UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

RECONOCIMIENTO DE OBJETOS EMPLEANDO TÉCNICAS PARA EL PROCESAMIENTO DE NUBES DE PUNTOS APLICADO A ROBOTS MÓVILES

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
LUIS ARTURO GONZÁLEZ JUÁREZ

TUTOR PRINCIPAL
DR. JESUS SAVAGE CARMONA, FACULTAD DE INGENIERÍA

CIUDAD UNIVERSITARIA, CD. MX. NOVIEMBRE 2016

JURADO ASIGNADO:

Presidente:	DR. GARCÍA UGALDE FRANCISCO
Secretario:	DR. ESCALANTE RAMÍREZ BORIS
Vocal:	DR. SAVAGE CARMONA JESÚS
1 er. Suplente:	DRA. MARTÍNEZ PÉREZ MARÍA ELENA
2 d o. Suplente:	M.I. ESCOBAR SALGUERO LARRY

Lugar donde se realizó la tesis: FACULTAD DE INGENIERÍA - UNAM

TUTOR DE TESIS:

DR. SAVAGE CARMONA JESÚS

FIRMA

Agradecimientos

. . . a mis familia, por su apoyo incondicional.

. . . al Dr. Jesus Savage Carmona, por su tiempo y sus consejos.

. . . a CONACYT, por los recursos económicos otorgados para la realización de esta maestría, así como al proyecto del PAPIIT-UNAM IG1100915, por el apoyo para el desarrollo de esta tesis.

Resumen

En el presente trabajo se propone un sistema para la detección y el reconocimiento de objetos el cual se encuentra basado en la obtención de sus características visuales y geométricas, así como de la información sobre la distribución espacial de éstas y sobre sus dimensiones. Para ello, se propone utilizar las nubes de puntos de los objetos generadas mediante el uso de sensores *RGB-D*.

Se utilizan dentro del sistema representaciones estadísticas de los objetos creadas a partir de las características anteriormente mencionadas basadas en distribuciones de frecuencias y modelos ocultos de Markov. Además, se presentan métodos para la segmentación y el filtrado de nubes de puntos ampliamente utilizados con la intención de enfocar el desarrollo del sistema a su aplicación en robots móviles.

Índice general

Índice de figuras

Capítulo 1

Introducción

La visión humana es uno de los sistemas más complejos que existen ya que se ocupa de capturar aproximadamente el 80 % de la información que recibe el cerebro, por lo que es considerado el más vital de los sentidos. La teoría de la *visión* del neurocientífico David Marr dictamina que ésta no es más que una tarea de procesamiento de información que tiene la finalidad de capturar y representar aspectos del mundo que nos sean de utilidad. Además propone que ésta puede ser dividida en tres niveles: (I) teoría computacional, (II) representación y algoritmos e (III) implementación física [1]. Al día de hoy, estas ideas han tenido un gran impacto en la investigación relacionada a la visión humana y computacional.

El objetivo de la visión computacional es desarrollar sistemas capaces de interpretar el contenido de escenas naturales, dotando a las computadoras con la capacidad de poder simular la visión humana. Es posible ver los avances en esta área en diversos ambientes, desde plantas manufactureras a quirófanos e incluso en la superficie de Marte [2]. Sin embargo, la visión computacional tiene aún muchos problemas por resolver. Esto queda en evidencia al comparar la facilidad con la que nuestro sistema visual nos permite realizar tareas que para esta disciplina pudieran ser muy complejas.

Uno de los los problemas más desafiantes dentro del área visión computacional es el reconocimiento de objetos. A pesar de que las personas reconocemos una gran cantidad de ellos sin mucho esfuerzo, no es el mismo caso para los sistemas de visión computacional. Esto se debe principalmente al cambio de condiciones a las que son sometidos los objetos en diferentes capturas como la iluminación, la orientación, la perspectiva u oclusiones par-

ciales que se pudieran presentar. Además, es difícil obtener una descripción matemática detallada de los procesos involucrados en la visión y percepción humana, lo cual obstaculiza su utilidad en el desarrollo de algoritmos [3]. Sin embargo, numerosos métodos se han creado a lo largo de los años que resuelven parcialmente este problema.

La visión computacional es ahora un componente esencial para muchas aplicaciones en robots móviles. Al proporcionarles este tipo de percepción sensorial, se les brinda también la capacidad de entender el medio en el que se desplazan y de realizar tareas cada vez más complejas. La posibilidad de utilizarlos como robots de servicio para la asistencia en hogares, hospitales u oficinas no es una idea descabellada, ya que el desarrollo de la robótica ha sido muy similar al que tuvieron las computadoras hace casi cuarenta años cuando resultaba impensable la existencia de ordenadores personales.

