algoritmo*

Capítulo 7

Resultados

A continuación se procederá a exponer los resultados que se han obtenido en las diferentes fases del proyecto, así como su análisis.

7.1. Filtrado

7.1.1. Filtrado por posición

Este es el primer tipo de filtro que aplicamos a la imagen. También es el que más información desprecia pues nos quedamos con una región específica y quitamos el resto de puntos. Dependiendo del tipo de objeto que tengamos delante, esa región puede ser mayor o menor. Cuanto más grande sea el objeto, menos puntos quitaríamos y viceversa.

Se puede ver un ejemplo claro en las dos siguientes tablas de la figuras 7.1 y 7.2. Hemos separado cuando encontramos mesa y cuando no porque este filtro no es efectivo para quitar el plano de la mesa, por eso como se puede constatar en las tablas, el porcentaje de filtrado cuando no está presente la mesa es bastante mayor.

Objeto	Puntos antes	Puntos después	Porcentaje filtrado
Lata	307200	30718	90.01 %
Rotulador	307200	27961	90.90 %
Pelota de tenis	307200	29725	90.32 %
Balón de fútbol	307200	43752	85.76 %
Caja	307200	37767	87.71 %

Figura 7.1: *Resultados filtro de posición con mesa*

Objeto	Puntos antes	Puntos después	Porcentaje filtrado
Lata	307200	5011	98.37 %
Rotulador	307200	2014	99.34 %
Pelota de tenis	307200	3071	99.00 %
Balón de fútbol	307200	19458	93.67 %
Caja	307200	22971	92.52 %

Figura 7.2: *Resultados filtro de posición sin mesa*

Como podemos comprobar, este es el filtro principal, ya que quitamos, alrededor de un 90 % de los datos, en caso de que esté la mesa, y más del 92 % en el caso de que no tengamos la mesa en la escena.

La diferencia del 2 % que existe entre la presencia y la ausencia de la mesa se debe a que este filtro es incapaz de remover aquellos puntos pertenecientes al plano de la mesa. Es por ello que se aplica más adelante un filtro específico orientado a ello.

Aunque pueda parecer que se pierde demasiada información, esto no es así, porque tras diversas pruebas se pudo confirmar que todos los puntos que suprimíamos eran innecesarios, ya que no pertenecen al objeto que queremos identificar.

Durante la realización de dichas pruebas, comprobamos que cuanto menos se filtra, más errores surgen. Estos problemas surgían de la posición de algunos de los puntos que quedaban sin filtrar. Dichos puntos no pertenecen al objeto, pero el ordenador no es capaz de identificarlos. Por ese motivo los incluía en la segmentación e intentaba localizar

un objeto con esas características. La respuesta del algoritmo era confundir el objeto con otro mayor, o bien, no encontrar ningún objeto porque excedía los límites de la segmentación.

7.1.2. Filtrado de densidad

El filtrado de densidad acelera los cálculos. Frente a la rapidez tenemos la eficacia en cuanto al reconocimiento de objetos. Cuanto más filtremos más rápidos serán los cálculos posteriores, pero sacrificamos cierta efectividad, sobre todo en el caso del rotulador.

Podemos ver en las siguientes tablas la relación entre tiempo de cálculo y eficacia en el caso de aplicar el filtro de densidad o no. Todos los tiempos se expresan en segundos.

Objeto	Tiempo con filtro	Tiempo sin filtro	Eficacia con filtro	Eficacia sin filtro
Lata	4.1	20.4	88.57 %	85.65 %
Rotulador	8.2	6.2	67.06 %	83.47 %
Pelota de tenis	5.4	14.4	87.14 %	61.23 %
Balón de fútbol	5.9	112	95.38 %	96.25 %
Caja	8.4	95	85.00 %	84.36 %

Figura 7.3: *Comparación del efecto del filtro en el tiempo y la eficacia sin mesa*

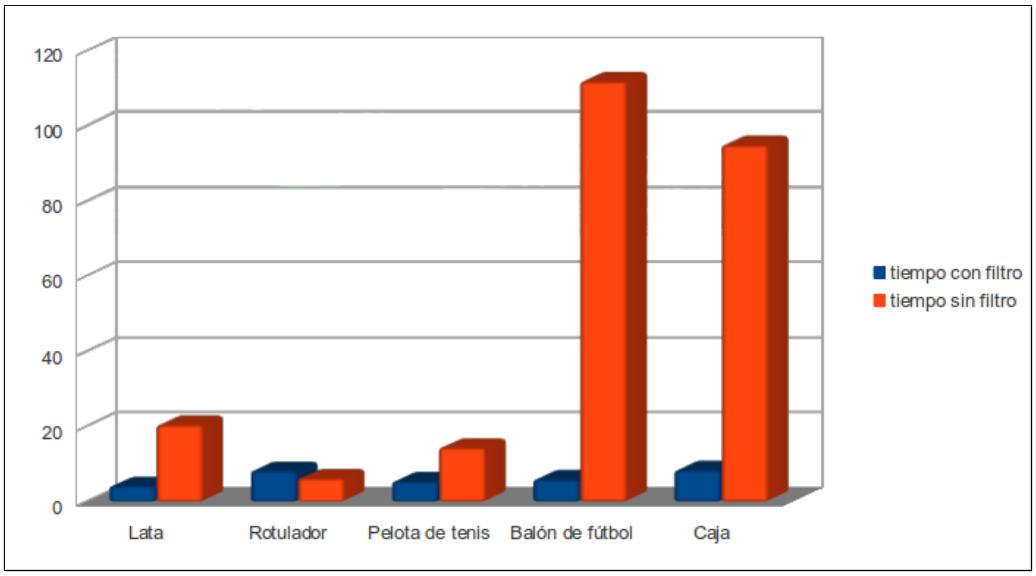

Figura 7.4: *Comparación del tiempo con filtro y sin filtro sin mesa*

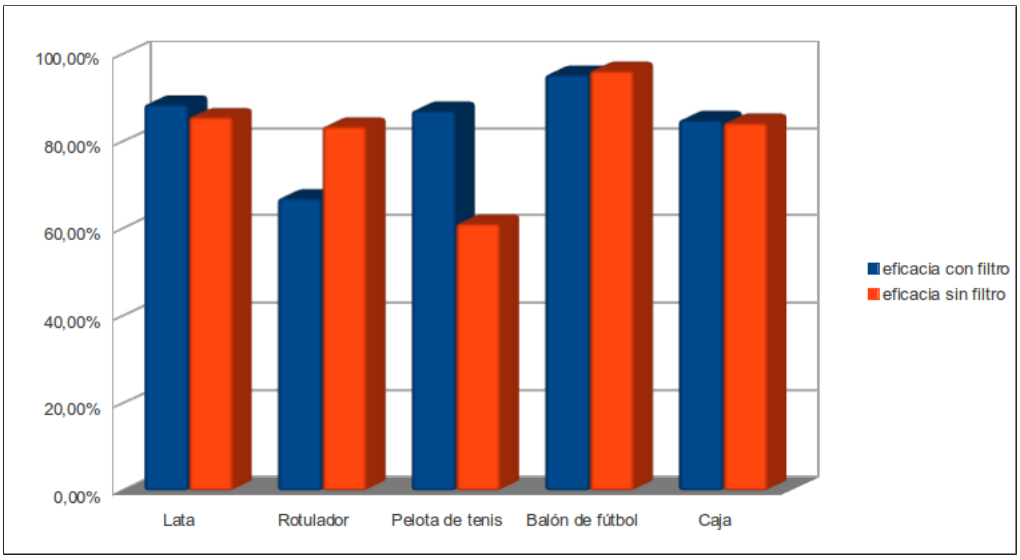

Figura 7.5: *Comparación de la eficacia con filtro y sin filtro sin mesa*

En este primer caso de estudio se ha considerado que la mesa no estaba presente en la escena (figura 7.3). Las conclusiones que extraemos del análisis son claras. En cuanto al tiempo apreciamos un aumento en todos los objetos menos en el caso del rotulador. Ese incremento de tiempo es mayor en los objetos más grandes porque al tener más puntos, los cálculos se ralentizan.

