Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Proyecto Final de Carrera
Ingeniería en Informática
Curso 2011-2012

Reconocimiento de Objetos en 3D Utilizando Sensores de Visión y Profundidad de Bajo Coste

David Bueno Monge

Mayo de 2012

Directora: Ana Cristina Murillo Arnal

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Reconocimiento de Objetos en 3D Utilizando Sensores de Visión y Profundidad de Bajo Coste

RESUMEN

La visión por computador tiene en nuestros días multitud de aplicaciones: desde entornos industriales, con sistemas de control de calidad y monitorización, a entornos domésticos, como aplicaciones en videojuegos, seguridad u otras más recientes como coches autónomos.

El presente proyecto se centra en el reconocimiento de objetos en entornos domésticos, y en particular teniendo en mente sistemas robóticos de servicio en estos entornos. Este campo de investigación tiene actualmente nuevos retos gracias a la reciente aparición de sensores de visión capaces de medir la profundidad a que están los objetos de la imagen, lo que supone un gran cambio de concepto respecto a los métodos basados en imágenes convencionales. Estos sensores se conocen como RGB-depth o RGB-d.

El objetivo de este proyecto ha sido diseñar e implementar un sistema de reconocimiento de objetos basado en la información 3D que son capaces de obtener estos nuevos sensores, para así comprobar las ventajas que ofrecen frente a las técnicas más convencionales. Para ello ha sido necesario un estudio de literatura relacionada con el problema, diseñar e implementar un sistema convencional de reconocimiento y evaluarlo, para poder encontrar las principales ventajas e inconvenientes que ofrecen las técnicas 2D. Después y como parte central del proyecto, diseñar e implementar un sistema de reconocimiento que mezcla técnicas 2D y 3D y evaluarlo para comprobar cuales son las principales ventajas que ofrece la tecnología 3D.

Todo este trabajo ha implicado el estudio de diversas técnicas de manejo de la información contenida en las imágenes, como la representación de la información en nubes de puntos o el uso de descriptores VFH, ORB, SURF, e histogramas de color. Se han estudiado varios algoritmos y medidas de similitud entre dichos descriptores, como el algoritmo de búsqueda del vecino más cercano o la representación de los descriptores en vocabularios de palabras visuales. También se estudiaron procesos de segmentación de imágenes basados en información 3D y métodos de aprendizaje y reconocimiento que permitan identificar los objetos.

Se ha desarrollado un sistema de reconocimiento que arroja unos resultados satisfactorios y de bajo coste de cómputo. Este sistema evidencia las mejoras que aportan las técnicas 3D frente a las convencionales. Por último, el sistema desarrollado se compara con otro sistema de características similares, donde se observa que el sistema desarrollado alcanza un buen rendimiento respecto a métodos recientes publicados en la literatura relacionada.

Índice

1. Introducción

Este capítulo describe la motivación que ha llevado al desarrollo de este proyecto y los objetivos y el alcance del mismo. Así mismo, describe las herramientas utilizadas y el entorno de trabajo, y finalmente incluye una descripción de la organización de la memoria.

1.1. Motivación

De todas las tareas de visión artificial, analizar una escena reconociendo todos los objetos que aparecen en ella sigue siendo una de las más complejas. Mientras que la tecnología ha avanzado hasta poder reconstruir de manera precisa escenas 3D, a partir de imágenes 2D o más recientemente gracias a nuevos sensores con capacidad para medir la profundidad, todavía no es fácil aprender a detectar y reconocer todos los objetos presentes en una foto. Ni siquiera hay un consenso entre los investigadores sobre cuando se podría alcanzar el nivel de aprendizaje visual de un niño.

Es precisamente el reconocimiento de objetos el tema general de este proyecto. La visión por computador ha visto mejoras espectaculares en los últimos años y sigue siendo un área de investigación en expansión. Este trabajo también tiene cierta relación con el campo de la robótica, para permitir que un sistema inteligente se desenvuelva de manera lo más autónoma posible hay que proveerle de capacidades para identificar objetos y eventos a su alrededor en situaciones realistas, con los que tendrá que interactuar.

La incorporación de un sistema de reconocimiento de objetos fiable a un robot puede multiplicar la cantidad de tareas que dicho robot puede realizar. Considérese por ejemplo un robot doméstico que pueda reconocer los objetos comunes que puedan encontrarse en una casa, y por tanto tomar decisiones de acuerdo con la naturaleza de dichos objetos.

Recientemente, gracias a la comercialización de dispositivos de bajo coste que incorporan cámaras y sensores de profundidad, también denominados cámaras RGB-d, están surgiendo muchas oportunidades y temas de investigación nuevos. Como se puede ver en la Figura 1.1, la información que aporta una de estas cámaras ayuda en problemas irresolubles con una sola imagen. Estos dispositivos ofrecen multitud de aplicaciones y capacidad de mejora respecto a los sistemas que trabajan con cámaras sin percepción de profundidad. Concretamente, este proyecto se ha realizado con el dispositivo *kinect*[1], una cámara RGB-d desarrollada por Microsoft y de reciente comercialización. Para más información sobre el dispositivo véase el anexo A.

[1] http://www.xbox.com/es-ES/kinect

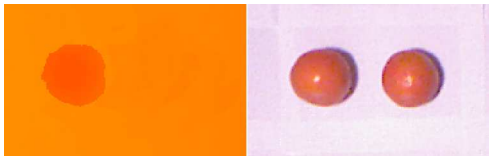

Figura 1.1: ¿Es una naranja o la foto de una naranja? A la derecha se muestra una imagen de color normal, tomada por una cámara corriente, y a la izquierda el mapa de profundidad de la misma imagen que pueden aportar las cámaras RGB-d. Estas cámaras nos permiten distinguir que la naranja de la derecha es en realidad la foto de una naranja.

1.2. Objetivos y alcance

El objetivo principal del proyecto es diseñar e implementar un sistema de reconocimiento de objetos utilizando técnicas 3D donde se demuestre el beneficio de aplicar dichas técnicas respecto a sistemas de reconocimiento convencionales. El sistema deberá aprender un modelo visual de cualquier objeto a partir de unas pocas imágenes que luego podrá ser utilizado para identificar dicho objeto en otras imágenes. La clave residirá en cómo aprovechar e integrar la información de profundidad que el sensor ofrece para lograr mejores resultados que los que se lograrían con una cámara convencional.

Las tareas a realizar para cumplir los objetivos son las siguientes.

- Estudio de documentación sobre los siguientes temas
  - Controladores y librerías del dispositivo *kinect*.
  - Librerías para trabajar con algoritmos de de procesamiento de imagen y visión por computador.
  - Artículos de investigación sobre técnicas típicas de reconocimiento de objetos.
  - Artículos de investigación sobre propuestas para reconocimiento de objetos utilizando información 3D
- Diseño e implementación de un sistema de aprendizaje de modelos y reconocimiento básico que solo utilice información 2D.
- Evaluación del sistema inicial para encontrar los puntos débiles y fuertes del reconocimiento 2D.
- Diseñar e implementar un sistema de reconocimiento que utilice técnicas 3D y aproveche las virtudes del reconocimiento 2D.
- Evaluar el sistema final y encontrar las principales mejoras que ofrece la aplicación de técnicas 3D.
- Comparar el sistema desarrollado con otros métodos de la literatura relacionada.

1.3. Herramientas y entorno de trabajo

Todo el desarrollo del proyecto se ha llevado a cabo bajo el sistema operativo Kubuntu 11.04 Natty[2] y se ha implementado utilizando el lenguaje C++.

Para el manejo del sensor *kinect* se ha utilizado el driver OpenKinect[3] para la primera versión del sistema que solo utiliza técnicas 2D, pues es muy sencillo de utilizar. Sin embargo, para el desarrollo del sistema final, se trabajó con el driver OpenNI[4] ya que es mucho más potente y ofrece mayor funcionalidad.

Durante todo el proyecto se ha hecho uso de las librerías de visión por computador OpenCV[5] para procesamiento de imágenes y técnicas de reconocimiento 2D. Para el manejo de la información 3D y el uso de técnicas de segmentación y reconocimiento 3D se ha empleado la novedosa librería PCL (Point Cloud Library)[1]. Por último, para manejar *kd-trees*[9] y realizar búsquedas sobre ellos se ha utilizado la librería FLANN (Fast Approximate Nearest Neighbours)[2].

1.4. Organización de la memoria

En el capítulo 2 se detallan los métodos estudiados y empleados para construir el sistema inicial de reconocimiento basado en técnicas 2D. También aparece la evaluación de dicho sistema y las decisiones que se tomaron a raíz de dicha evaluación. En el capítulo 3 se muestran todas las técnicas 3D estudiadas y que posteriormente se utilizaron para construir el sistema de reconocimiento final. En el capítulo 4 se detalla el funcionamiento y el rendimiento del sistema desarrollado, mientras que en el capítulo 5 se describen las conclusiones extraídas de la elaboración del proyecto.

El documento también consta de varios anexos que detallan información adicional sobre el proyecto. El Anexo A describe el sensor *kinect*. El Anexo B muestra la gestión que se ha seguido para elaborar el proyecto, junto a un diagrama de Gantt que detalla el tiempo dedicado a cada tarea. El Anexo C describe todos los experimentos llevados a cabo para evaluar el sistema de reconocimiento inicial implementado, mientras que en el Anexo D se exponen todos los experimentos realizados sobre el sistema de reconocimiento final. En el Anexo E aparece una descripción de los módulos que forman el sistema desarrollado. Finalmente, en el Anexo F aparece información de los objetos que se han utilizado como modelos para la base de datos del sistema de reconocimiento.

[2] http://www.kubuntu.org
[3] http://openkinect.org/wikiMain_Page
[4] http://openni.org
[5] http://opencv.willowgarage.com/wiki

2. Reconocimiento de objetos con información 2D

2.1. Introducción

Como paso previo al reconocimiento con cámaras RGB-d, este proyecto estudia las técnicas en reconocimiento basado solo en visión, es decir, toda la información con la que se cuenta proviene de las fotos que una cámara convencional pueda tomar. En este contexto se han desarrollado multitud de técnicas que intentan resolver el problema, muchas de las cuales pueden encontrarse en [3]. Estos trabajos son el punto de partida sobre el cual se aplicarán las mejoras que aportan las técnicas 3D.

En este capítulo se van a presentar algunas de las citadas técnicas que se han estudiado, por ser de uso extendido o por su reciente aparición. Todas estas técnicas comparten una estructura similar que se resume en la Figura 2.1. La entrada al sistema corresponde con una imagen que contiene los objetos a reconocer, se extraen características de dicha imagen que luego son comparadas con las características de las imágenes almacenadas en una base de datos y finalmente, según que imágenes de referencia tengan características más similares se estima que objetos aparecen en la imagen.

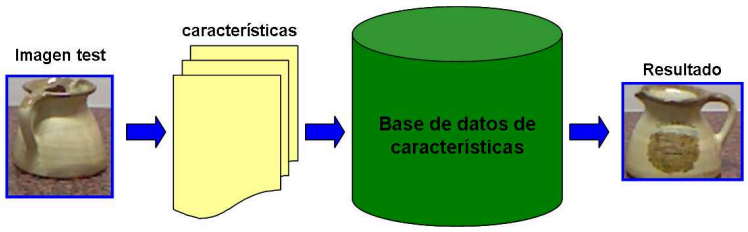

Figura 2.1: La imagen de entrada al sistema se procesa para extraer sus característica descriptivas, posteriormente se busca en la base de datos la imagen con las características más similares y se propone dicha imagen como el objeto reconocido.

En las siguientes secciones se resume por tanto como representar las imágenes y como utilizar dicha representación para evaluar la similitud entre imágenes. Finalmente se muestra la evaluación que se ha llevado a cabo de dichos métodos y que decisiones se han tomado a raíz de dicha evaluación.

2.2. Representación de las imágenes

El primer paso a tomar en un problema de procesamiento de imágenes es el de decidir cuál será la representación de la información de la imagen. En la mayoría de los casos no es conveniente trabajar con la información en crudo de cada píxel de la imagen, sino que se realiza un procesamiento para obtener descriptores más complejos que sean representativos de la imagen. Estos descriptores pueden ser de distinta naturaleza, aunque pueden dividirse en dos categorías.