1.1. Objetivos

Diseñar un sistema para la detección y el reconocimiento de objetos domésticos utilizando información proporcionada por sensores *RGB-D* de bajo costo para su uso en aplicaciones con robots de servicio, **teniendo como objetivos particulares los siguientes**:

- Capturar, segmentar y filtrar la información sobre los objetos de interés que se encuentren en una escena.
- Obtener representaciones adecuadas de los objetos para su uso en combinación con los clasificadores propuestos.
- Combinar las características geométricas y visuales proporcionadas por las nubes de puntos para el uso de sólo un clasificador.
- Utilizar un método de verificación para determinar los *falsos positivos* en el proceso de reconocimiento.

1.2. Hipótesis

El uso de información visual y geométrica de los objetos, así como la distribución espacial de ésta información sobre sus dimensiones mejoran en conjunto los resultados para un sistema de reconocimiento de objetos en comparación con el uso de esta información de manera independiente.

1.3. Justificación

En el laboratorio de bio-robótica de la Universidad Nacional Autónoma de México (UNAM) se desarrolla el proyecto *Justina* que consiste en la creación de un robot que sea capaz de asistir a las personas en tareas del hogar. En él se requiere la integración de distintas disciplinas de la ingeniería como son la inteligencia artificial, el control, la visión computacional, entre otras. Éste se encuentra a cargo del Dr. Jesús Savage Carmona, quien promueve la robótica en México desde hace más de diez años. En la Figura 1.1 se muestra una fotografía de una de las versiones de *Justina*.

De la necesidad de que los robots realicen tareas cada vez más complejas y de manera más eficiente surgen proyectos como *Robot Competitions Kick Innovation in Cognitive Systems and Robotics* (RoCKIn) [4] el cual es financiado por la Unión Europea y tiene como objetivo promover el progreso científico e innovación en el campo de la robótica. Se han puesto en práctica competencias entre equipos de trabajo afiliados a diferentes universidades para el desarrollo de distintos tipos de robots, entre ellos los que entran en la categoría de servicio doméstico como *Justina*. En la competencia celebrada en Lisboa, Portugal en noviembre del 2015 se logró obtener un segundo lugar con este prototipo en la prueba de *Object Perception*, la cual evalúa la capacidad de los robots para identificar y reconocer objetos.

En la prueba anteriormente mencionada, el uso de cámaras *RGB-D* fue fundamental para la obtención de dichos resultados. Actualmente, estos sensores se han convertido en un parte esencial de los sistemas de visión para aplicaciones en robótica ya que, a pesar de ser concebidas originalmente para emplearse en videojuegos, su gran potencial y bajo costo las llevó a ser utilizadas como herramienta de investigación y desarrollo de proyectos. Además, la existencia de librerías de software para el procesamiento de imágenes y de nubes de puntos facilitan el manejo de los datos proporcionados por estos sensores.

Debido al buen desempeño que tuvo el método para el reconocimiento de objetos implementado en el sistema de visión de *Justina*, se seguirá trabajando con algunas de las ideas utilizadas previamente. Además, se buscará utilizar métodos estadísticos para el reconocimiento de patrones que se han empleado exitosamente en el campo del reconocimiento de voz, los cuales tienen la virtud de encontrarse bien fundamentados, matemáticamente hablando. Cabe destacar que el empleo de estos métodos en el campo de la vi-

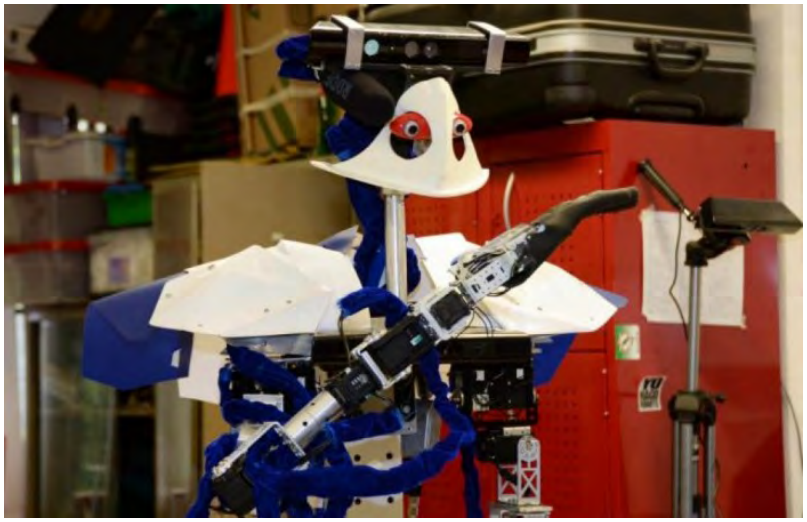

Figura 1.1: Robot *Justina*.

sión computacional comenzó a investigarse hace algunos años, obteniéndose resultados prometedores [5]. Existen procedimientos para el reconocimiento de objetos en 2D que utilizan estos métodos, sin embargo, su uso en el contexto del reconocimiento de objetos en 3D ha sido poco investigado y se realizó cuando aún no ocurría el auge de los sensores *RGB-D*.