Sin embargo el tiempo que tardamos en localizar el rotulador es menor. Esto se debe a que el programa no tiene que buscar el rotulador tantas veces como antes, lo encuentra más rápido porque la segmentación que hace es más precisa. Además el incremento en la nube de puntos al no aplicar el filtro no influye en gran medida a la velocidad de los cálculos.

Por otro lado podemos estudiar también el efecto que tiene el filtrado en la eficacia en el reconocimiento del objeto. En este primer caso en el que no se incluye la mesa, podemos ver que el rendimiento es similar en casi todos los casos menos en dos, el rotulador y la pelota de tenis.

En el caso del rotulador la eficacia aumenta pues tenemos mayor cantidad de puntos para realizar la segmentación, con lo cuál conseguimos mayor precisión.

En cuanto a la pelota de tenis el rendimiento disminuye significativamente. La principal causa de fallo es la confusión con otro objeto, es decir, que el programa nos indica que tenemos una lata cuando lo que en realidad tenemos es una pelota de tenis.

En la siguiente tabla podemos ver el efecto del filtro cuando entra en escena la mesa.

Objeto	Tiempo con filtro	Tiempo sin filtro	Eficacia con filtro	Eficacia sin filtro
Lata	4.7	54.6	88.16 %	81.43 %
Rotulador	5.8	38.5	57.89 %	80.95 %
Pelota de tenis	6.4	45.1	85.19 %	68.25 %
Balón de fútbol	6.9	185	95.71 %	92.67 %
Caja	11.4	172	84.21 %	82.87 %

Figura 7.6: *Comparación del efecto del filtro en el tiempo y la eficacia con mesa*

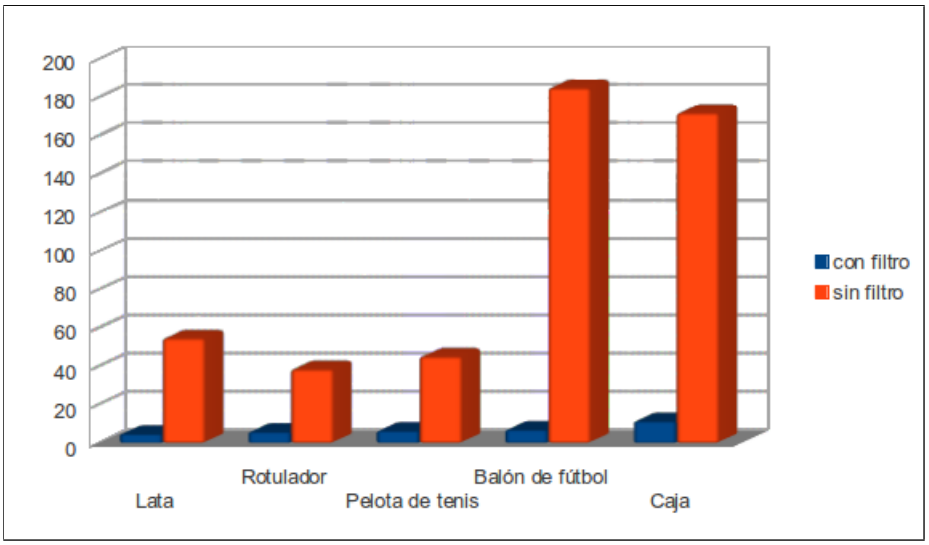

Figura 7.7: *Comparación del tiempo con filtro y sin filtro con mesa*

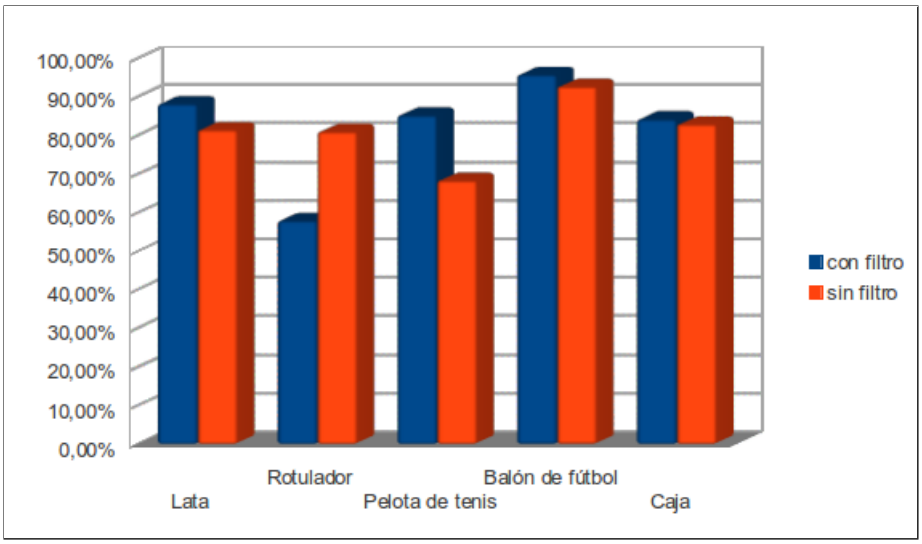

Figura 7.8: *Comparación de la eficacia con filtro y sin filtro con mesa*

Este segundo caso de estudio en el que hemos incorporado la mesa (figura 7.6) podemos ver que los resultados son similares.

En cuanto al tiempo podemos apreciar un aumento más que significativo en todos los casos. Esto se debe a la cantidad de puntos que tenemos de la mesa y que a pesar de filtrarlos, tenemos que realizar ciertas operaciones con ellos, con lo que retrasa el resto

del programa.

Por otro lado, en cuanto a la eficacia en la identificación de los objetos, vemos que los datos son similares a la situación en la que no tenemos la mesa. Se mantienen todos los casos menos el del rotulador, que aumenta considerablemente, y el de la pelota de tenis, que ve reducido su porcentaje de éxito.

Las causas de estas variaciones en la efectividad tienen que ver con el aumento de puntos en la nube para hacer los cálculos de segmentación. Estos cálculos resultan más precisos para el rotulador, pero reducen la precisión de la pelota de tenis.

A la vista de los resultados de la aplicación del filtro de densidad, podemos asegurar que es necesario su uso. Apreciamos una gran reducción en los tiempos de cálculo, sobre todo en los objetos de mayores dimensiones. Estos tiempos de cálculos los reducimos un 82,07 % de media en el caso que no haya mesa y un 90,35 % en el caso de que tengamos la mesa en la escena.

Además el aumento en la eficacia para reconocer el rotulador, se ve compensado con la pérdida de éxito en el caso de la pelota de tenis. Por estos motivos decidimos aplicar el filtro de densidad.

Los resultados de aplicar el filtro los podemos ver en las figuras 7.9 y 7.10. Recordamos que las nubes de puntos viene del filtro de posición.

Objeto	Puntos antes	Puntos después	Porcentaje filtrado
Lata	5011	1536	69.35 %
Rotulador	2014	645	67.97 %
Pelota de tenis	3071	891	70.99 %
Balón de fútbol	19458	6744	65.34 %
Caja	22971	6556	71.46 %

Figura 7.9: *Resultados filtro de densidad sin mesa*

Objeto	Puntos antes	Puntos después	Porcentaje filtrado
Lata	30718	9705	68.41 %
Rotulador	27961	9179	67.17 %
Pelota de tenis	29725	10136	65.90 %
Balón de fútbol	43752	13935	68.15 %
Caja	37767	11303	70.07 %

Figura 7.10: *Resultados filtro de densidad con mesa*

Podemos comprobar que el porcentaje de filtrado es muy similar en todos los casos. El motivo se debe a que este filtro no suprime más datos en función de los que ya tengamos, sino que lo realiza para todos igual.

7.1.3. Filtrado de la mesa

En esta sección que hemos llamado filtrado de la mesa nuestra intención es la de quitar los puntos que pertenecen a la mesa para que no interfieran en los cálculos posteriores de segmentación.