2.2.1. Descriptores locales

En este caso, la imagen se representa por un conjunto de descriptores que capturan las partes más representativas de la imagen. Las zonas de la imagen en las que se calculan dichos descriptores vienen determinadas mediante distintos procesos heurísticos, dependiendo del descriptor calculado. Existen multitud de descriptores de esta naturaleza, aunque con filosofías muy diferentes. Por ejemplo, los hay basados en esquinas, contornos, gradientes... Su principal ventaja es su alta capacidad representativa, es decir, el conjunto de los descriptores locales que describen a la imagen, realmente consiguen capturar las zonas más representativas a un nivel de detalle suficiente para poder tratar el problema del reconocimiento de objetos. Por otra parte, estos descriptores suelen ser bastante costosos de calcular, las operaciones para comparar su similitud son pesadas y es necesaria más cantidad de memoria para almacenarlos. Se han estudiado dos alternativas de descriptores locales.

Descriptores ORB

Los descriptores ORB (*Oriented FAST and Rotated BRIEF*) [4] se han estudiado por ser una alternativa reciente a otros descriptores de uso más extendido. Como su propio nombre indica, los descriptores ORB están basados en otros métodos anteriores, como el detector de puntos de interés FAST [5] y los descriptores BRIEF [6]. La idea de estos descriptores es combinar la rapidez del detector de puntos de interés FAST con las propiedades que aporta el recientemente desarrollado descriptor BRIEF, consiguiendo así un descriptor rápido de calcular y que presenta un buen rendimiento. Además, el tamaño del descriptor es reducido, lo que ayuda a mitigar el problema de memoria que arrastran los descriptores locales. El problema que presenta este descriptor, así como todos los basados en cálculos de gradiente, es que aplicado al problema del reconocimiento de objetos, solo tiene utilidad con objetos que presentan una gran cantidad de cambios del gradiente en sí mismos, los llamados objetos con textura, ya que en caso contrario el algoritmo no detecta puntos clave o de interés donde calcular el descriptor.

Descriptores SURF

Los descriptores SURF (*Speeded-Up Robust Features*) [7] se han estudiado por haber sido durante años uno de los descriptores más utilizados y que mejor rendimiento han experimentado en aplicaciones de reconocimiento y en general de visión por computador. Estos descriptores surgen como alternativa a los descriptores SIFT [8], descriptores muy robustos, invariantes a escala y orientación, pero a la vez demasiado

complejos como para realizar aplicaciones de reconocimiento en tiempo real, pues el cálculo de similitud entre descriptores es excesivamente pesado. Así pues, los descriptores SURF proponen una aproximación para mantener la robustez de los descriptores SIFT pero con una mayor eficiencia en el cálculo y evaluación de similitud de los descriptores. Sin embargo, los descriptores SURF sufren el mismo problema que los descriptores ORB, solo son aplicables a objetos con textura. En la Figura 2.2 puede verse el resultado de la extracción de puntos SURF de un mismo objeto en dos vistas diferentes. Se observa que el reconocimiento mediante descriptores SURF solo se podrá aplicar a la vista con textura.

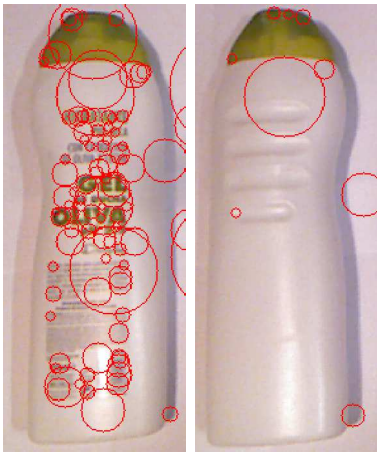

Figura 2.2: Se observa la gran diferencia entre la cantidad de puntos SURF encontrados en la figura de la izquierda respecto a la de la derecha, debido a la poca textura de la imagen de la derecha.

2.2.2. Descriptores globales

En este caso, un único descriptor representa a la imagen completa. Estos descriptores pueden ser histogramas de color de la imagen, nivel de gris de la imagen, histogramas del gradiente entre píxeles vecinos... La principal ventaja de estos descriptores radica en la rapidez de su cálculo y las operaciones que se realizan con ellos, así como la poca memoria necesaria para almacenarlos. Sin embargo, los descriptores globales son menos representativos que los locales, por lo que el rendimiento que ofrecen al nivel de detalle requerido para identificar un objeto puede ser demasiado bajo. Además, son más sensibles a los cambios de escena. Como ejemplo de descriptor global, se han estudiado los histogramas de color.

Histogramas de color

Los histogramas de color se han estudiado como una alternativa para el reconocimiento en objetos sin textura, donde los descriptores mencionados anteriormente carecen de utilidad.

Existen numerosas alternativas de representar la información del color, tradicionalmente, el sistema RGB (*Red, Blue, Green*) ha sido el más utilizado, el cual consiste en representar el color de un determinado píxel como la composición en términos de la intensidad de los colores primarios con que se forma. Sin embargo, este sistema

es muy sensible a cambios de escena y los histogramas fabricados a partir de este espacio de colores son de poca utilidad.

Por tanto, hay que encontrar otro sistema para representar el color, y para la realización de este proyecto, se ha optado por el sistema HSV (*Hue, Saturation, Value* - Matiz, Saturación, Valor). Concretamente, solo se ha utilizado el campo *Hue* puesto que es este valor el que denota el tono o matiz de un color y es relativamente invariante a cambios de escena, mientras que los otros valores están relacionados con la pureza de excitación y la iluminación, por lo que son muy sensibles a un cambio de escena. El matiz, que se representa como un ángulo, puede tomar valores entre 0° y 360°, por lo que podríamos construir un histograma de 360 componentes. No obstante, se ha decidido reducir el número de componentes del histograma hasta 16, evitando así que ligeras variaciones del matiz degraden las medidas de similitud. En la Figura 2.3 se observa unas imágenes con sus histogramas asociados.

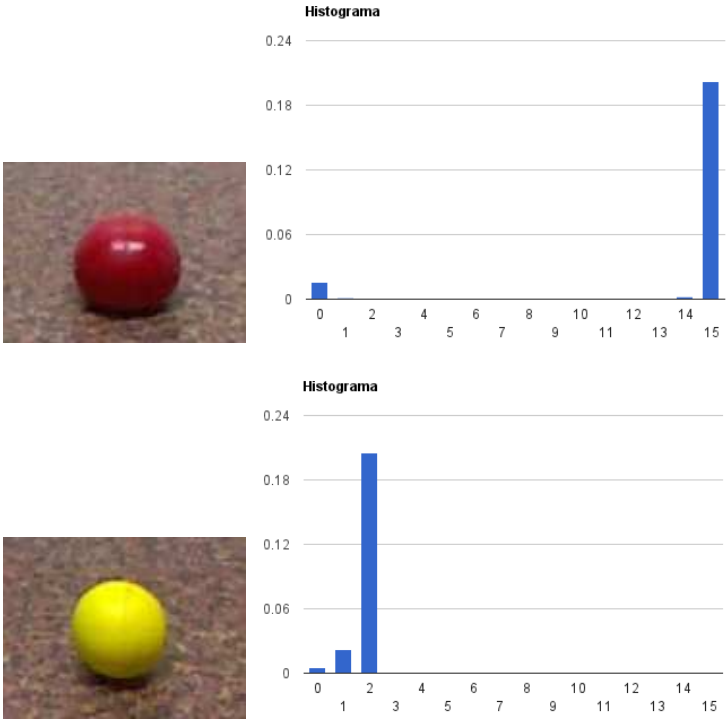

Figura 2.3: De izquierda a derecha, el objeto con su histograma asociado. Puede notarse la gran diferencia entre los histogramas de los dos objetos. El histograma solo pertenece a los píxeles del objeto, sin tener en cuenta el fondo, que deberá ser segmentado con anterioridad.

2.3. Algoritmos de similitud

Una vez que se ha decidido como va a representarse la información de las imágenes, el siguiente paso natural es el de decidir que medida de similitud se va a utilizar. Elegir de forma adecuada la medida de similitud que mejor se ajuste a las necesidades del problema es de gran importancia, ya que el comportamiento del sistema puede variar drásticamente dependiendo de la medida utilizada.

En esta sección se presentan las distintas técnicas estudiadas, aunque todas ellas comparten una estructura que se resume en la Figura 2.4:

1. **Fase de entrenamiento:** El sistema extrae los descriptores deseados de las imágenes de referencia y los organiza en una estructura que facilite una búsqueda eficiente.

2. **Fase de consulta:** Por cada nueva imagen a reconocer, el sistema extrae sus descriptores para luego evaluar la similitud en la estructura almacenada, decidiendo así qué imagen de referencia es más similar.

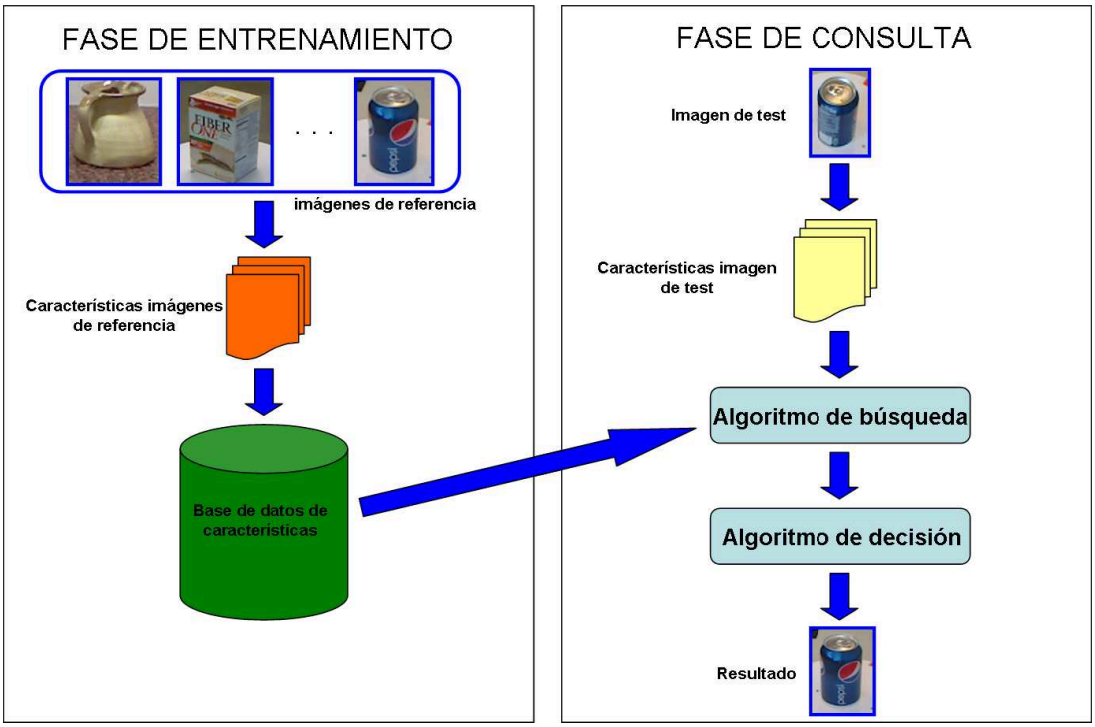

Figura 2.4: Algoritmo de búsqueda de imágenes similares en una base de datos. Se divide en dos fases, la fase de entrenamiento y la de consulta.

2.3.1. Búsqueda del vecino más cercano

Esta técnica consiste en comparar los descriptores de la imagen de test con todos los descriptores de las imágenes de referencia, obteniendo un número de correspondencias para cada imagen. La imagen de referencia con más correspondencias será considerada la más similar. En la Figura 2.5 aparece un esquema del funcionamiento del algoritmo.