1.4. Estructura de la tesis

El presente trabajo de tesis se encuentra dividido en seis capítulos: en el Capítulo 2 se presentan algunos de los antecedentes que existen relacionados con la tarea del reconocimiento de objetos; en el Capítulo 3 se expondrán algunos de los conceptos que dan sustento al desarrollo de este trabajo; en el capítulo 4 se describe el sistema propuesto, el cual se encuentra compuesto por los procesos de entrenamiento y reconocimiento; en el Capítulo 5 se presenta la configuración del sistema en las pruebas y los resultados experimentales obtenidos con un conjunto de objetos; en el Capítulo 6 se presentan las conclusiones con base en estos resultados e ideas para continuar con el trabajo en el futuro que pudieran permitir mejoras en el sistema.

Capítulo 2

Antecedentes

La finalidad de un sistema de reconocimiento de objetos es identificarlos en una imagen o secuencia de video a partir de modelos que se conocen a priori [6]. Una gran cantidad de algoritmos y sistemas han sido propuestos para lograrlo, sin embargo, aún no existe una solución general para reconocer cualquier tipo de objeto bajo diferentes condiciones, por lo que se considera un problema abierto.

De acuerdo a [7], un sistema de reconocimiento de objetos debe contener los siguientes componentes (ver Figura 2.1):

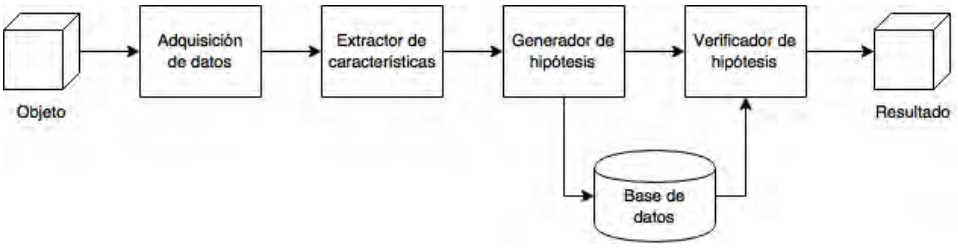

Figura 2.1: Diagrama de los componentes que conforman un sistema de reconocimiento de objetos.

- Base de datos: es la parte del sistema donde se almacenan los modelos conocidos. La información almacenada depende del método utilizado para la etapa de reconocimiento.
- Adquisición de datos: es la etapa en donde la información analógica

del mundo físico se obtiene a través de un transductor y se convierte a formato digital.

- Extractor de características: realiza operaciones sobre las imágenes o nubes de puntos para detectar ***características***, las cuales son atributos que describen a los objetos (ej. tamaño, color, geometría, etc.).
- Generador de hipótesis: asigna probabilidades a los objetos presentes en la escena, de acuerdo a las *características* encontradas, con el objetivo de reducir el espacio de búsqueda.
- Verificador de hipótesis: utiliza los modelos de la base de datos para verificar las hipótesis del paso anterior. Ésta y la etapa de generación varían en importancia dependiendo del enfoque utilizado.

El sistema reconoce al objeto en la escena cómo el almacenado en la base de datos cuyo modelo genera la mayor probabilidad a partir de las características observadas.

Es posible dividir la tarea del reconocimiento de objetos en dos enfoques [8]: reconocimiento bidimensional (2D) y reconocimiento tridimensional (3D). La diferencia entre éstos radica esencialmente en que el primero utiliza imágenes *RGB* como entrada de datos, mientras que el segundo se emplea imágenes de profundidad o nubes de puntos.

2.1. Reconocimiento en 2D

Los algoritmos para el reconocimiento de objetos se dividen históricamente en dos clases: métodos basados en modelos geométricos y métodos basados en apariencia [9] . Hay algunos que consideran una tercer clase en la que se incluyen los modelos basados en *características* [10].

Los métodos basados en modelos geométricos utilizan el modelo tridimensional de un objeto y su clasificación se realiza buscando la transformación que mejor se ajuste entre el modelo y la vista observada. Los modelos contienen información detallada del objeto como su estructura y la relación espacial entre sus partes. Sin embargo, omite el uso de propiedades como el color y la textura debido a que se enfoca en describir la forma en 3D.

Los métodos basados en apariencia no requieren de un modelo tridimensional debido a que el reconocimiento se realiza comparando las imágenes de

entrada con un conjunto exhaustivo de imágenes de entrenamiento, las cuales se obtienen desde diferentes puntos de vista y cambios en la iluminación. Entonces, obteniendo una nueva toma de un objeto, éste será reconocido asociándola a una vista de un modelo previamente entrenado. Estos métodos son atractivos debido a que no requieren una búsqueda de *características* ni la búsqueda de sus correspondencias, sin embargo, son poco robustos a oclusiones y requieren una previa segmentación. Los métodos de comparación de histogramas entran en esta categoría y fueron inicialmente, propuestos por Swain [11], quién plantea representar a un objeto mediante su histograma de color y compararlo con los modelos previamente almacenados mediante alguna métrica.