Mediante nuestro algoritmo de eliminación de la mesa conseguimos unos resultados de filtrado que podemos ver en la siguiente tabla (figura 7.36).

Objeto	Puntos antes	Puntos después	Porcentaje filtrado
Lata	9705	2120	78.15 %
Rotulador	9179	938	89.78 %
Pelota de tenis	10136	1082	89.33 %
Balón de fútbol	13935	7281	47.76 %
Caja	11303	7677	32.08 %

Figura 7.11: *Resultados filtrado de la mesa*

Observamos que existe una gran diferencia en los resultados entre los objetos pequeños y los mayores. Este efecto es debido precisamente al tamaño de los objetos. Cuanto más pequeño es el objeto, más puntos de la mesa obtenemos y por tanto mayor será el porcentaje que tengamos que filtrar. Por el otro lado cuanto más grande es el objeto, menos puntos de la mesa vemos, con lo cual, menor será la cantidad de puntos filtrados.

Para acabar con la sección de filtrado, resumimos en las siguientes tablas el resultado de aplicar todos los filtros. Hemos dividido en dos tablas los dos casos que nos podemos encontrar; la presencia de la mesa (figura 7.12) y la ausencia de ella (figura 7.13).

Objeto	Puntos iniciales	Filtro posición	Filtro densidad	Filtro mesa	Porcentaje filtrado
Lata	307200	30718	6711	2120	99.31 %
Rotulador	307200	27961	2857	938	99.69 %
Pelota de tenis	307200	29725	3173	1082	99.65 %
Balón de fútbol	307200	43752	22858	7281	97.63 %
Caja	307200	37767	25652	7677	97.50 %

Figura 7.12: *Resultados filtro total con mesa*

Objeto	Puntos iniciales	Filtro posición	Filtro densidad	Porcentaje filtrado
Lata	307200	5011	1536	99.50 %
Rotulador	307200	2014	645	99.79 %
Pelota de tenis	307200	3071	891	99.71 %
Balón de fútbol	307200	19458	6744	97.80 %
Caja	307200	22971	6556	97.87 %

Figura 7.13: *Resultados filtro total sin mesa*

Se puede apreciar que los puntos restantes, con los que trabajaremos más adelante, son un porcentaje muy reducido pero, aunque pueda parecer que el filtrado es excesivo, con esa información conseguimos resultados óptimos con una gran velocidad.

7.2. Reconocimiento de objetos

Pasaremos ahora a analizar la efectividad que hemos conseguido para llevar a cabo el principal objetivo del proyecto, la identificación de una serie de objetos.

Dividiremos este análisis en dos, como hemos hecho antes con el filtrado de posición, según tengamos que filtrar la mesa o no.

7.2.1. Sin presencia de la superficie de apoyo

En esta primera parte de nuestro análisis veremos el caso en el que la cámara está colocada de tal forma que no aprecia la superficie sobre la que se apoyan los objetos. En esta opción los resultados son algo más favorables que en la segunda, porque ante la reducción en la cantidad de los puntos apreciados, se reduce la posibilidad de errores.

Para llevar a cabo las pruebas se han utilizado los cinco objetos mencionados colocándolos en distintas posiciones y con diferentes orientaciones de forma que comprobemos la eficacia en cualquier situación.

El primer objeto a analizar es la lata. Se trata de una lata de refresco normal, de unos 3 cm de radio. Con este objeto tenemos un porcentaje de éxito del 88,57 %. Es fácil para el programa analizar la imagen y encontrar la lata porque se trata de una forma geométrica sencilla de segmentar y tiene suficientes puntos para llevar a cabo los cálculos necesarios.

A continuación se muestran algunas imágenes de la lata en distintas posiciones para comprobar que el programa la localiza. Se ha coloreado de verde para indicar que se ha encontrado el objeto. En la figura 7.14 podemos ver a la lata en posición vertical, mientras que en la figura 7.15 el envase se encuentra en posición horizontal.

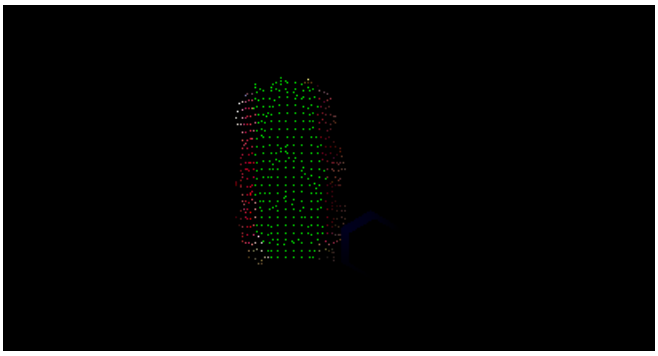

Figura 7.14: *Captura de la lata en posición vertical*

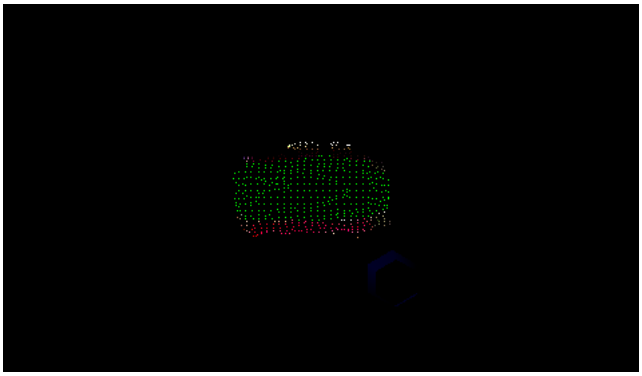

Figura 7.15: *Captura de la lata en posición horizontal*

El siguiente objeto en nuestro análisis es el rotulador. En esta situación la complicación es mayor a la hora de hallar el objeto al tratarse de un cuerpo que tiene muchos menos puntos que la lata.

Al igual que la lata, se ha tratado al rotulador como un cilindro. El problema es que el radio de este cilindro es muy pequeño y de hecho en ocasiones el programa no lo encuentra y nos indica que no tenemos ningún objeto delante.

Podemos constatar la dificultad de localizar este objeto en los resultados. Se obtiene un porcentaje de éxito del 67,06 %.

Podemos ver unas capturas del rotulador cuando ha sido encontrado por el programa, figuras 7.16 y 7.17. Se ha pintado de gris el rotulador para indicar el éxito.

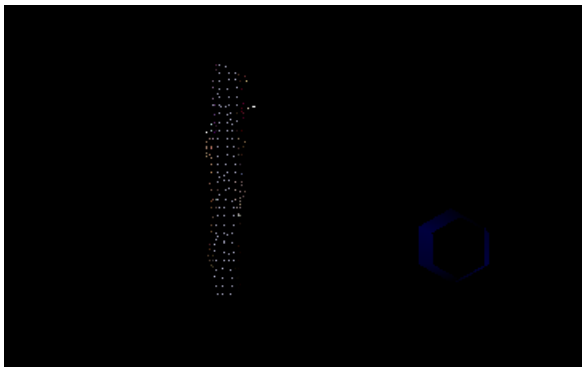

Figura 7.16: *Captura del rotulador en posición vertical*

Figura 7.17: *Captura del rotulador en posición horizontal*

Los dos siguientes objetos de nuestra lista son los esféricos. Prácticamente no encontramos ninguna dificultad, puesto que ambos objetos son fácilmente clasificables como esféricos y además tienen suficientes puntos para analizarlos.

Podríamos decir que son los dos objetos con mejor porcentaje de éxito porque la identificación de la pelota de tenis se realiza correctamente un 87,14 % de las veces, mientras que para del balón de fútbol el porcentaje de acierto es de 95,38 %.

Resulta lógico que el balón de fútbol tenga menos problemas debido a su gran volumen. Podemos añadir que los objetos esféricos tienen exactamente la misma forma independientemente del punto de vista del observador. Por ese motivo la identificación de los mismos resulta más sencilla.