Se pueden utilizar diversas alternativas como algoritmo de búsqueda entre correspondencias. Las expuestas a continuación son las estudiadas:

Búsqueda exhaustiva

La versión básica del algoritmo. En la fase de entrenamiento simplemente se habrán almacenado todos los descriptores de todas las imágenes de referencia. La búsqueda

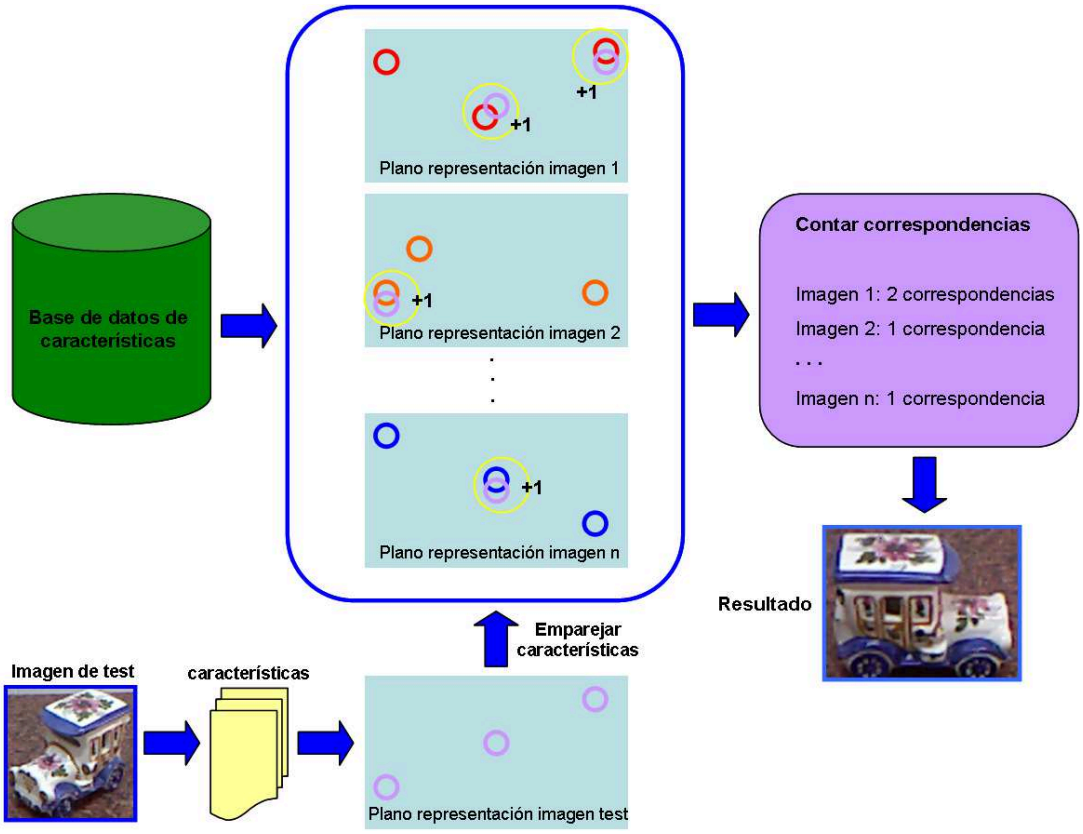

Figura 2.5: Algoritmo de búsqueda del vecino más cercano. Las características de la imagen de test se comparan con las características de las imágenes de referencia, obteniendo como resultado aquella imagen de referencia donde se encuentren más correspondencias con la imagen de test.

recorre todos los descriptores de cada imagen de referencia y las compara con los descriptores de la imagen de test. Se garantiza que se encuentra la solución óptima, pero el coste de ejecución es cuadrático con el número de descriptores que se quieren emparejar: $O(n^2)$.

Búsqueda aproximada

En esta alternativa, se sustituye la búsqueda exhaustiva o de fuerza bruta por una búsqueda aproximada. Para ello, en la fase de entrenamiento se genera una estructura más compleja, los *kd-trees* [9] para ordenar los datos. Esta estructura de árbol logra reducir el coste de ejecución a $O(\log n)$. La clara ventaja de esta alternativa es la disminución del coste de cómputo, sin embargo, el algoritmo no garantiza encontrar la solución óptima, por lo que los resultados pueden empeorar.

Estas técnicas sirven para comparar imágenes representadas tanto con descriptores globales como locales, la diferencia radica en cómo medir la distancia entre descriptores. A continuación se explica con más detalle cuales han sido las medidas utilizadas, dependiendo de la naturaleza del descriptor.

Búsqueda del vecino más cercano con descriptores locales

Para medir la distancia entre los descriptores locales estudiados, SURF y ORB, se ha aplicado la distancia euclídea. No se ha estudiado una mediada más compleja ya que el número de descriptores a comparar puede ser muy grande, así como las componentes de cada descriptor, por lo que una medida más compleja resultaría en un tiempo de cómputo demasiado elevado.

Por otra parte, con los descriptores locales se puede realizar una comprobación más robusta de las correspondencias (vecinos más cercanos) encontradas mediante el algoritmo RANSAC, explicado a continuación.

RANSAC

La medida explicada anteriormente puede obtener correspondencias incorrectas entre las imágenes, por lo que puede ser recomendable añadir un paso de estimación robusta con objeto de eliminar dichas correspondencias erróneas. El algoritmo RANSAC (*RAndom SAmple Consensus*) [10] es un buen complemento para conseguirlo. Este método añade una restricción geométrica que consigue rechazar las correspondencias que no sean consistentes con el modelo geométrico de la escena. Aunque aplicar el algoritmo RANSAC supone una importante mejora en los resultados basados en la búsqueda del vecino más cercano, no siempre es aconsejable utilizarlo, pues su coste de cómputo es elevado y supone un notable incremento en el tiempo de ejecución. En la Figura 2.6 se muestra el resultado de aplicar búsqueda del vecino más cercano y RANSAC.

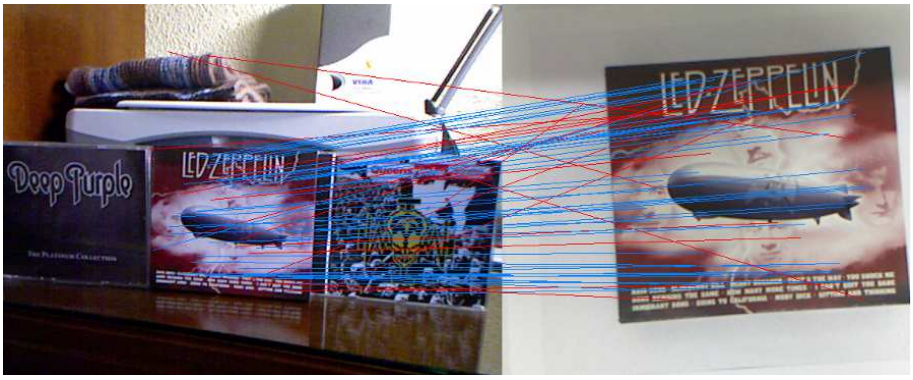

Figura 2.6: Correspondencias entre imagen de test (izquierda) e imagen de referencia(derecha). Las líneas azules representan las correspondencias encontradas por el algoritmo de búsqueda de vecino más cercano y admitidas por RANSAC. Las líneas rojas representan las correspondencias rechazadas por RANSAC.

Búsqueda del vecino más cercano con descriptores globales

En el caso de los descriptores globales, como el número de comparaciones es menor, pues solo hay un descriptor por imagen que comparar, se ha evaluado el uso de una distancia más compleja y robusta que la euclídea, aquí resumida.

Earth Mover's Distance

La técnica Earth Mover's Distance (EMD) [13] se ha estudiado para ser utilizada

como medida de similitud entre los histogramas de color. Se ha elegido esta medida porque el concepto se ajusta muy bien a las necesidades del problemas, ya que la EMD es una medida de distancia entre histogramas, en vez de emparejamiento. Esto resulta muy útil ya que histogramas de color del mismo objeto pero en diferentes escenas pueden sufrir cambios de iluminación, lo que causaría un desplazamiento de dichos histogramas, pero mantendrían la misma "forma". En estos casos la distancia EMD logra unos resultados más deseables que los conseguidos con otras técnicas conocidas de comparación de histogramas, como el método Chi-Cuadrado o la distancia de Bhattacharyya.

Conceptualmente, la distancia EMD se define como la cantidad de trabajo que llevaría encajar la forma de un histograma en la del otro. Calcular esta distancia se basa en resolver el conocido problema de transporte Monge-Kantorovich [14]. Supongamos una red de proveedores, cada uno con una cantidad de provisiones, y otra red de consumidores, cada uno con unas necesidades de consumo, y un coste de transporte entre cada proveedor y consumidor. El problema se reduce a encontrar el flujo de mínimo coste para que los proveedores satisfagan las necesidades de los consumidores.

Extrapolado al caso de los histogramas, la red de proveedores sería un histograma, cada componente un proveedor y el peso su cantidad de provisiones. El histograma restante representaría la red de consumidores, cada componente un consumidor y el peso las necesidades de consumo. El coste de transporte vendría determinado por la distancia entre las componentes de los histogramas.

2.3.2. Bolsa de palabras

Esta técnica, estudiada como alternativa a la búsqueda del vecino más cercano, pretende almacenar una representación más comprimida de los descriptores de la base de datos. Para ello primero hay que crear un vocabulario de palabras durante la fase de entrenamiento, que consiste en agrupar los n descriptores en un conjunto de k palabras o *clusters*, siendo $k \leq n$. Este vocabulario almacena por un lado los *centroides* de cada palabra y por otro una estructura denominada *inverted file index* [11] que almacena en que imágenes y con que frecuencia aparece cada palabra. Utilizando diversas medidas de similitud, basadas en analizar que palabras aparecen en la imagen de test, se puede decidir que imagen de referencia es la más similar. El proceso queda esquematizado en la Figura 2.7, y explicado más detalladamente a continuación.

Creación del vocabulario (fase de entrenamiento)

Para crear el vocabulario, se ha estudiado el algoritmo K-means [12], sin duda uno de los algoritmos de clusterización más utilizados. Este algoritmo necesita como entrada la información que queremos agrupar, en este caso los descriptores de las imágenes de referencia, y el número de *clusters* en las que queremos agruparla. Como salida obtendremos los *centroides* de cada *cluster* y una referencia sobre a que *cluster* queda asignado cada descriptor.

En este punto, se construye la matriz *inverted file index* con objeto de tener la información de forma más compacta y ordenada. Esta matriz se define como un histograma de votos a cada palabra de cada imagen de referencia (Figura 2.8).

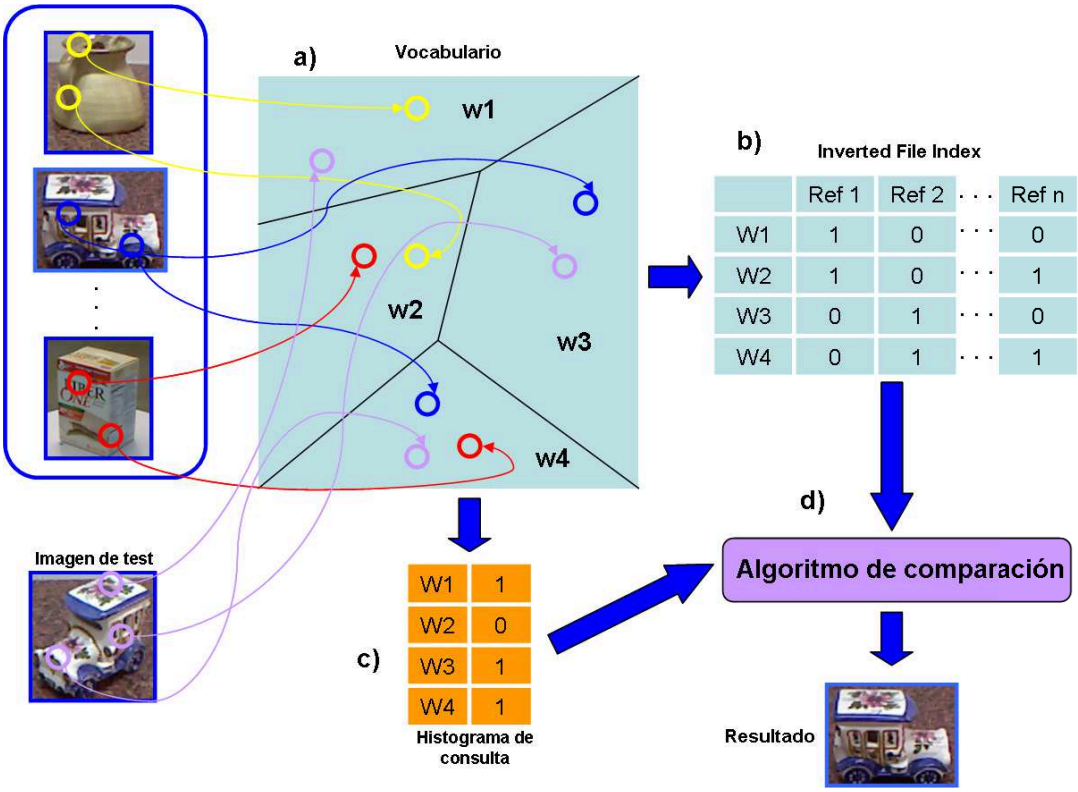

Figura 2.7: Algoritmo de la bolsa de palabras. A partir de los descriptores de las imágenes de referencia se forma el vocabulario a) de k palabras, en este caso 4. A partir de dicho vocabulario se crea la matriz *inverted file index* b), almacenando la información de manera más compacta y ordenada. Para cada imagen de test se crea su histograma de consulta c) a partir del vocabulario. Finalmente, con el histograma de consulta y la *inverted file index* se puede emplear un algoritmo de comparación d) para decidir qué imagen es la más similar.