Los métodos basados en *características* tienen como objetivo encontrar puntos de interés en la escena que sean invariantes a escala, iluminación y transformaciones que se utilizan para generar vectores conocidos como ***descriptores***, los cuales se encargan de capturan la información de una imagen en vecindades locales alrededor de estos puntos. El reconocimiento de objetos con estos métodos se realiza haciendo una comparación entre los *descriptores* obtenidos en la etapa de entrenamiento y los que se detectan en la escena con alguna medida de similitud. Entre los *descriptores* más populares se encuentran SIFT [12] y SURF [13], ambos basados en el cálculo de histogramas de gradientes.

2.2. Reconocimiento en 3D

Debido a los avances recientes en la tecnología de sensores de profundidad, ha crecido el interés por parte de la comunidad científica por utilizar información tridimensional para la tarea del reconocimiento de objetos [14]. Los métodos de reconocimiento de objetos con información 3D se pueden dividir en aquellos que se encuentran basados en el uso de (I) características locales y (II) características globales [7]. El primer tipo trata de establecer correspondencias entre puntos de la escena y los modelos almacenados, usualmente mediante el uso de *descriptores*, los cuales se obtienen utilizando información de las vecindades de los puntos. Estos *descriptores* se encargan de codificar la geometría local alrededor de cada punto, por lo que no tienen la noción de lo que es un objeto. Algunos de los más populares son *Fast Point Feature Histograms* (FPFH) [15], *Signature of Histograms of Orientations* (SHOT) [16] y *Radius-Based Surface Descriptor* (*RSD*) [17]. Cabe destacar que de los *descriptores* presentados, solamente SHOT se ha desa-

rrollado para incluir tanto información geométrica como visual. El segundo tipo utiliza *descriptores* globales y busca codificar la geometría del objeto por lo que no se calculan para sus puntos individuales, sino para todo el conjunto. Además, tienen la cualidad de representar un objeto con un solo vector. Algunos ejemplos de estos *descriptores* son *Viewpoint Feature Histogram* (VFH) [18] y *Ensemble of Shape Functions* (ESF) [19].

2.3. Reconocimiento utilizando métodos estadísticos

El proceso para el reconocimiento de objetos depende en gran medida de los modelos utilizados y la función de similitud empleada. El *emparejamiento de características* es usualmente un método robusto a oclusiones, sin embargo, suele ser lento debido a que requiere mucho tiempo de procesamiento para la búsqueda de las mejores correspondencias y de la transformación rígida para evitar correspondencias erróneas. Por esto, se han propuesto métodos alternativos que evitan la búsqueda de correspondencias y describen a los objetos estadísticamente [20].

En [8], se justifica el uso de métodos probabilísticos dentro de la visión computacional con las siguientes razones:

1. El éxito de métodos de reconocimiento de patrones, como los usados en sistemas de reconocimiento de voz, se basan en métodos estadísticos.

2. Su capacidad de lidiar con ambigüedades generadas en diferentes capturas de una misma escena.

3. La disponibilidad de métodos matemáticos muy estudiados.

Dentro de un marco probabilístico, el reconocimiento de objetos se puede realizar aplicando la regla de decisión de Bayes, obteniendo las probabilidades a posteriori mediante el uso de los modelos estadísticos de los objetos y las características observadas. Si se tienen clases de objetos$\Omega_1, \Omega_2, \ldots, \Omega_n$ y se observa un vector de características $\overrightarrow{c}$ obtenido de la observación de un objeto, se decide que este objeto pertenece a la claseΩ_k si$\Omega_k = \underset{\Omega_k}{\operatorname{argmax}}(P(\Omega_i)|\overrightarrow{c})$.

2.3.1. Reconocimiento utilizando modelos ocultos de Márkov

Mientras que muchos sistemas probabilísticos para el reconocimiento se basan en diferentes esquemas de la teoría bayesiana de decisión, existen otros métodos que se basan en el uso de modelos estadísticos conocidos como *modelos ocultos de Márkov* (HMM, por sus siglas en inglés: *Hidden Markov Models*), los cuales se explicarán con más detalle en la Sección 3.2.2. Estas herramientas han demostrado su efectividad en el campo de la visión computacional en aplicaciones como el reconocimiento de gestos, de comportamientos, entre otros [5]. Otras aplicaciones que aprovechan la relación espacial entre variables en lugar de una temporal es el reconocimiento de rostros, de texto escrito y de objetos.