A continuación se muestra unas capturas que demuestran estos resultados. Podemos ver la pelota de tenis teñida de rojo en la figura 7.18. Por otro lado tenemos el balón de

fútbol pintado de amarillo en la figura 7.19.

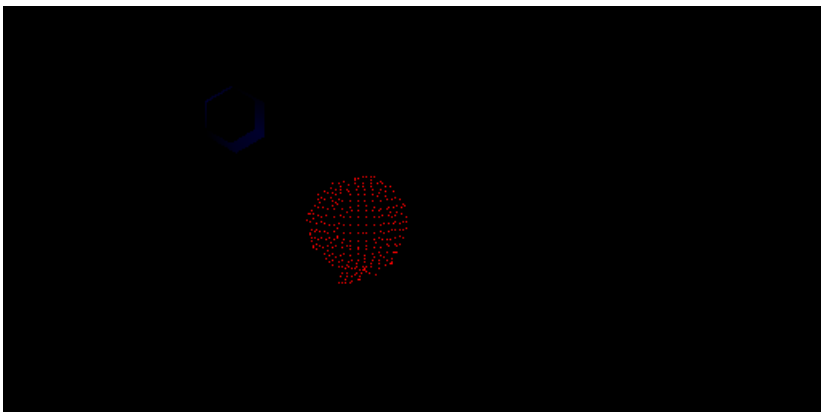

Figura 7.18: *Captura de la pelota de tenis*

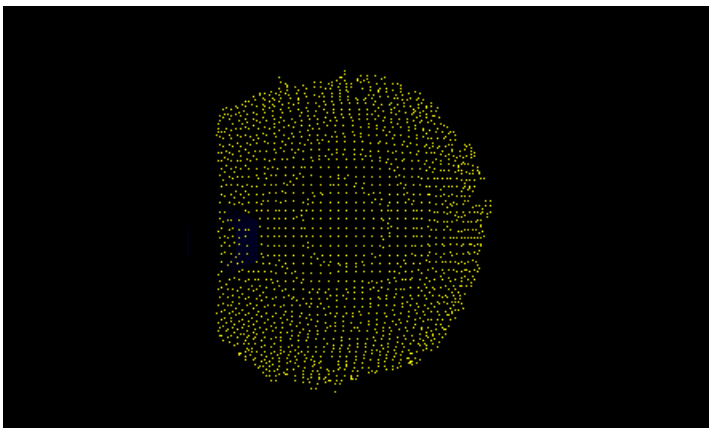

Figura 7.19: *Captura del balón de fútbol*

Como hemos mencionado anteriormente los objetos esféricos son iguales desde cualquier punto de vista, por eso hemos incluído una imágenes en las que se demuestra que el programa también identifica los objetos a pesar de captar una imagen parcial de los mismos (figuras 7.20 y 7.21).

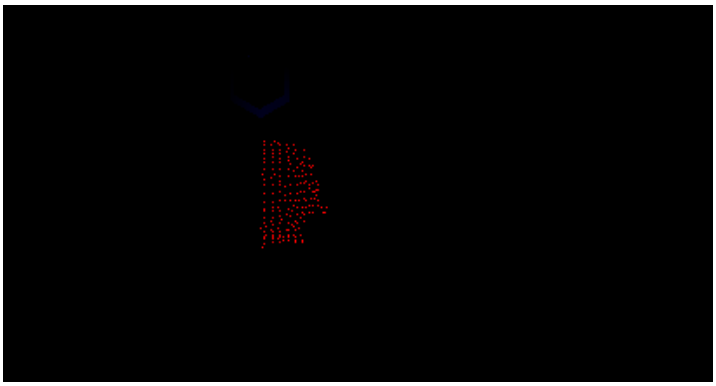

Figura 7.20: *Captura de la pelota de tenis incompleta*

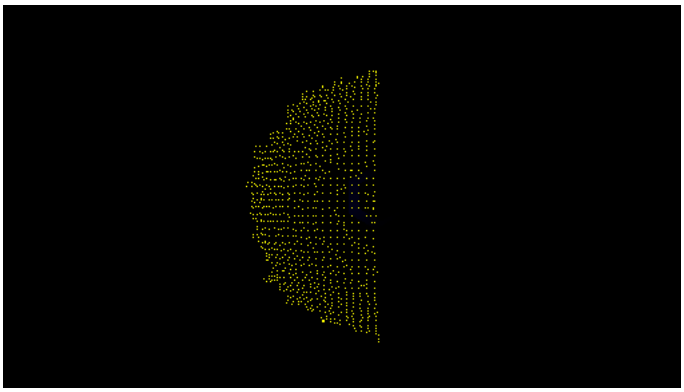

Figura 7.21: *Captura del balón de fútbol incompleto*

Por último dentro de esta sección encontramos a la caja. Se trata de un elemento formado por varios planos. Al no tener en la imagen ningún otro plano que pueda interferir en nuestros cálculos, la caja se identifica de manera sencilla. Por ese motivo tenemos que el éxito es del 85 %.

En las figuras 7.22, 7.23 y 7.24 se puede apreciar la identificación de la caja. Para facilitar la visualización se colorean los puntos que le pertenecen de morado.