Fase de consulta

Cuando se quiere evaluar la similitud de una imagen de test con las de referencia, se siguen los siguientes pasos:

1. Extracción de características de la imagen de test. En este caso se extraerían los descriptores ORB o SURF.

2. Comparando con el valor de los *centroides* del vocabulario, se asocia a cada nuevo descriptor de la imagen de test el *cluster* al que pertenece, según el centroide al que más se parece.

3. Con lo obtenido en el paso anterior, se crea un histograma de longitud el número de *clusters* del vocabulario, en el que queda reflejado la frecuencia con la que cada *cluster* o palabra aparece en la imagen.

4. Con la ayuda de este histograma y la matriz *inverted file index* pueden estudiarse diversas medidas de similitud entre los histogramas para decidir que imagen de referencia es la más similar. En la Sección C.0.2 se detallan las medidas estudiadas y los resultados obtenidos.

Inverted File Index

	Ref 1	Ref 2	···	Ref n
W1	5	5	···	2
W2	7	3	···	8
W3	2	10	···	7
W4	0	9	···	9
⋮	⋮	⋮		⋮
Wn	6	11	···	0

Figura 2.8: *Inverted File Index.* Representa para cada imagen de referencia el número de de votos a cada palabra del vocabulario, es decir, su histograma.

2.4. Experimentos y decisiones

En esta sección se presentan las decisiones que se tomaron gracias a los experimentos llevados a cabo con el fin de de decidir que formas de representar las imágenes y que algoritmos de similitud son los más adecuados para el desarrollo del proyecto. Los resultados detallados de todos los experimentos se encuentran en el Anexo C. Además, es necesario evaluar y tener un algoritmo de partida que solo utilice información 2D para poder evaluar las mejoras adquiridas gracias al uso de los nuevos sensores con información 3D. Por ello, para llevar a cabo estos experimentos se implementó un sencillo sistema de reconocimiento basado en técnicas 2D que utilizaba los algoritmos explicados anteriormente, exceptuando los dedicados a los histogramas de color, pues este sistema de base no incluye ningún tipo de segmentación, por lo que los histogramas que se calcularan incorporarían información del fondo de la imagen, incorporando demasiado ruido al histograma como para que éstos fueran de utilidad. La evaluación de este tipo de similitud se realiza más adelante, donde el sistema completo desarrollado en este proyecto cuenta con un proceso previo de segmentación.

El sistema mantiene una pequeña base de datos de 13 objetos, con 8 fotos de cada objeto, lo que hace un total de 104 fotos de referencia.

Una vez realizados todos los experimentos, se tomaron las siguientes decisiones.

- Es necesaria una etapa de segmentación de la imagen. El reconocedor mantiene unos buenos resultados mientras los objetos a reconocer se encuentren sobre un fondo blanco, pero cuando se enfrenta situaciones más reales, donde el objeto se encuentra rodeado de otros elementos, el rendimiento baja hasta extremos inaceptables.

- Se desechó la alternativa del algoritmo de bolsa de palabras como algoritmo de similitud. Los experimentos reflejan que los resultados obtenidos con el algoritmo de búsqueda del vecino más cercano superan claramente el rendimiento ofrecido por la bolsa de palabras. Por tanto, se decidió que el sistema de reconocimiento final integraría el algoritmo de similitud de búsqueda del vecino más cercano para reconocer objetos cuando se tratara de utilizar técnicas 2D.

- El descriptor local utilizado será el SURF. Los resultados muestran que aunque el coste de cómputo es mayor para los descriptores SURF, ofrecen unos resultados mucho mejores que la alternativa de los descriptores ORB.

- La medida de similitud entre dos imágenes Im_i y Im_j que mejores resultados ofrece,

y por tanto la que se utilizará en el sistema final, será la siguiente.

$$Sim = Matches_{ij}/Max(NumDesc_i, NumDesc_j) \tag{2.1}$$

Siendo $Matches_{ij}$ el número de correspondencias entre las imágenes i y j y $NumDesc_i$ el número de descriptores extraídos de la imagen i.

- La búsqueda exhaustiva ofrece unos resultados ligeramente mejores que la búsqueda aproximada, aunque también es más lenta. Sin embargo, la ganancia en cuanto a tiempo de ejecución que implica el uso de la búsqueda aproximada no es muy alta debido al tamaño de la base de datos utilizada. Por tanto, hasta que no se conozca con más detalle la arquitectura del sistema de reconocimiento final y el tamaño de la base de datos a manejar, se deja la puerta abierta a las dos opciones.
- Aplicar el algoritmo RANSAC mejora los resultados, sobretodo en experimentos de entorno más real, pero el coste de ejecución aumenta considerablemente. Por ello se decidió esperar ha implementar el proceso de segmentación, lo que permitirá evaluar si es necesario seguir aplicando RANSAC o el hecho de segmentar la imagen en regiones de posibles objetos ofrece robustez suficiente.

3. Reconocimiento de objetos con información 3D

3.1. Introducción

La aparición en el mercado de cámaras RGB-d con un bajo coste ha servido como catalizador para que las investigaciones sobre reconocimiento y en general sobre visión por computador utilizando información 3D avancen rápidamente. Antes de la aparición de dichos sensores, la visión en 3D podía conseguirse mediante la sincronización de dos cámaras estándar a una distancia conocida, a partir de la información obtenida de dichas cámaras se pueden obtener los valores de profundidad de la escena y posteriormente construir el modelo 3D. Sin embargo, construir dicha escena conlleva un coste de ejecución alto, por lo que resulta un problema para desarrollar sistemas que trabajen en tiempo real.

Los nuevos sensores obtienen los valores típicos de color, por ejemplo en formato RGB, y de profundidad de forma sincronizada y casi instantánea, por lo que el mayor problema queda eliminado. En la Figura 3.1 se observa una imagen típica de color, que podría haber sido tomada por cualquier cámara del mercado, con su correspondiente mapa de profundidad. En este caso el mapa de profundidad, que teóricamente marca para cada píxel de la imagen la distancia al sensor, se ha convertido en una imagen donde los colores más cálidos denotan más cercanía. Las zonas de color negro son valores de profundidad indeterminados, porque el sensor no ha podido realizar la medición correctamente.

Figura 3.1: Escena RGB (derecha) y su mapa de profundidad asociado (izquierda). Los píxeles del mapa de profundidad de valor negro indican valores indeterminados. Estas zonas suelen encontrarse en objetos transparentes o que reflejan la luz, como en este caso el cristal que se encuentra encima de la mesa.

En este capítulo se van a presentar las técnicas estudiadas que implican uso de

información 3D. Primero se discutirá sobre la necesidad de un nuevo modelo de representación de la información, (Sección 3.2) para continuar después con técnicas propias del problema de reconocimiento de objetos, como son la segmentación de la escena, que se pueden abordar de manera muy eficaz gracias al uso de información 3D (Sección 3.3) y el uso de descriptores de información 3D para construir los modelos de los objetos a reconocer (Sección 3.4).

3.2. Representación de la información

Un nuevo paradigma de trabajo, como es añadir una tercera dimensión a la información con la que se trabaja, conlleva plantear un nuevo sistema para representar dicha información. Hay que plantear un nuevo modelo que se ajuste a las necesidades del problema, ya no es suficiente con una matriz de píxeles que represente el color de cada imagen. Para ajustarse a estas necesidades se ha optado por trabajar con el modelo denominado nube de puntos, que se explica a continuación.

Nube de puntos

Una nube de puntos representa la posición XYZ de cada píxel de la escena respecto a la cámara con la que se ha tomado la foto, así como el color de dicho píxel. Por tanto, para cada punto que forme parte de la nube, existirán 4 números en coma flotante o *floats* que lo describan: Tres para denotar la posición XYZ y un cuarto que almacene de forma compacta los valores RGB del píxel.

Conseguir los valores XYZ de los puntos de la nube es relativamente sencillo si se cuenta con un mapa de profundidad que indique en metros la distancia de cada píxel a la cámara. Concretamente, para calcular los valores XYZ de un elemento $[i, j]$ del mapa de profundidad basta con aplicar las siguientes fórmulas.

$$X = (i - c_i) * depth[i, j]/f$$

$$Y = (j - c_j) * depth[i, j]/f$$

$$Z = depth[i, j]$$

Siendo c_i y c_j los índices del centro de la matriz de profundidad (en caso de que la matriz fuera de 640x480 elementos los valores serían 320 y 240 respectivamente), $depth[i, j]$ el valor de la profundidad en la posición $[i, j]$ y f la distancia focal, un parámetro propio de la cámara con la que se tome la escena. Adicionalmente, a esta información se le podría añadir los valores de color que correspondan al elemento $[i, j]$ del mapa de profundidad.

En la Figura 3.2 puede observarse varios puntos de vista de una nube de puntos que representa parte de una habitación, mientras que en la Figura 3.3 se observa de forma gráfica la información de todos los campos que forman los puntos de la nube.

3.3. Segmentación de la escena

Gracias a los resultados que arrojaron los experimentos de la Sección C.0.1, quedó claro que era necesaria una etapa de segmentación si se querían conseguir buenos resultados

Figura 3.2: Distintas perspectivas de una misma nube de puntos. La imagen de la izquierda corresponde con el punto de vista original con el que se tomó la foto, mientras que las de la derecha representan rotaciones de dicha escena.

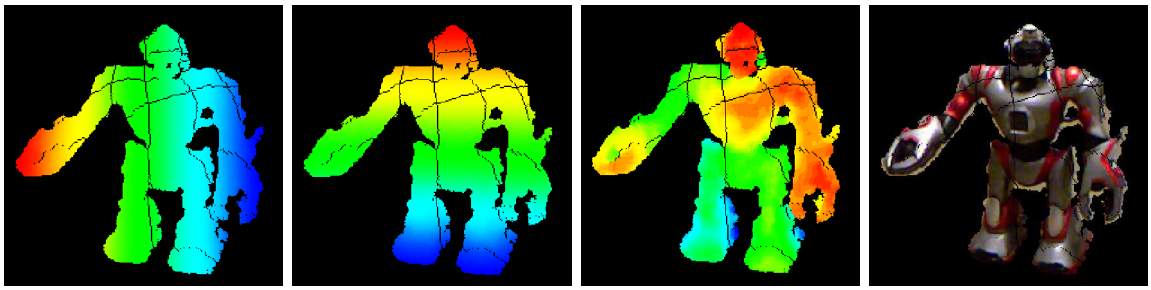

Figura 3.3: Distinta información que almacena la nube de puntos. De izquierda a derecha, componentes X, Y, Z (donde colores más cálidos indican distancias menores) y RGB del modelo.

trabajando en entornos reales, objetos sobre fondo con ruido. Aunque trabajos como [16] o [17] logran resultados aceptables trabajando solo con información 2D, pero en algunos casos el coste es muy elevado, para la realización de este proyecto se ha optado por utilizar técnicas que trabajen con nubes de puntos, pues técnicas basadas en 3D facilitan y mejoran los resultados que se obtendrían con herramientas 2D.