Un intento temprano de realizar el reconocimiento de objetos utilizando esta herramienta puede ser encontrado en [21], donde se calculan características invariantes afines del contorno y el área de los objetos en imágenes, las cuales son utilizadas para alimentar un HMM. En [22] se propone un método en el cual se utilizan coeficientes de curvaturas de las siluetas obtenidas de diferentes vistas del objeto. En una investigación posterior [23], el mismo autor emplea vectores de coeficientes de ondeleta obtenidos a partir de la división de cada vista de un objeto en sub-imágenes. En [24] se utilizan una serie de características tridimensionales en imágenes de profundidad, como el tipo de superficie y los momentos, en combinación con los HMM y redes neuronales para la tarea del reconocimiento de objetos en 3D. Todos estos métodos se basan en el cálculo de *descriptores* que se pueden ordenar en secuencias, siendo adecuado su uso con HMM, y además se basan en la generación de un modelo por vista. En [25], otro algoritmo que utiliza siluetas es propuesto pero, a diferencia del presentado anteriormente, calcula un *descriptor* global de ellas logrando secuencialidad mediante el ordenamiento de sus vistas. A pesar de la existencia de tempranas sugerencias en [21] del empleo de curvaturas tridimensionales en conjunto con los HMM, ninguno de los métodos anteriores hace uso de ellas ni de otra información relevante como el color.

Capítulo 3

Marco Teórico

3.1. Representaciones

Existen diversas representaciones que se pueden realizar de una escena en el mundo real para su procesamiento digital las cuales pueden encontrarse en forma de datos bidimensionales o tridimensionales, esto dependiendo de los sensores que se utilicen para capturar la escena. En esta sección se presentan algunas de las principales representaciones que se utilizan con dicho fin.

3.1.1. Imagen digital

Una imagen se define como una función bidimensional $f(x, y)$, donde x y y representan coordenadas espaciales en un plano y la amplitud en cualquier par de coordenadas representa comúnmente valores de intensidad, en el caso de imágenes RGB o en escala de grises, o valores de profundidad, en el caso de mapas de profundidad. Para poder realizar operaciones computacionales en imágenes, es necesario digitalizarlas mediante un muestreo del espacio y la cuantificación de los valores. La unidad básica de una imagen digital es el ***píxel*** y se encuentra definida por sus coordenadas discretas y su valor cuantificado.

3.1.2. Nube de puntos

Una nube de puntos es una estructura de datos utilizada para representar coordenadas geométricas de una colección de puntos en tres dimensiones. Además de su posición en el espacio, estos puntos pueden incluir información

sobre otras propiedades como su color y la orientación de su normal. Las nubes de puntos son utilizadas comúnmente para representar la superficie externa de objetos que se suelen adquirir mediante el uso de escáneres 3D. Se les da el nombre de "nubes" debido a que sus puntos carecen de conectividad y, cuando son visualizados, da la impresión de que se encuentran flotando.

Es posible generar una nube de puntos a través de un mapa de profundidad conociendo los parámetros intrínsecos de la cámara utilizada, los cuales se pueden obtener mediante un proceso de calibración. Este proceso se realiza utilizando lo que se conoce como modelo de cámara estenopeica en el que una escena es creada proyectando un conjunto de puntos tridimensionales en el plano de la imagen a través de una transformación, como se muestra en la Figura 3.1. Una vez determinada la coordenada en Z, la obtención de las coordenadas en X y en Y se realizará mediante las siguientes ecuaciones de proyección:

$$X = \frac{(x - c_x)Z}{f_x}, \quad Y = \frac{(y - c_y)Z}{f_y} \tag{3.1}$$

Z: Profundidad

x, y: Coordenadas del punto en la imagen

c_x, c_y: Centro óptico

f_x, f_y: Distancia focal

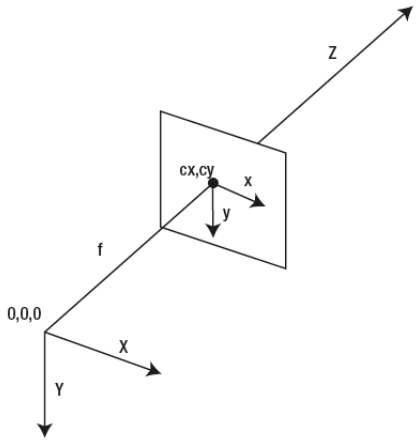

Figura 3.1: Modelo de cámara estenopeica. Imagen extraída de [26]

3.1.3. Volumen digital

Un volumen digital se describe como un arreglo tridimensional de elementos de volumen llamados ***vóxeles***. Así como un los *píxeles* son generados a partir de la digitalización de una imagen, los *vóxeles* son el resultado de la división del espacio continuo tridimensional en una malla regular, tal como se muestra en la Figura 3.2. Sin embargo, como los *píxeles*, los *vóxeles* no contienen información acerca de su posición en el espacio, sino que esta se deduce a partir de la posición relativa de los *vóxeles* circundantes.