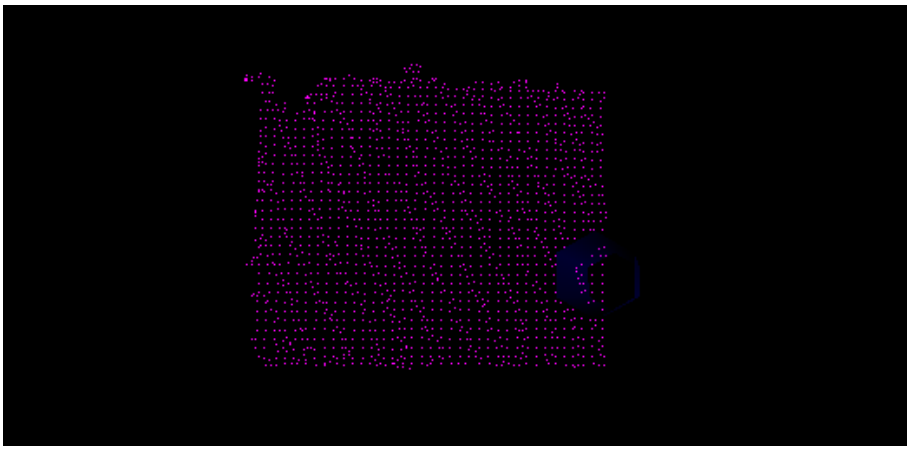

Figura 7.22: *Captura de la caja frontal*

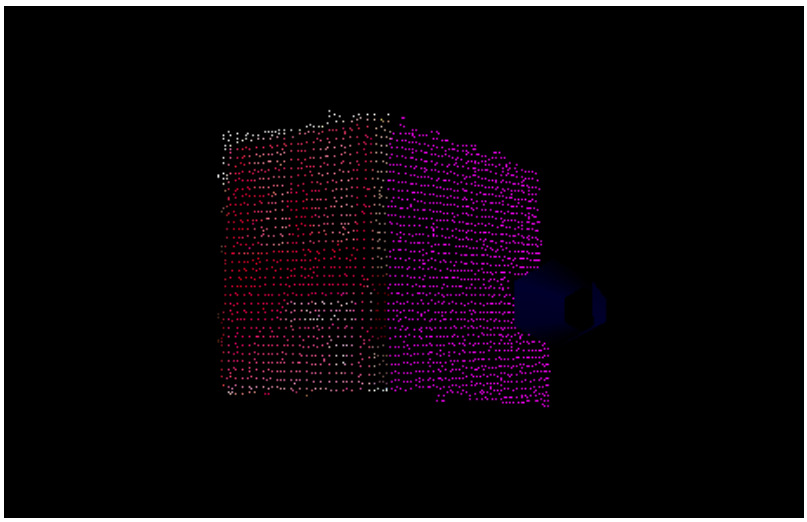

Figura 7.23: *Captura de la caja desde ángulo de 45º*

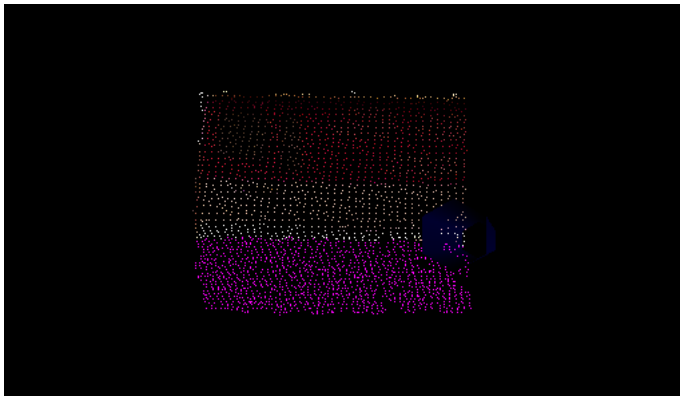

Figura 7.24: *Captura de la caja desde otra perspectiva*

Para resumir los datos de acierto y error, se adjunta una tabla (figura 7.25) con los siguientes resultados. Se puede apreciar el porcentaje de éxito, así como el de los errores.

Los fallos los hemos dividido en dos. El primero de ellos (tipo 1) se da cuando, a pesar de tener un objeto delante, el programa no lo reconoce y nos indica que no tenemos ningún objeto. Por otro lado, el segundo fallo (tipo 2) se refiere a la equivocación entre distintos objetos.

Objeto	Casos totales	Aciertos	Fallos tipo 1	Fallos tipo 2	Porcentaje éxito	Porcentaje tipo 1	Porcentaje tipo 2
Lata	70	62	3	5	88.57 %	37.50 %	62.50 %
Rotulador	85	57	25	3	67.06 %	89.29 %	10.71 %
Pelota de tenis	70	61	1	8	87.14 %	11.11 %	88.89 %
Balón de fútbol	65	62	3	0	95.38 %	100 %	0 %
Caja	80	68	4	8	85.00 %	33.33 %	66.67 %

Figura 7.25: *Resultados reconocimiento objetos sin mesa*

7.2.2. Con presencia de superficie de apoyo

En esta nueva ocasión tenemos que filtrar la mesa en la que se apoyan los objetos antes de intentar identificarlos, porque sino lo hacemos podemos incurrir en ciertos errores. El funcionamiento del programa ahora es algo mas deficiente que antes. Esto se debe a que es posible que algún punto del plano no lo considere como tal y crea que pertenece al objeto que está encima.

A pesar de este pequeño problema, la lata de refresco funciona de manera similar. Apenas apreciamos una reducción del rendimiento significativo. En este nuevo caso su porcentaje de éxito es del 88,16 %.

Veamos ahora unas imágenes que muestran esta nueva situación. Para hacer más claras las imágenes, la lata sigue de color verde y hemos pintado la mesa de azul. Podemos

ver la lata en posición vertical en la figura 7.26, mientras que en la figura 7.27 la podemos ver en posición horizontal sobre la mesa. Por último, en la figura 7.28 la vemos en un ángulo distinto.

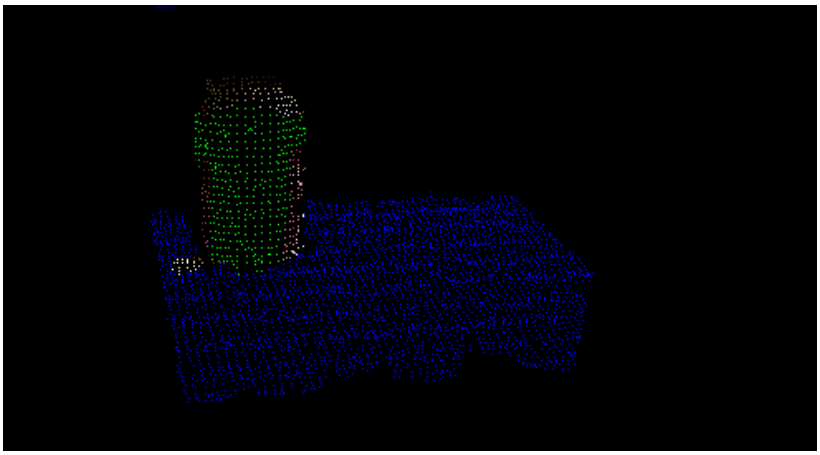

Figura 7.26: *Captura de la lata en posición vertical*

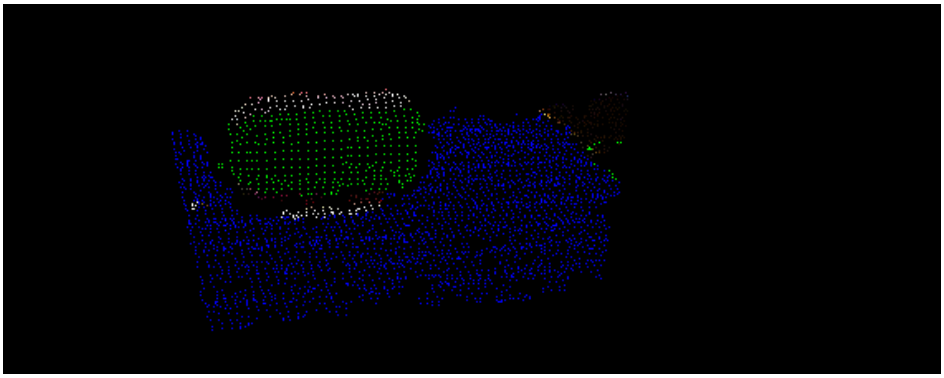

Figura 7.27: *Captura de la lata en posición horizontal*

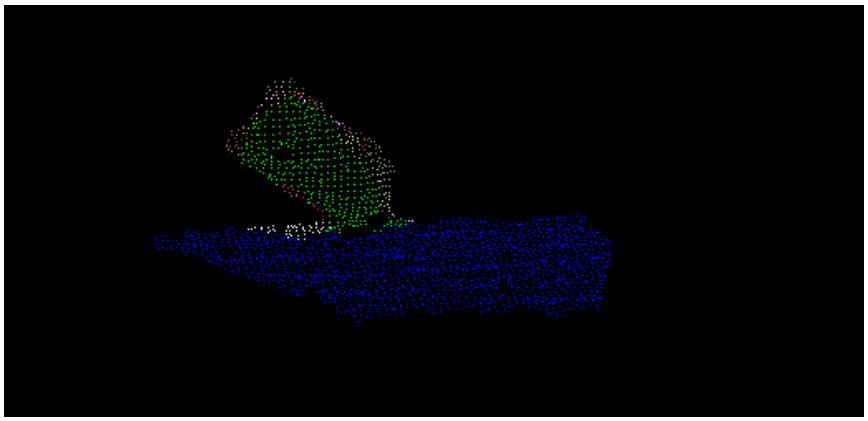

Figura 7.28: *Captura de la lata desde un ángulo diferente*

Nuestro siguiente objeto es el rotulador. Vemos que el resultado es algo peor que el anterior. El porcentaje de acierto se reduce al 57,89 %, algo no muy bueno. De todos modos habría que distinguir entre dos casos, el rotulador en vertical y cuando este está apoyado horizontalmente sobre la superficie de la mesa. Cuando se encuentra en posición vertical el reconocimiento es prácticamente igual al caso anterior, sin embargo cuando se encuentra en posición horizontal perdemos bastante precisión. Esto se debe principalmente a que el radio del rotulador es tan pequeño, que el programa piensa que son puntos que pertencen a la mesa y por tanto los elimina.

Se muestran a continuación una capturas en las que vemos la dificultad de captar el rotulador en caso de que esté horizontal (figura 7.29). Mientras tanto, en la figura 7.30 vemos que no tiene mayor dificultad de encontrar el objeto.