Por otra parte, aplicar segmentación a una escena para conservar solo la información que realmente interesa para el reconocimiento tiene también la ventaja de permitir que el proceso sea más rápido, pues habrá menos información que procesar.

Las dos técnicas de segmentación que se han utilizado en este proyecto son la eliminación de planos que suelen pertenecer al fondo de la escena y la clusterización de puntos contiguos que son probables de pertenecer a un mismo objeto. Estas técnicas se explican en las siguientes secciones, pero antes de ello, va a mostrarse un filtro que se ha utilizado como paso previo a la segmentación para hacerla más sencilla.

Filtro *Pass Through*

Este filtro se sirve de la representación de la información en nubes de puntos para poder desechar fácil y rápidamente todos los puntos de la nube que estén más allá de una distancia dada, en cualquiera de las direcciones de los ejes de coordenadas. Por ejemplo, podemos optar por descartar toda la información de la escena que se encuentre a una profundidad mayor de 1.5 metros con respecto a la cámara, lo que correspondería en este caso con la dirección Z.

Pensando en el problema de reconocimiento de objetos pequeños, donde está centrado el proyecto, resulta muy útil descartar de antemano toda la información que

estaría demasiado lejos como para poder reconocer algo con éxito. En la Figura 3.4 se puede observar las ventajas de utilizar este filtro.

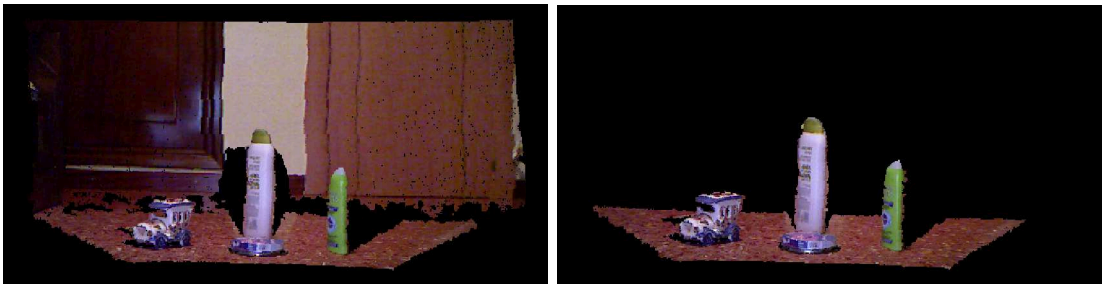

Figura 3.4: Filtro *Pass Through*. A la izquierda la nube original, a la derecha la nube eliminando todo lo que se encuentre más allá de una distancia dada.

3.3.1. Eliminación de planos de fondo

Como normal general, sobretodo cuando se trabaja sobre entornos reales, los objetos a reconocer suelen estar encima de una mesa, escritorio, suelo... En definitiva, sobre un plano. Resulta claro que una eliminación de dicho plano en la nube de puntos facilitaría enormemente el proceso de reconocimiento, pues se prescinde de aquella parte de la escena que no contiene los objetos. Lograr esto implica que el sistema de reconocimiento no pueda confundir una superficie o parte de ella con un objeto o parte de él. Pensando en superficies con textura como podría ser una pared con baldosas decoradas o un suelo con diferentes colores, la necesidad de este paso cobra fuerza.

Recordando que ahora se trabaja con la información almacenada en nubes de puntos, donde para cada punto tenemos las componentes XYZ respecto a la cámara, no resulta muy complicado encontrar los planos que se encuentran en la escena. Para lograrlo se ha utilizado una función de la librería PCL [1] que utilizando el algoritmo RANSAC [10] para eliminar puntos atípicos (los que no corresponden al plano) y considerando una distancia dada la máxima entre dos puntos que pertenezcan al mismo plano, devuelve los coeficientes del plano dominante encontrado en la forma $ax + by + cz + d = 0$.

Una vez que se ha encontrado la ecuación del plano, no hay más que eliminar los puntos de la nube que correspondan a dicho plano. En la Figura 3.5 se observa el resultado de aplicar la eliminación del plano dominante a una escena.

3.3.2. Clusterización de puntos pertenecientes al mismo objeto

En este momento, si los pasos de segmentación anteriores han ido bien, la nube de puntos solo contará con cuerpos aislados entre si que no formen un plano. Esta información desordenada, que contiene los objetos que se quieren reconocer más ruido u otros elementos de la escena, necesita ser ordenada en *clusters* que logren separar posibles objetos de otros, y así poder reconocer cada objeto de forma separada. Este paso es imprescindible para hacer posible el reconocimiento de varios objetos en la misma escena, pues no serviría de nada hacer una medida de similitud entre los objetos almacenados en la base de datos y una imagen de test en la que aparecen varios objetos.

Figura 3.5: Eliminación de planos. A la izquierda la nube original, a la derecha la nube tras eliminar el plano del suelo.

Para lograr esta agrupación de la información se ha optado por utilizar un algoritmo, integrado en la librería PCL [1] que de nuevo se sirve de las ventajas del formato de nube de puntos. Se trata de un extractor de *clusters* euclídeo, cuyos pasos se explican a continuación.

1. Crear una representación con un *kd-tree* para la nube de puntos de entrada P;
2. Crear una lista de clusters C, y una cola de puntos que necesitan ser comprobados Q;
3. Entonces, para cada punto $p_i \in P$ hacer:
   - Añadir p_i a la cola actual Q;
   - Para cada punto $p_i \in Q$ hacer:
     - Buscar el conjunto P_k^i de vecinos de p_i en una esfera con radio $r < d$;
     - Para cada vecino $p_k^i \in P_k^i$ comprobar si el punto ya ha sido procesado, y si no añadirlo a Q;
   - Cuando la lista de todos los puntos en Q se ha procesado, añadir Q a la lista de clusters C, y resetear Q a una cola vacía;
4. El algoritmo termina cuando todos los puntos $p_i \in P$ han sido procesados y son ahora parte de la lista de clusters C;

Cuando se ha acabado el proceso de clusterización, es conveniente aplicar un filtro que deseche los clusters demasiado pequeños o demasiado grandes. Esto tiene como objetivo eliminar puntos espurios aislados que han quedado como proceso de la eliminación del plano o los clusters que superen en tamaño la naturaleza de los objetos que se quieren reconocer. En la Figura 3.6 queda plasmado el resultado de la clusterización a una escena.

Figura 3.6: Proceso de clusterización de la escena. Arriba a la izquierda una escena tras la etapa de segmentación del plano. A la derecha la misma escena pero coloreada de un mismo tono para notar más claramente el ruido que queda tras eliminar el plano. Debajo, el resultado de agrupar la imagen, donde cada tonalidad representa un *cluster* distinto. Nótese que se ha eliminado el ruido, ya que forma *clusters* demasiado pequeños.

3.4. Descriptores de información 3D

Al igual que se hizo cuando se habló de las técnicas 2D, donde se introdujo el concepto de descriptores, ahora también es interesante representar la información de la nube de forma más compacta logrando capturar sus características más discriminantes. En este proyecto, después de estudiar las opciones disponibles en la literatura y las librerías de manejo de *kinect*, se ha optado por utilizar el descriptor de forma VFH (Viewpoint Feauture Histrogram) [18].

Descriptor VFH

Este descriptor encaja dentro de los descriptores globales. En concreto es un histograma calculado por cada imagen, que captura tanto la forma del objeto como el punto de vista desde donde se toma la imagen.

Entre las razones encontradas para utilizar este descriptor, resaltan los buenos resultados que muestran sistemas de reconocimiento estudiados como [19] que utiliza dicho descriptor. Además, al tratarse de un descriptor global, la memoria necesaria para almacenarlos es mínima, pues solo se calcula un descriptor por objeto, y las operaciones de comparación entre ellos son rápidas. No obstante, para almacenar el modelo de un objeto

en la base de datos de forma efectiva es necesario almacenar varios de estos descriptores desde distintos puntos de vista, para así capturar la forma y posibles puntos de vista del objeto por completo.

Para calcular este descriptor, primero es necesario calcular las componentes normales de la nube de puntos, en nuestro caso del objeto. Estas componentes se utilizan en gran variedad de áreas y existen numerosos métodos para calcularlas. Dada una superficie geométrica, suele ser trivial inferir la dirección de la normal de un punto en dicha superficie como el vector perpendicular a la superficie en dicho punto. Sin embargo, puesto que las nubes de puntos representan conjuntos de puntos en la superficie real es necesario usar aproximaciones para inferir las normales directamente desde el punto de la nube. Para calcular dichas normales se ha utilizado una función implementada en la librería PCL. Para más información de los cálculos utilizados, referirse a [20]. En la Figura 3.7 se observa una representación de las normales de un subconjunto de los puntos de un objeto.

Figura 3.7: Representación de las normales de un objeto. El objeto, en este caso una jarra, se muestra como fondo y los vectores de color blanco representan la dirección de las normales en un subconjunto de los puntos de la nube.

Una vez calculadas las componentes normales, se puede calcular el descriptor VFH. Para ello, se calculan los ángulos *pan-tilt-yaw* entre las normales de cada punto del objeto y las normales del centroide del objeto, para después plasmar la información en un histograma. Concretamente, para cada punto p_i del objeto, su componente normal n_i y el centroide p_c se calculan los siguientes ángulos:

$$\alpha = v \cdot n_i$$

$$\phi = u \cdot \frac{p_i - p_c}{d}$$

$$\theta = \arctan(w \cdot n_i, u \cdot n_i)$$

Donde *u, v, w* representan un marco de Darboux[1] elegido en p_i. La Figura 3.8 representa la selección del marco de Darboux y una representación gráfica de los tres ángulos calculados.

A parte de los datos calculados hasta ahora, que describirían la forma del objeto, el descriptor incorpora información sobre el punto de vista desde el que se tomó la foto. Esta información resulta útil para poder reconocer la pose en la que se encuentra el objeto, aplicación interesante en campos como la robótica. Para obtener la información sobre

[1]En geometría diferencial de superficies, un marco móvil construido en una superficie.

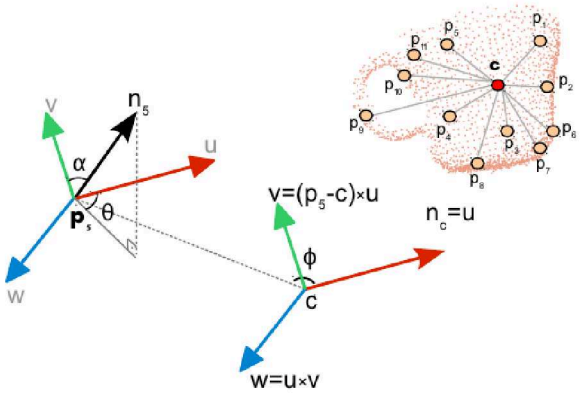

Figura 3.8: El descriptor almacena estadísticas de los ángulos relativos entre las normales de cada punto a la normal del centroide del objeto. La parte de abajo a la izquierda describe los tres ángulos calculados para un par de puntos de ejemplo. Figura obtenida de [18].

el punto de vista se calcula un histograma de los ángulos que hacen cada componente normal con el punto de vista central trasladado a dicha normal. La Figura 3.9 representa gráficamente el concepto.

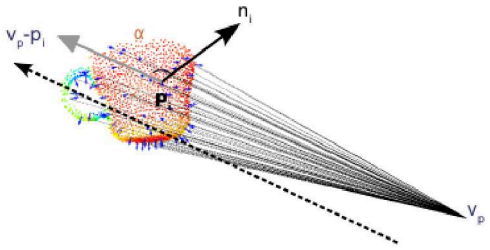

Figura 3.9: Parte de la información del descriptor VFH se calcula a partir de las estadísticas de los ángulos relativos del punto de vista central con cada componente normal de la nube. Figura obtenida de [18].