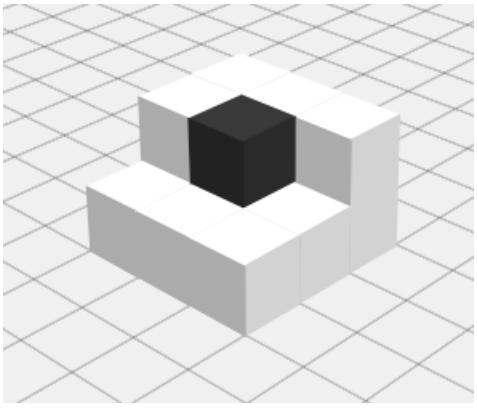

Figura 3.2: Conjunto de *vóxeles* apilados con un vóxel sombreado

3.2. Clasificadores

En un sistema de reconocimiento de patrones, el clasificador tiene la tarea de decidir a qué categoría, de un conjunto finito y predefinido de categorías, pertenece un patrón de clase desconocida. En el caso del reconocimiento de objetos, el clasificador se utiliza para decidir cuáles de los objetos presentes en una escena pueden asociarse a modelos previamente entrenados.

3.2.1. K-Vecinos más cercanos

El *método de los k-Vecinos más cercanos* (k-NN, por sus siglas en inglés: *k-Nearest Neighbors*) es un algorítmo que consiste en clasificar nueva información utilizando datos de entrenamiento en base a una medida de similitud como las funciones de distancia.

Los datos de entrenamiento consisten en vectores en un espacio característico p-dimensional cuyas entradas contienen valores de atributos y pertenecen a una clase $C = c_1, c_2, ..., c_q$. Un vector de entrada $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ es asignado a una de las clases de C si ésta es la clase más frecuente entre los k vectores de entrenamiento más cercanos. La distancia euclidiana entre el vector de entrada x_i y un vector de entrenamiento x_j se presenta en la Ecuación 3.2:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p}(x_{ir} - x_{jr})^2} \tag{3.2}$$

Cuando $k = 1$, se le conoce a lo anterior como *método del vecino más cercano* (1-NN). El vector de entrada x_i es clasificado como perteneciente a la clase del vector de entrenamiento x_l si se cumple la siguiente desigualdad:

$$d(x_i, x_l) < d(x_i, x_j) \quad \forall j, \quad j \neq l \tag{3.3}$$

Un concepto importante relacionado con 1-NN son los *diagramas de Voronoi*, los cuales se encargan de descomponer el espacio en regiones de tal forma que si el vector de entrada x_i se encuentra en la región perteneciente al vector x_j, x_i se encuentra más cerca a x_j que ha otro vector de entrenamiento. La representación visual de 1-NN para el caso bidimensional proporciona implicitamente un *diagrama de Voronoi* de segundo orden como el que se muestra en la Figura 3.3.

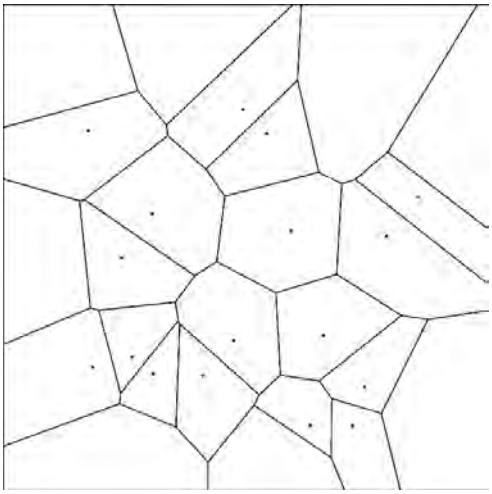

Figura 3.3: *Diagrama de Voronoi* para el caso en el que se tiene un espacio característico bidimensional con $k = 1$.

3.2.1.1. Árboles kd

Un *árbol kd* es una estructura de datos que sirve para organizar puntos en un espacio k-dimensional, los cuales son muy utilizados para la búsqueda de los vecinos más cercanos. Los *árboles kd* son binarios, lo que quiere decir que cada nodo padre tendrá dos hijos, los cuales se pueden definir como el derecho y el izquierdo. Además, cada nivel divide el espacio en una dimensión específica.

En una nube de puntos tridimensional, la construcción de un *árbol kd* sería como sigue: se comienza utilizando el punto mediana para después dividir en dos conjuntos el resto de los puntos basados en la primera dimensión que en este caso se considerará la que corresponde al eje X, por lo tanto, los puntos con un valor menor de la mediana en este eje se colocan a la izquierda del *árbol kd* y aquellos con un valor mayor a la derecha. En el segundo nivel, la división se realiza sobre el eje Y con el mismo criterio que en el paso anterior. El eje Z se utiliza en el tercer nivel y en el cuarto, se vuelve a utilizar el eje X, repitiendo el proceso. En la Figura 3.4 se presenta el caso bidimensional para un mejor entendimiento.