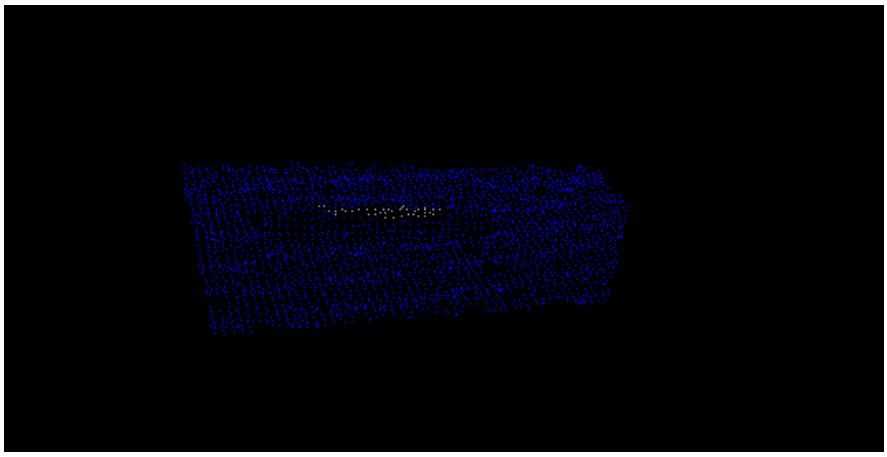

Figura 7.29: *Captura del rotulador en posición horizontal*

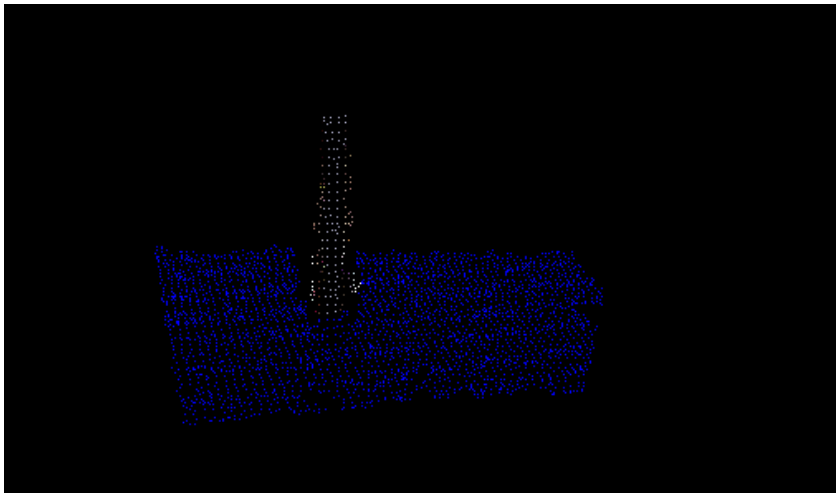

Figura 7.30: *Captura del rotulador en posición vertical*

Con los objetos esféricos sucede lo mismo que ante la ausencia de la mesa, no existen dificultades debido a que su condición esférica sólo vemos al objeto con una orientación, y a la cantidad suficiente de puntos para realizar los cálculos.

El porcentaje de éxito se mantiene. Para la pelota de tenis tenemos un 85,19 % de acierto mientras que para el balón de fútbol es del 95,71 %. En las siguientes figuras 7.31 y 7.32 vemos ambos objetos bien localizados. Al igual que hicimos anteriormente, se muestran unas imágenes en las que también se reconocen los objetos aunque estén de manera parcial en la imagen. Estas son las figuras 7.33 y 7.34.