Toda esta información se distribuye en un histograma de la siguiente manera: 60 subdivisiones para cada uno de los tres ángulos *pan-tilt-yaw*, y 128 subdivisiones para la componente del punto de vista, lo que en total hace un total de 308 componentes, es decir, solamente son necesarios 308 *floats* para almacenar el descriptor. En la Figura 3.10 se muestra una representación de dicho histograma.

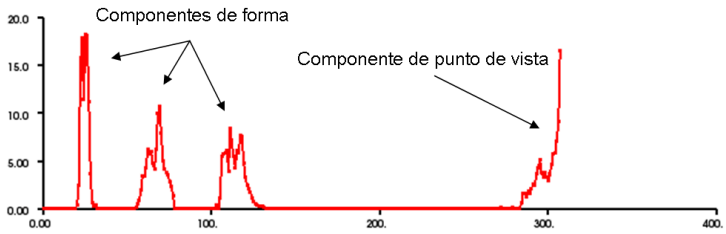

Figura 3.10: Ejemplo de un histograma VFH para uno de los objetos usados. Nótese la concatenación de las dos componentes.

Los resultados obtenidos de aplicar este descriptor se muestran en la Sección 4.3, junto con el rendimiento de todo el sistema implementado.

4. Sistema de reconocimiento de objetos diseñado

En este capítulo va a exponerse todo lo relacionado con el sistema diseñado e implementado basándose en los estudios explicados anteriormente. Primero se abordará el proceso de creación de los modelos de los objetos en la base de datos (Sección 4.1) para después describir cómo funciona el sistema (Sección 4.2), el método de decisión seguido (Sección 4.2.1) y finalmente evaluar el rendimiento conseguido (Sección 4.3). Para más información sobre los módulos que forma el sistema, véase el Anexo E.

4.1. Creación de modelos de los objetos a reconocer

Cada modelo de objeto guardado en la base de datos consta de un conjunto de descriptores VFH que denotan su forma y punto de vista desde que se tomó la imagen, un conjunto de histogramas de color y si el objeto tuviera textura, descriptores SURF que lo describen.

Todo este conjunto de datos proviene de las nubes de puntos que se han debido tomar de cada objeto. Como normal general, cuantas más imágenes y puntos de vista distintos se almacenen del objeto en cuestión, mejor descrito quedará en la base de datos. Sin embargo, un exceso de información puede retrasar demasiado el funcionamiento del sistema, por lo que es necesario alcanzar un compromiso. En este proyecto se ha optado por trabajar con 3 puntos de vista distintos para cada objeto (lo más comunes según el objeto) y 8 fotos por cada punto de vista, lo que hace un total de 24 nubes de puntos tomadas para cada objeto. No obstante, el sistema es flexible en cuanto al número de datos almacenados por objeto, por lo que podría variar.

La entrada al sistema de creación del modelo consta por tanto de un número de nubes de puntos del mismo objeto. En la Figura 4.1 se explica el proceso de creación de dicho modelo. Primero, la nube de puntos original se segmenta para obtener la nube de puntos del objeto, una imagen 2D del tamaño mínimo que englobe el objeto y una imagen de máscara que denota cuáles de los píxeles de la imagen 2D pertenecen realmente al objeto, ver Figura 4.2. Cabe destacar que para obtener la imagen recortada tras la segmentación se añadieron dos campos adicionales a cada punto de la nube para facilitar la tarea, que son los índices $[i, j]$ de la imagen 2D completa a los que pertenece cada punto de la nube. Así, una vez eliminados de la nube los puntos que no forman parte del objeto, se puede extraer fácilmente los píxeles de la imagen mínima y su máscara. Tras la segmentación, se extrae el descriptor VFH del *cluster* a partir de la nube recortada. Después, gracias a la imagen 2D y la máscara, se intentan extraer los descriptores SURF del objeto, almacenando

únicamente los descriptores de las imágenes que alcancen un determinado número de puntos SURF. Nótese que un objeto puede tener textura o no dependiendo del punto de vista, por lo que el número de imágenes de las que se guardan sus descriptores SURF puede variar desde 0 (objeto sin textura) al total de imágenes tomadas (objeto con mucha textura). Por último, y de nuevo a partir de la imagen 2D y la máscara, se construye el histograma de color del objeto, para detalles de la construcción del histograma véase Sección 2.2.2. Conviene señalar que cada descriptor, de la clase que sea, almacenado en la base de datos está asociado a una etiqueta que denota de que objeto se trata, que se almacena al mismo tiempo que el propio descriptor.

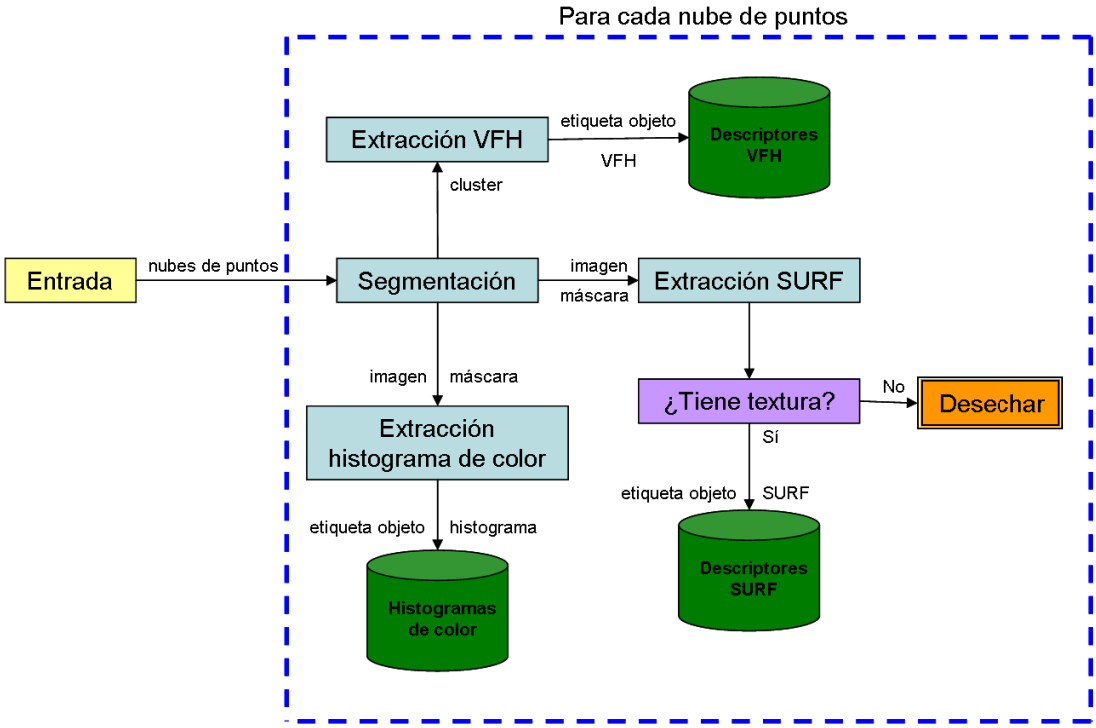

Figura 4.1: Diagrama de creación de modelos de los objetos. Cada nube de puntos se segmenta, a partir de cada nube de puntos segmentada se almacena el descriptor VFH y a partir de cada imagen mínima y su máscara se almacena el histograma de color y los descriptores SURF en caso de que el objeto tuviera textura. También se almacena una etiqueta junto a cada descriptor que denota de que objeto se trata.

Figura 4.2: Datos de la nube de puntos tras la segmentación. De izquierda a derecha la nube segmentada, la imagen 2D mínima y la máscara.

Así pues, para cada objeto se almacena alrededor de 24 descriptores VFH, el mismo número de histogramas de color, y un número de descriptores SURF variable asociado a cada imagen en la que se haya encontrado textura en el objeto. Con toda esta información

queda construido el modelo del objeto, definiéndolo a un nivel de detalle suficiente como para poder tratar el problema del reconocimiento de objetos con garantías.

4.2. Funcionamiento del sistema

En esta sección se va a explicar cómo funciona el sistema de reconocimiento, describiendo todos los pasos que sigue hasta dar con la solución. En la Figura 4.3 se describe gráficamente el proceso, y a continuación se explica con detalle los pasos del algoritmo.

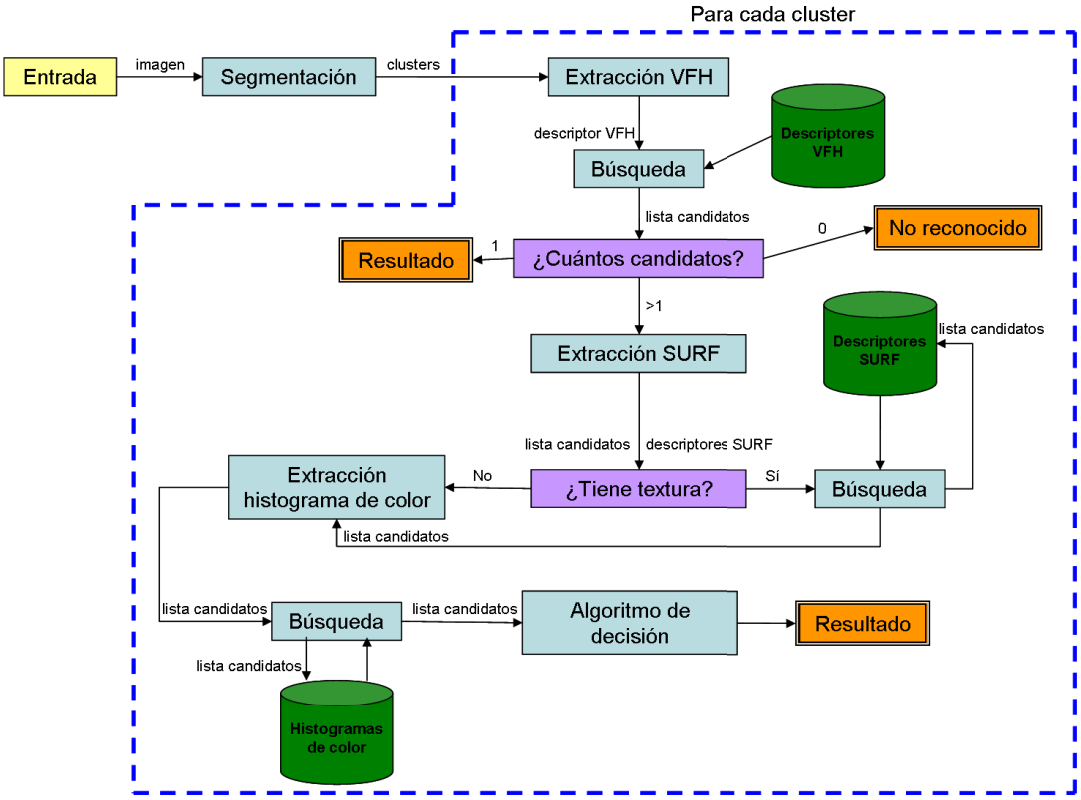

Figura 4.3: Diagrama de reconocimiento del contenido de una imagen de test. Tras la segmentación, para cada cluster se extraen los distintos descriptores, se realizan búsquedas y se obtienen medidas de similitud con los descriptores de los objetos almacenados en la base de datos. Una vez obtenida dicha información, se decide que objeto es el cluster que se está reconociendo.

1. La entrada al sistema corresponde con una nube de puntos, que es segmentada para conseguir dividirla en diferentes *clusters* que representarán los distintos objetos que aparezcan en la escena. Por tanto, para cada *cluster*, al igual que en el proceso de creación del modelo, se tiene la nube de puntos recortada, la imagen 2D mínima y su máscara. A partir de este momento, los siguientes pasos se realizan para cada *cluster* encontrado.