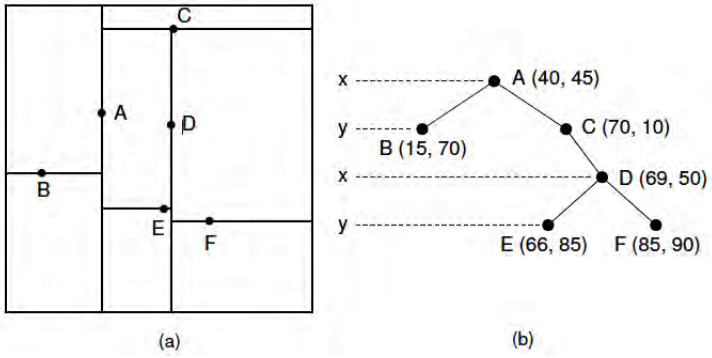

Figura 3.4: Ejemplo de un *árbol kd* (a) Descomposición del espacio para una región de 128x128 unidades con siete puntos (b) *árbol kd* resultante para la región presentada en (a). Imagen extraída de [27]

3.2.2. Modelos ocultos de Márkov

Las *máquinas de estado finito* son modelos matemáticos que producen una salida como respuesta a una entrada, los cuales se encuentran compuestos por un conjunto de estados y las condiciones que provocan que exista una transición entre estados. Estos modelos se pueden ver como computadoras abstractas que pueden encontrarse en uno de un número finito de estados.

Una *cadena de Márkov* es una herramienta para representar distribuciones de probabilidad de una secuencia de observaciones, por lo que se pueden ver como una manera de representar máquinas de estado finitas con probabilidades de transición: se pasa del estado s_i al estado s_j con una probabilidad p_{ij}. Cabe destacar que estas probabilidades pueden ser representadas como una matriz de transición.

Un *modelo oculto de Márkov* (HMM) es esencialmente una *cadena de Márkov* con la diferencia de que los estados de ésta se encuentran asociados a una distribución de probabilidad sobre todos los valores de salida. Los HMM han sido extensamente utilizados en aplicaciones relacionadas con el reconocimiento de voz, sin embargo, su uso ha ido en aumento en los últimos años en el área de visión computacional [5]. El enfoque clásico sugiere que sean utilizadas en el dominio temporal; no obstante, es posible pensar en utilizarlas en el dominio espacial. En la Figura 3.5 se muestra un ejemplo de una representación de un HMM.

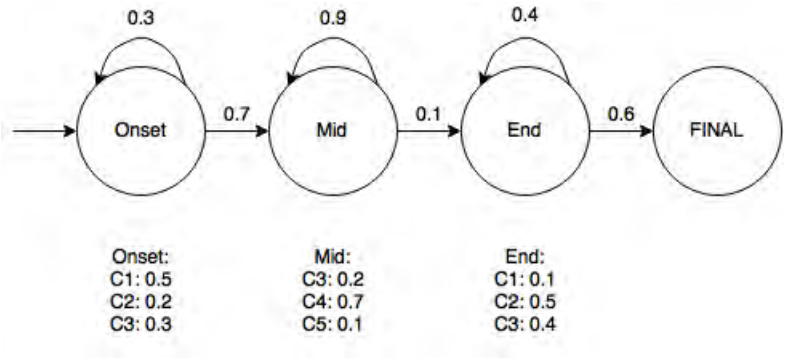

Figura 3.5: *Modelo oculto de Márkov* para un fonema. Cada estado tiene asociadas observaciones con sus respectivas probabilidades de emisión. Imagen extraída de [28].

3.2.2.1. Elementos

Los elementos de un *modelo oculto de Márkov* son:

- N, el número de estados en el modelo. Se denotan los estados individuales como $S = s_1, s_2, \ldots, s_N$ y el estado al tiempo t como q_t.
- M, el número de distintos símbolos de observación por estado. Se denotan los símbolos individuales como $V = v_1, v_2, \ldots, v_M$ y la observación al tiempo t como o_t.
- La distribución de probabilidad de las transiciones entre estados $A = a_{ij}$, donde:

$$a_{ij} = P[q_{t+1} = s_j | q_t = s_i] \tag{3.4}$$

$$1 \leq i, \quad j \leq N$$

- La distribución de probabilidad de los símbolos de observación $B = \{b_j(k)\}$, donde:

$$b_j(k) = P[v_k \text{ en } t | q_t = s_j] \tag{3.5}$$

$$1 \leq j \leq N, \quad 1 \leq k \leq M$$

- La distribución de probabilidad inicial $\pi = \{\pi_i\}$, donde

$$\pi_i = P[q_1 = s_i] \tag{3.6}$$

$$1 \leq i \leq N$$

La notación para indicar los parámetros del modelo es la siguiente:

$$\lambda = (A, B, \pi) \tag{3.7}$$

3.2.2.2. Propiedades

Los *modelos ocultos de Márkov* tienen dos propiedades especiales para codificar una señal a las cuales se le atribuye su nombre [5]:

- La observación en un tiempo t es generada por un estado q_t que se encuentra "oculto" del observador.
- El estado de este proceso "oculto" satisface la propiedad de Márkov, la cual indica que, dado el valor de q_{t-1}, el actual estado q_t es independiente de todos los estados anteriores a $t - 1$.