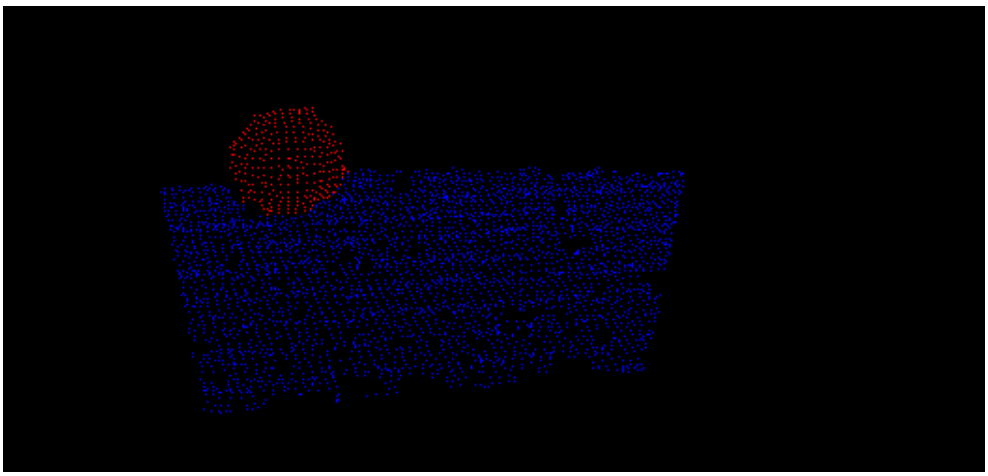

Figura 7.31: *Captura de la pelota de tenis*

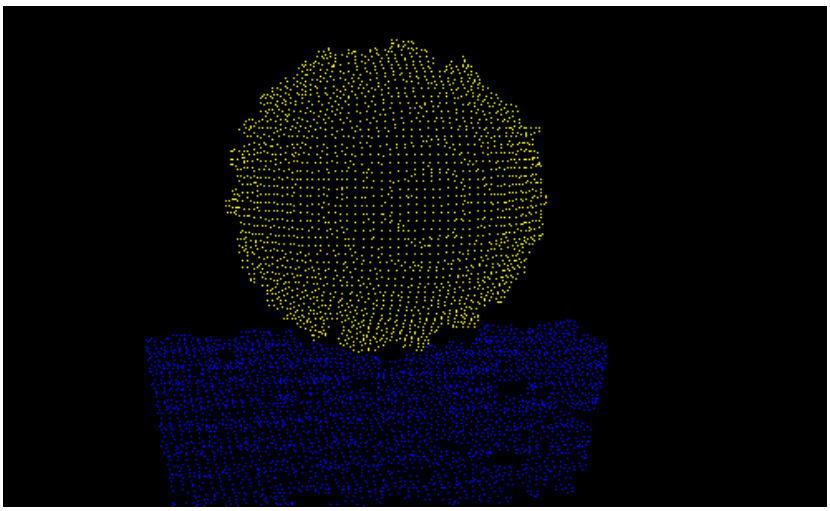

Figura 7.32: *Captura del balón de fútbol*

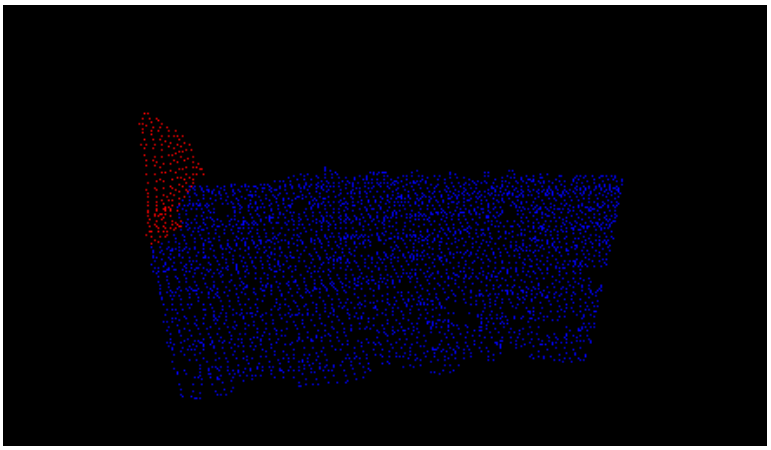

Figura 7.33: *Captura de la pelota de tenis parcial*

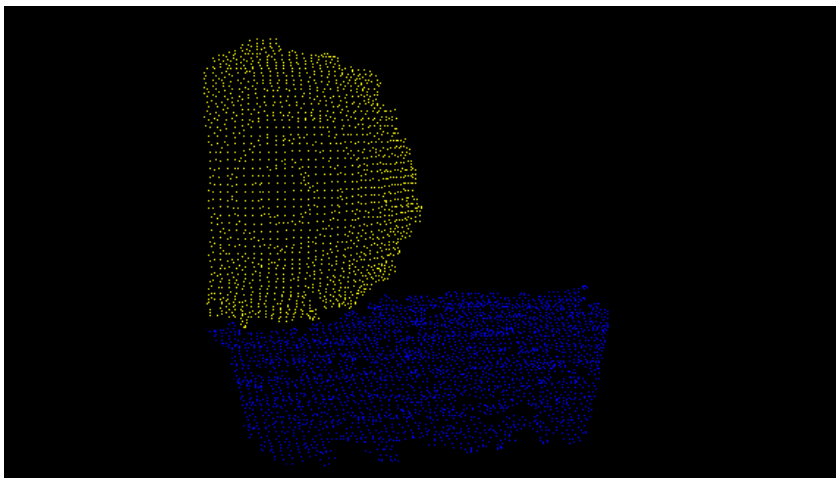

Figura 7.34: *Captura del balón de fútbol parcial*

Por último encontramos a la caja. Cabría pensar que debido al plano de la mesa, la caja se vería enormemente afectada para realizar los cálculos apropiados. Esto no es del todo acertado, pues el programa distingue perfectamente el plano de la mesa del plano de la caja mediante el algoritmo explicado en el capítulo anterior.

Debido a eso el rendimiento disminuye pero en una cifra casi inapreciable. El éxito de este caso es del 84.21 %.

A continuación podemos apreciar la distinción de los planos de mesa y caja en la figura 7.35.

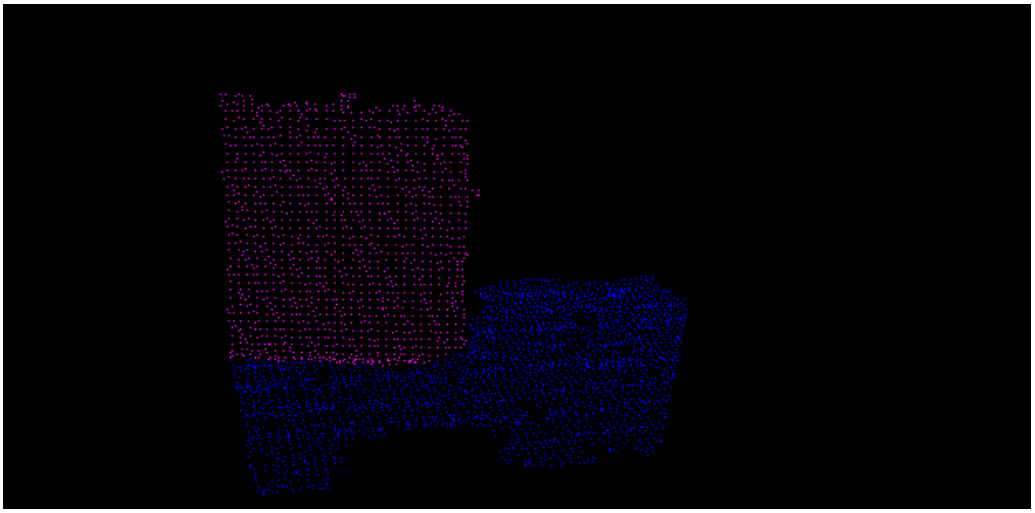

Figura 7.35: *Captura de la caja encima de la mesa*

Para apreciar de una forma más rápida y directa los resultados, se incluye una tabla con los resultados de la clasificación de los objetos en el caso de que se encuentre la mesa presente (figura 7.36). Al igual que en el caso anterior, también se incluye los fallos.

Objeto	Casos totales	Aciertos	Fallos tipo 1	Fallos tipo 2	Porcentaje éxito	Porcentaje tipo 1	Porcentaje tipo 2
Lata	76	67	3	6	88.16 %	33.33 %	66.67 %
Rotulador	95	55	36	4	57.89 %	90 %	10 %
Pelota de tenis	81	69	3	9	85.19 %	25 %	75 %
Balón de fútbol	70	67	3	0	95.71 %	100 %	0 %
Caja	95	80	6	9	84.21 %	40 %	60 %

Figura 7.36: *Resultados reconocimiento objetos con mesa*

Po último se incluye una gráfica (figura 7.37) en la que se muestra la comparativa de resultados entre el reconocimiento con la presencia de la mesa y sin su presencia.

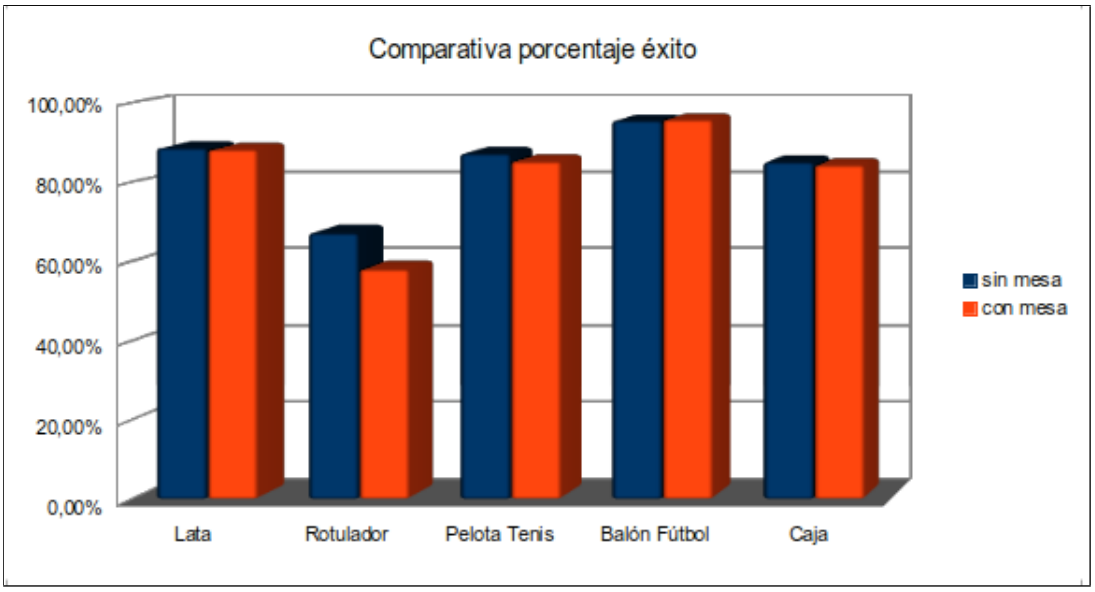

Figura 7.37: *Comparación de resultados*

Se puede apreciar en el gráfico de la figura 7.37 que los resultados en cuanto al reconocimiento no varían de manera considerable en función de la presencia o no de la mesa. El resultado más significativo es el del rotulador, que disminuye en gran parte por la mala eficacia que se obtiene cuando reposa horizontalmente sobre la mesa.

Capítulo 8

Conclusiones y futuras líneas de trabajo

8.1. Conclusiones

El objetivo principal del proyecto era la identificación y reconocimiento de una serie de objetos mediante el uso del sensor Kinect. Se puede concluir que hemos conseguido cumpir este objetivo de manera satisfactoria. Podemos apreciarlo en los porcentajes de éxito que obtenemos de intentar reconocer los objetos.

Como es natural, para completar esta meta teníamos que seguir unos pasos que han sido ejecutados de manera adecuada. En la parte de filtrado hemos conseguido aislar el objeto del resto del entorno, así como quitar cierta densidad de puntos que ralentizaban los cálculos posteriores.

Después de ello, hemos sido capaces de realizar correctamente la segmentación que nos ayuda a la hora de la identificación del objeto. Conseguimos diferenciar entre cilindro, esfera y plano.