2. Se extrae el descriptor VFH del *cluster* y se realiza una búsqueda con la base de datos de descriptores VFH almacenados. Esta búsqueda se realiza en un *kd-tree*

y se calcula la distancia Chi-Cuadrado del descriptor de test a los k vecinos más cercanos. La fórmula de dicha distancia para el descriptor x del *cluster* a reconocer y un descriptor y de un objeto de la base de datos es la siguiente:

$$\chi^2 = \sum_i \frac{(x_i - y_i)^2}{y_i}$$

El hecho de llevar a cabo la búsqueda en un árbol implica que el tiempo de búsqueda conlleva un coste de $O(\log n)$ siendo n el número de objetos almacenados en la base de datos. Que el coste sea logarítmico respecto al aumento de la base de datos es una característica muy deseable en sistemas de reconocimiento que necesitan ser flexibles a posibles incrementos de tamaño.

3. Gracias a la búsqueda realizada se construye una lista de posibles objetos candidatos. Esta lista representa los objetos que el sistema cree que puede ser el *cluster* en cuestión. Esta característica permite que las siguientes búsquedas se realicen sobre un subconjunto de la base de datos de objetos, permitiendo mayor rapidez. Esta jerarquización en la búsqueda de los distintos descriptores también es una característica deseable y muy usada en sistemas de reconocimiento, como por ejemplo en [19, 21, 22]. Más información sobre cómo se construye esta lista de candidatos y qué información almacena durante todo el proceso de reconocimiento se verá a continuación en la Sección 4.2.1.

4. Si la lista no contiene ningún candidato, se dice que el objeto a reconocer no corresponde con ningún objeto de la base de datos, si solamente hay un candidato, esa será la solución. Pero si la lista contiene más de un candidato, son necesarios los siguientes pasos para intentar decidir de qué objeto se trata.

5. Se extraen los puntos SURF de la imagen del *cluster*, con ayuda de la máscara para solo extraer en las zonas que pertenezcan al objeto. Si se supera un umbral determinado de puntos encontrados, se considera que el objeto tiene textura y se realiza una búsqueda, mientras que si no se alcanza el umbral se pasa al paso siguiente. Esta búsqueda está basada en el algoritmo de búsqueda del vecino más cercano, pero solo se compara con los objetos de la lista de candidatos. Por este motivo la búsqueda que se hace es exhaustiva en lugar de aproximada, pues se comparará con pocos candidatos. No obstante el sistema permite configurar que esta búsqueda sea aproximada.

6. En este paso se computa el histograma de color del *cluster* y se realiza una búsqueda con los histogramas de los objetos de la lista de candidatos. Como medida de similitud se utiliza la distancia EMD explicada en 2.3.1.

7. Al llegar ha este punto la lista de candidatos ha ido recolectando información de las búsquedas entre los distintos descriptores, almacenando una puntuación entre el *cluster* de test y los objetos candidatos. Sin embargo, la puntuación adquirida para cada una de las tres búsquedas es de muy diferente naturaleza, por lo que debe normalizarse para poder ser tenida en cuenta por igual. Un algoritmo de decisión se

encarga de ello y de estimar a partir de esa información cual es el objeto. Para más información sobre este paso del algoritmo véase la Sección 4.2.1.

8. Por último, cuando ya se ha decidido qué objeto es cada *cluster* de la nube de puntos de entrada, se muestra una imagen señalando los objetos que el sistema estima en la posición de los *clusters* originales. El sistema también cuenta con 3 niveles de verbosidad que permiten obtener por consola mayor o menor información sobre la traza de ejecución del sistema. En la Figura 4.4 se muestra una entrada al sistema de reconocimiento y la salida obtenida.

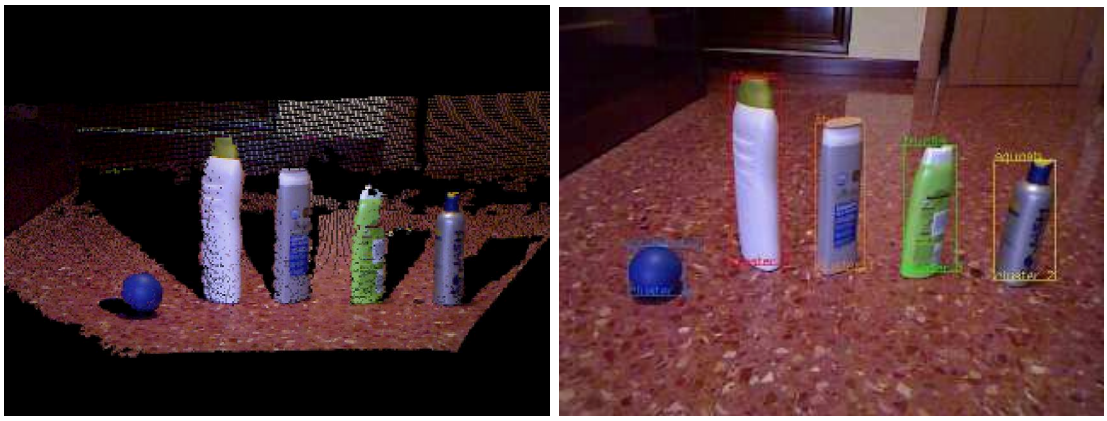

Figura 4.4: Entrada y salida del sistema de reconocimiento. A la izquierda la nube de puntos de entrada, a la derecha una imagen 2D, donde un recuadro envuelve los clusters encontrados y especifica el objeto que ha reconocido.

4.2.1. Selección y evaluación de los objetos candidatos

En esta sección se va a explicar con más detalle el método que sigue el sistema de reconocimiento para seleccionar y posteriormente evaluar una lista de objetos candidatos. Concretamente se va a comentar como se crea la lista de posibles objetos reconocidos (objetos candidatos), la información que almacena a lo largo del proceso y finalmente como se normaliza esa información para decidir que objeto de la lista es el más similar al objeto de test.

Lista de candidatos

La creación de la lista de candidatos es un paso muy importante, pues hay que asegurar que en ella este contenido el objeto al que corresponde el *cluster* que se está reconociendo, aunque debido a ello se introduzcan más candidatos. Los siguientes pasos que evalúan más descriptores ya se encargarán de pulir la decisión. Por tanto, en la lista de candidatos se crea de la siguiente manera: tras la extracción del descriptor VFH en el cluster que estamos evaluando de la imagen de test, y calcular la distancia Chi-Cuadrado entre este cluster y los ejemplos de la base de datos, se eligen los n vecinos más cercanos dentro de la base

de datos. De entre estos vecinos se incluyen en la lista todos los objetos cuya distancia d cumpla estas condiciones:

$$d < 3mindist \quad \text{si } mindist \leq 20$$

$$d < 2mindist \wedge d < 100 \quad \text{si } mindist > 20$$

Siendo $mindist$ la distancia del primer vecino más cercano. Se decidió ampliar la condición de entrada a la lista de candidatos hasta $3mindist$ para casos con $mindist$ muy pequeña para asegurar más la inclusión del objeto al que corresponde el *cluster* a reconocer en la lista de candidatos.

Es muy posible que un mismo objeto de la base de datos corresponda a varios de los vecinos más cercanos que cumplen las condiciones arriba expuestas, pero solo conviene guardar una distancia por objeto en esta lista de candidatos. Por tanto, la distancia (x_i) para el descriptor VFH evaluado del objeto i de la lista de candidatos es la siguiente:

$$x_i = (\sum_{j=1}^{k} \frac{d_i^j}{k}) \cdot (1 - \frac{rep_i}{100})$$

Siendo d_i^j la j-ésima mejor distancia Chi-Cuadrado del objeto i, rep_i el número de veces que el objeto i satisface las condiciones para entrar en la lista de candidatos y k obtenido con la siguiente expresión:

$$k = \begin{cases} 3 & \text{si } rep_i \geq 3 \\ rep_i & \text{si } rep_i < 3 \end{cases}$$

Esta distancia almacenada es la media de hasta las tres mejores distancias, pero se le resta un 1 % a esa distancia por cada vez que se repita el objeto, para premiar la repetición. Todas las constantes numéricas de estas expresiones están tomados empíricamente.

descripción mas detallada de objetos candidatos. Una vez extraídas las características SURF de los elementos de la lista de candidatos, para los que resulten como objetos con textura, porque se encontraron suficientes puntos SURF, se almacena la mejor medida de similitud Sim (Ecuación 2.1) encontrada. En cuanto a los histogramas de color, se almacena la mejor medida de similitud encontrada para cada objeto de la lista. Dicha medida de similitud corresponde con la distancia EMD explicada en la Sección 2.3.1.

Evaluación de los candidatos y elección de la solución

En este punto, la lista de candidatos cuenta con distintas medidas para cada candidato, que representan la probabilidad de que dicho candidato sea el objeto a reconocer. Sin embargo, estas medidas, las distancias o medidas de similitud de cada descriptor, son de distinta naturaleza y orden de magnitud, por lo que es necesario normalizarlas. Se ha optado por una normalización por el máximo, consiguiendo de esta manera para la medida de cada descriptor un ratio entre 0 y 1, donde mayor valor significa mayor probabilidad de que el objeto candidato sea el objeto de test, es decir, se consigue una medida de similitud normalizada. Concretamente, estas son las operaciones que se realizan:

- Para la normalización del descriptor VFH se calcula:

$$simNorm_{vfh} = 1 - \frac{min_{vfh} - x_{vfh}}{max_{vfh}}$$

Siendo x_{vfh} la distancia a normalizar, min_{vfh} la menor distancia encontrada en la lista de candidatos y max_{vfh} la distancia límite que no debía superarse para que un objeto entrara en la lista de candidatos.

- Para la normalización del descriptor SURF se calcula:

$$simNorm_{surf} = \frac{x_{surf}}{max_{surf}}$$

Siendo x_{surf} la medida de similitud a normalizar y max_{surf} la mejor similitud encontrada en la lista de candidatos.

- Para la normalización de los histogramas de color se calcula:

$$maxdist = \begin{cases} max_{color} & \text{si } max_{color} \leq 7{,}5 \\ 7{,}5 & \text{si } max_{color} > 7{,}5 \end{cases}$$

$$simNorm_{color} = 1 - \frac{x_{color}}{maxdist}$$

Siendo x_{color} la medida de similitud a normalizar y max_{color} la mayor distancia EMD encontrada en la lista de candidatos. En este caso no se normaliza por el máximo absoluto, sino que si max_{color} supera cierto umbral, se considera ese umbral como el máximo. Esto se decidió así porque en la práctica, valores más allá de 7.5 reflejan una coincidencia nula entre los objetos. El valor 7.5 es la mitad del valor máximo que la distancia EMD puede devolver, que es 15, ya que los histogramas manejados son de 16 componentes.

Selección final del objeto. Una vez las medidas están normalizadas, simplemente se calcula la media y se escoge como elección aquel objeto de la lista que tenga una media más alta. Sin embargo, en el caso de que el objeto tenga textura y por tanto se tenga en cuenta la valoración de los descriptores SURF, la valoración del histograma de color pierde la mitad de su peso a la hora de hacer la media. Esta decisión se tomó tras comprobar empíricamente que los objetos con textura, susceptibles de tener muchos colores, no presentaban resultados excesivamente buenos a la medida de similitud propuesta.

4.3. Rendimiento del sistema

Para poder determinar el rendimiento del sistema se llevaron a cabo una serie de experimentos, detallados en el anexo D, que permiten averiguar como se comporta dependiendo de la naturaleza de los objetos a reconocer, la combinación de descriptores usados y si los objetos sufren oclusiones o no. Como medida de rendimiento, en todos los experimentos se han utilizado los coeficientes *precision* y *recall*, medida muy habitual para evaluar reconocedores, y que son definidos de la siguiente manera:

$$precision = \frac{tp}{tp+fp} \quad recall = \frac{tp}{tp+fn} \tag{4.1}$$

Siendo tp el número de verdaderos positivos, fp el de falsos positivos y fn el de falsos negativos.

En esta sección se van a presentar los dos experimentos que mejor resumen el rendimiento del sistema. En el anexo D hay experimentos adicionales que evalúan las combinaciones de descriptores según la forma de los objetos a reconocer, (se intenta reconocer objetos con forma diferente, parecida o igual a otros objetos de la base de datos), y si tienen textura o no. También se evalúa el sistema cuando se intentan reconocer objetos que sufren oclusiones.

Experimento con objetos variados y personalmente fotografiados

En este experimento se pretende averiguar el rendimiento del sistema cuando se enfrenta al reconocimiento de objetos de distinta naturaleza, todo tipo de formas y teniendo o no textura. La base de datos durante la ejecución de este experimento contenía 42 objetos, (aproximadamente 24 imágenes por objeto para crear el modelo), la mayoría de ellos fotografiados personalmente, aunque se incorporaron varios modelos procedentes de un *dataset* público [23] con objeto de que no todos los modelos provengan de la misma fuente. En el Anexo F aparece información sobre los objetos capturados, así como de los objetos del *dataset* público citado. Se han reconocido 118 objetos repartidos en 28 fotos de test. La Tabla 4.1 muestra los resultados.

descriptores	**Tiempo de ejecución (seg)**	**Precision**	**Recall**
forma	207.523	0.602	1.0
forma+color	208.311	0.576	1.0
forma+surf	214.181	0.737	1.0
forma+surf+color	214.821	0.805	1.0
surf+color	228.684	0.475	1.0

Tabla 4.1: Resultados de reconocimiento sobre tests objetos de diferentes tipos.

Los resultados muestran un buen rendimiento general del sistema cuando se utilizan todos los descriptores posibles, pues dependiendo de la naturaleza del objeto, será un descriptor u otro el que discrimine con más garantías el objeto del que se trata. Cabe señalar que el peor resultado es el de la combinación de descriptores que no utilizan información 3D para reconocer, no solo en aciertos, sino también en tiempo de ejecución. Esto era de esperar ya que al no existir para ese caso la lista de candidatos creada de acuerdo al descriptor de forma, se deben comparar SURF e histogramas de color con todos los objetos de la base de datos. Esto muestra las ventajas de las técnicas de reconocimiento 3D, nótese que los resultados arrojados por la combinación "surf+color" son muy bajos, pero aún lo serían más si realmente no se hubiera segmentado la región de interés utilizado información 3D.

Comparación con otro sistema de reconocimiento

En este experimento se quiere comparar el sistema de reconocimiento desarrollado en este proyecto por el propuesto recientemente en [23], que también utiliza información 3D.

Para ello se va a utilizar el mismo *dataset*, aunque en este caso un subconjunto del mismo debido a su gran tamaño, y se va a replicar uno de los experimentos llevados a cabo en el sistema citado. El experimento consiste en almacenar como modelo de un objeto imágenes tomadas desde dos puntos de vista diferentes, y como imágenes de test las de ese mismo objeto pero desde un punto de vista diferente. Cabe destacar que el sistema citado almacena descriptores extraídos del orden de 250 imágenes por modelo, mientras que el sistema propuesto aquí almacena solo descriptores extraídos de 24 imágenes por modelo. Para este experimento se ha mantenido una base de datos con 60 objetos diferentes, y se han reconocido 448 objetos, uno por imagen. En la Figura 4.5 se muestra la gráfica *precision-recall* de los dos sistemas.

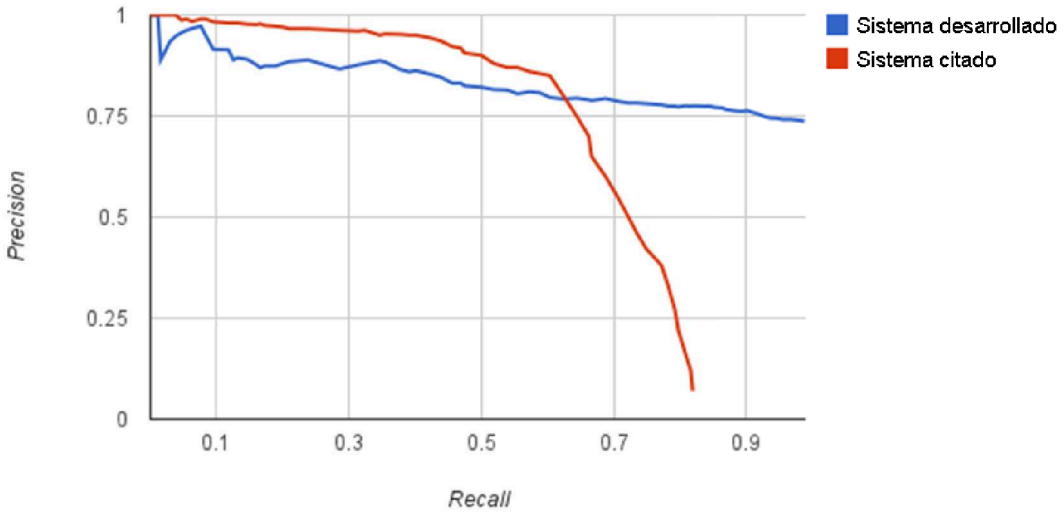

Figura 4.5: Curvas Precision-Recall de los dos sistemas.

La gráfica muestra que el sistema desarrollado no alcanza el rendimiento del sistema propuesto en [23], aunque la diferencia no es demasiado elevada, y sin embargo muestra la ventaja de ser mucho más estable en cuanto a fallos de tipo falso negativo, es decir, el sistema raramente reconoce un objeto donde realmente no lo hay. También debe tenerse en cuenta que el número de imágenes por modelo en la base de datos es alrededor de 10 veces menor en el sistema desarrollado en este proyecto.

5. Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones extraídas tras la realización de este proyecto, así como propuestas de trabajo futuro.

5.1. Conclusiones

La aparición del sensor *kinect* en el mercado ha supuesto poco menos que una revolución en el mundo de la visión por computador. Su capacidad para extraer en tiempo real el mapa de profundidad de una escena ha impulsado varias líneas de investigación que siguen muy activas. El deseo de formar parte de dicho trabajo unido con el interés por la visión por computador han sido motivación más que suficiente para llevar a cabo el proyecto.

El objetivo principal del proyecto ha sido diseñar e implementar un sistema de reconocimiento con información 3D que demuestre el beneficio que aportan dichos sensores frente a sistemas de reconocimiento convencionales, objetivo que se ha cumplido con unos resultados satisfactorios. Para lograrlo, se estudiaron diversas librerías para el manejo de *kinect*, así como literatura relacionada con reconocimiento con imágenes convencionales necesaria para construir un sistema inicial que evidenciara los puntos más débiles y más fuertes del reconocimiento que se ha denominado como 2D. Posteriormente se estudió literatura relacionada con técnicas de reconocimiento utilizando información de profundidad de cada píxel de la imagen (que se ha denominado como reconocimiento 3D). Finalmente se diseñó e implementó un sistema basado en la combinación de ambas técnicas, que mejorara lo implementado con anterioridad en la literatura estudiada.

Los puntos donde más diferencia ha marcado el uso de técnicas 3D han sido la segmentación de la imagen a analizar y la jerarquización en las comparaciones entre descriptores de imágenes. Sin las técnicas 3D el proceso de segmentación no alcanzaría los niveles de rendimiento obtenidos, ya que sería mucho mas costoso eliminar el plano de fondo correspondiente al suelo o una mesa, agrupar los píxeles de la imagen en *clusters* que parezcan pertenecer a un objeto e incluso descartar las partes de la escena demasiado lejanas como para ser de interés. En cuanto a la jerarquización del sistema de evaluación de similitud, esta ha sido posible gracias al descriptor de forma 3D, ya que es aplicable a todos los objetos, todos tienen forma, y es lo suficientemente discriminante como para poder descartar rápidamente la mayoría de objetos de la base de datos, dejando como candidatos un pequeño porcentaje del total. Esto consigue un funcionamiento mucho más rápido y que el coste del reconocimiento crezca de forma logarítmica respecto al tamaño de la base de datos, pues la búsqueda se realiza en un árbol.

5.2. Trabajo futuro

Tras la realización de este proyecto, ase han abierto nuevas líneas de trabajo que servirían para aumentar o complementar al sistema de reconocimiento diseñado.

- Segmentación de las regiones de interés mezclando técnicas 2D y 3D. La segmentación que realiza el sistema propuesto falla en un caso, cuando varios objetos se encuentran demasiado juntos como para que la clusterización euclídea los considere como cuerpos distintos. En esos casos los objetos serían considerados como un *cluster* a evaluar, con lo que con toda probabilidad se fallaría en reconocerlos todos. Se podría combinar la clusterización de puntos 3D realizada con técnicas para segmentar imágenes basadas en información 2D, como los contornos que se pueden extraer en una imagen, para intentar decidir si alguno de los clusters obtenidos contiene en realidad más de un objeto.

- Integración con un robot. Este proyecto podría ser de utilidad para aplicaciones de robótica, ya que consigue resultados fiables en tiempos aceptables para ejecutar en sistemas con restricciones temporales. Para ejecutar este sistema de reconocimiento dentro de un robot manipulador que deba buscar y coger objetos, habría que adaptar el sistema para funcionar dentro del sistema operativo ROS[1], especialmente creado para trabajar con robots.

[1] http://www.ros.org/wiki/

Bibliografía

[1] Radu Bogdan Rusu and Steve Cousins, *3D is here: Point Cloud Library (PCL)*. In IEEE International Conference on Robotics and Automation (ICRA), 2011.

[2] Marius Muja and David G. Lowe, *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*. in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009.

[3] Richard Szeliski, *Computer Vision: Algorithms and Applications*. pages 654–734, 2010.

[4] Ethan Rublee, Vicent Rabaud, Kurt Konolige, Gary Bradski, *ORB: an efficient alternative to SIFT or SURF*. International Conference on Computer Vision, 2011.

[5] E. Rosten and T. Drummond, *Machine learning for highspeed corner detection*. In European Conference on Computer Vision, volume 1, 2006.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, *Brief: Binary robust independent elementary features*. In European Conference on Computer Vision, 2010.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, *Surf: Speeded up robust features*. In European Conf. on Computer Vision, 2006.

[8] David Lowe, *Object recognition from local scale-invariant features*. Proceedings of the International Conference on Computer Vision, pages 1150–1157, 1999.

[9] Jon Louis Bentley, *Multidimensional binary search trees used for associative searching*. In Communications of the ACM, 1975.

[10] Martin A. Fischler and Robert C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, Volume 24 Issue 6, 1981.

[11] D. E. Knuth, *The Art of Computer Programming*. Addison-Wesley, 1973.

[12] Stuart LLoyd, *Least square quantization in PCM*. IEEE Transactions on Information Theory, 1982.

[13] Y. Rubner, C. Tomasi, and L. J. Guibas, *The earth mover's distance as a metric for image retrieval*. In International Journal of Computer Vision, 2000.

[14] S. T. Rachev, *The Monge-Kantorovich mass transferene problema and its stochastic applications*. Theory of probability and its applications XXIX(4) pages 647–676, 1984.

[15] Javier Arroyo Espallargas, *Guía turística de monumentos basada en reconocimiento visual con un móvil*. PFC Universidad de Zaragoza, 2012.

[16] X. Ren and J. Malik, *Learning a classification model for segmentation*. In Proc. 9th Int. Conf. Computer Vision, volume 1, pages 10–17, 2003.

[17] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, *Efficient graph-based image segmentation*. In International Journal of Computer Vision, Volume 59, Number 2, 2004.

[18] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux and John Hsu, *Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram*. Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[19] Marius Muja, Radu Bogdan Rusu, Gary Bradski and David G. Lowe, *REIN - A Fast, Robust, Scalable REcognition INfrastructure*. In IEEE International Conference on Robotics and Automation (ICRA), 2011.

[20] Radu Bogdan Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD Thesis Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[21] Kevin Lai, Liefeng Bo, Xiaofeng Ren and Dieter Fox, *A Scalable Tree-based Approach for Joint Object and Pose Recognition*. In AAAI Conference on Artificial Intelligence 2011.

[22] Kevin Lai, Liefeng Bo, Xiaofeng Ren and Dieter Fox, *Object Recognition with Hierarchical Kernel Descriptors*. In IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2011.

[23] Kevin Lai, Liefeng Bo, Xiaofeng Ren and Dieter Fox, *A Large-Scale Hierarchical Multi-View RGB-D Object Dataset*. In IEEE International Conference on Robotics and Automation (ICRA) 2011.