8.2. Mejoras

Cabe la posibilidad de haber introducido algún tipo de mejoras, que podrán ser incluídas en un trabajo posterior.

La primera de estas mejoras tiene que ver con el reconocimiento del rotulador. Es el elemento que mayor problema nos ha supuesto para su identificación al tener unas dimensiones tan pequeñas. Se podría encontrar una forma alternativa de abordar el problema mediante el uso de algún otro tipo de segmentación.

Por otro lado, también se pueden incluir en futuras investigaciones el reconocimiento de más objetos a partir de la segmentación que hemos implementado en este proyecto. Reconocer esferas más pequeñas, como por ejemplo una pelota de golf o cilindros de distintos tamaños.

Además otra línea de investigación podría ser la de diferenciar objetos con las mismas dimensiones pero distintas características, como una lata de refresco y una taza. Mediante la utilización de otro tipo de algoritmos podríamos ser capaces de distinguirlas, tal vez analizando las diferencias que se dan en el color.

Por último, como futuros trabajos se podría incluir el reconocimiento de dos objetos en la misma escena y mejorar el algoritmo en rapidez y eficacia.

Apéndice A

Presupuesto

En este anexo se incluye una estimación del presupuesto de este proyecto. Se ha hecho una división por componentes necesarios y recursos humanos.

- Presupuesto de material

  En la siguiente tabla se puede apreciar el coste del material utilizado para la realización de este proyecto.

Concepto	Cantidad	Precio
Sensor Kinect	1	150 €
PC	1	650 €
Total		800 €

- Presupuesto de personal

  En este segundo apartado se calcula el importe del personal. Se distinguen las horas de investigación y documentación que fueron necesarias para la preparación del proyecto. Estas horas se cobran a un valor estimado de 10 €/hora. También se incluyen las horas de diseño e implementación del algoritmo que se cobran a 20 €/hora. Por último, las horas de pruebas que se han tenido que realizar para comprobar el correcto funcionamiento del programa.

Actividad	Nº horas	Precio/hora	Coste
Documentación e investigación	200	15	2000 €
Diseño e implementación	95	20	1900 €
Pruebas	5	10	50 €
Total	300		3950 €

- Presupuesto final

  A continuación se detalla el total del presupuesto.

Tipo de coste	Precio
Coste de material	800 €
Coste de personal	3950 €
SUBTOTAL	4750 €
I.V.A (21 %)	997.50 €
TOTAL	5747.50 €

Bibliografía

[1] G.J. Agin and T.O. Binford. Computer description of curved objects. *Computers, IEEE Transactions on*, 100(4):439–449, 1976.

[2] S. Ahn, M. Choi, J. Choi, and W.K. Chung. Data association using visual object recognition for ekf-slam in home environment. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2588–2594. IEEE, 2006.

[3] B.G. Baumgart. Geometric modeling for computer vision. Technical report, DTIC Document, 1974.

[4] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(2):167–192, 1988.

[5] P. Checchin, L. Trassoudaine, and J. Alizon. Segmentation of range images into planar regions. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 156–163. IEEE, 1997.

[6] YH Chen and CY Liu. Robust segmentation of cmm data based on nurbs. *The International Journal of Advanced Manufacturing Technology*, 13(8):530–534, 1997.

[7] M.B. Clowes. On seeing things. *Artificial intelligence*, 2(1):79–116, 1971.

[8] L. Cruz, D. Lucio, and L. Velho. Kinect and rgbd images: Challenges and applications. *SIBGRAPI Tutorial*, 2012.

[9] B. Curless. From range scans to 3d models. *ACM SIGGRAPH Computer Graphics*, 33(4):38–41, 1999.

[10] T.J. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3-d surfaces. *Robotics and Automation, IEEE Journal of*, 3(6):527–538, 1987.

[11] J. Garcia, Z. Zalevsky, et al. Range mapping using speckle decorrelation, October 7 2008. US Patent 7,433,024.

[12] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[13] D.I.B. Hagebeuker. A 3d time of flight camera for object detection. 2007.

[14] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[15] R. Hoffman and A.K. Jain. Segmentation and classification of range images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):608–620, 1987.

[16] Y.Y. Huang and M.Y. Chen. 3d object model recovery from 2d images utilizing corner detection. In *System Science and Engineering (ICSSE), 2011 International Conference on*, pages 76–81. IEEE, 2011.

[17] A. Hub, T. Hartter, and T. Ertl. Interactive tracking of movable objects for the blind on the basis of environment models and perception-oriented object recognition methods. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 111–118. ACM, 2006.

[18] D.A. Huffman. Impossible objects as nonsense sentences. *Machine intelligence*, 6(1):295–323, 1971.

[19] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[20] R. Jenkins and AM Burton. 100 % accuracy in automatic face recognition. *Science*, 319(5862):435–435, 2008.

[21] A. Kasper, Z. Xue, and R. Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012.

[22] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[23] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[24] G. Levin. Computer vision for artists and designers: pedagogic tools and techniques for novice programmers. *AI & Society*, 20(4):462–482, 2006.

[25] Alejandro Lumbier Álvarez. Detección y extracción de las características de la mano humana mediante técnicas de visión. Master's thesis, Universidad Carlos III, Madrid, 2011.

[26] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[27] MJ Milroy, C. Bradley, and GW Vickers. Segmentation of a wrap-around model using an active contour. *Computer-Aided Design*, 29(4):299–320, 1997.

[28] A. Moro, E. Mumolo, and M. Nolich. Building virtual worlds by 3d object mapping. In *Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning*, pages 31–36. ACM, 2010.

[29] C. Pheatt, J. Ballester, and D. Wilhelmi. Low-cost three-dimensional scanning using range imaging. *Journal of Computing Sciences in Colleges*, 20(4):13–19, 2005.

[30] J. Ponce. *Toward category-level object recognition*, volume 4170. Springer-Verlag New York Inc, 2006.

[31] A. Rosenfeld. Quadtrees and pyramids for pattern recognition and image processing. In *Proc. Fifth Int. Conf. on Pattern Recognition*, pages 802–807, 1980.

[32] A. Rosenfeld and A.C. Kak. Digital picture processing. volumes 1 & 2//(book). *New York, Academic Press, 1982,*, 1982.

[33] A. Saxena, M. Sun, and A.Y. Ng. 3-d reconstruction from sparse views using monocular vision. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[34] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[35] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data.

[36] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24(1):52–96, 1983.

[37] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[38] T. Varady, R.R. Martin, and J. Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997.

[39] G. Vouzounaras, J.D. Perez-Moneo Agapito, P. Daras, and M.G. Strintzis. 3d reconstruction of indoor and outdoor building scenes from a single image. In *Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning*, pages 63–66. ACM, 2010.

[40] I. Weiss and M. Ray. Model-based recognition of 3d objects from single images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):116–128, 2001.

[41] Wikipedia. 3d scanner — wikipedia, the free encyclopedia, 2012. [Online; accessed 17-August-2012].

[42] M. Yang and E. Lee. Segmentation of measured point data using a parametric quadric surface approximation. *Computer Aided Design*, 31(7):449–457, 1999.

[43] N. Yokoya and M.D. Levine. Range image segmentation based on differential geometry: A hybrid approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(6):643–649, 1989.

[44] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 13–13. Ieee, 2006.

[45] D. Zhao and X. Zhang. Range-data-based object surface segmentation via edges and critical points. *Image Processing, IEEE Transactions on*, 6(6):826–830, 1997.

[46] Hui Zheng, Jie Yuan, and Renshu Gu. A novel method for 3d reconstruction on uncalibrated images. In *Proceedings of the Third International Conference on Internet Multimedia Computing and Service*, ICIMCS '11, pages 138–141, New York, NY, USA, 2011. ACM.

[47] J. Zhu, G. Humphreys, D. Koller, S. Steuart, and R. Wang. Fast omnidirectional 3d scene acquisition with an array of stereo cameras. In *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*, pages 217–224. IEEE, 2007.