Otra suposición que se realiza es que la variable oculta es discreta: q_t sólo podrá tomar valores de un conjunto de N elementos.

3.2.2.3. Problemas básicos

Los modelos ocultos de Márkov son una poderosa herramienta matemática cuya ejecución computacional es relativamente simple. Existen tres problemas básicos que deben ser resueltos para ser útiles en aplicaciones reales: evaluación, decodificación y aprendizaje. La solución a estos problemas se presenta en [29] y se desarrollan en las siguientes subsecciones.

3.2.2.3.1. Evaluación El problema de evaluación consiste en el cálculo eficiente de las probabilidad de una secuencia de observaciones $O = o_1 o_2 ... o_T$ dado un modelo $\lambda = (A, B, \pi)$, es decir, $P(\lambda|O)$. Existen dos algoritmos que se utilizan para resolver este problema, el *algoritmo de avance* y el *algoritmo de retroceso.*

3.2.2.3.1.1. Algoritmo de avance Se introduce una variable $\alpha_t(i)$, denominada *variable de avance*, la cual se define en la Ecuación 3.8:

$$\alpha_t(i) = P(o_1 o_2 \dots o_t|\lambda) \tag{3.8}$$

La *variable de avance* indica la probabilidad de tener una secuencia de observaciones parcial $o_1 o_2 \dots o_t$ con S_i como estado actual hasta el tiempo t dado el modelo λ.

Se presenta a continuación la solución inductiva para $\alpha_t(i)$:

1. Inicialización:

$$\alpha_1(i) = \pi_i b_i(o_1) \tag{3.9}$$

$$1 \leq i \leq N$$

2. Inducción:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(o_{t+1}) \tag{3.10}$$

$$1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

3. Terminación:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.11}$$

3.2.2.3.1.2. Algoritmo de retroceso Del mismo que en el *algoritmo de avance*, se introduce una variable $\beta_t(i)$, denominada *variable de retroceso*, la cual se define en la Ecuación 3.12:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \dots o_T|q_t = s_i, \lambda) \tag{3.12}$$

En este caso, la *variable de retroceso* indica la probabilidad de tener una secuencia de observaciones parcial $o_{t+1}, \dots, o_T$ con S_i como estado actual a partir del tiempo t dado el modelo λ.

Se presenta a continuación la solución inductiva para $\beta_t(i)$:

1. Inicialización

$$\beta_T(i) = 1 \tag{3.13}$$

$$1 \leq i \leq N$$

2. Inducción

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{3.14}$$

$$t = T-1, T-2, \ldots, 1, \quad 1 \leq i \leq N$$

3. Terminación

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{3.15}$$

3.2.2.3.2. Decodificación El problema de decodificación consiste en la determinación de la secuencia de estados $Q = q_1 q_2 ... q_T$ que mejor explique las observaciones $O = o_1 o_2 ... o_T$ utilizando un modelo $\lambda = (A, B, \pi)$, es decir, $max[P(Q|O, \lambda)]$. Este problema se resuelve con el *algoritmo de Viterbi*. Se introduce la variable $\delta_t(i)$, la cual se define en la Ecuación 3.16:

$$\delta_t(i) = \max_{q_1, q_2, .., q_{t-1}} P(q_1 q_2 ... q_{t-1}, q_t = s_i, o_1 o_2 ... o_t | \lambda) \tag{3.16}$$

Esta variable contiene la probabilidad más alta a lo largo de una trayectoria en el tiempo t.

Por inducción:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(o_{t+1}) \tag{3.17}$$

La trayectoria más probable es aquella que se compone de los argumentos que hacen máxima la Ecuación 3.17 en cada instante de tiempo y para cada estado. Se presenta a continuación el proceso completo en donde la variable $\psi_t(j)$ se utiliza para recuperar esta secuencia de estados:

1. Inicialización

$$\delta_1(i) = \pi_i b_i(o_1) \tag{3.18}$$

$$1 \leq i \leq N$$

$$\psi_t(i) = 0 \tag{3.19}$$

2. Recursión

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) \tag{3.20}$$

$$2 \leq t \leq T, \quad 1 \leq j \leq N$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \tag{3.21}$$

$$2 \leq t \leq T, \quad 1 \leq j \leq N$$

3. Terminación

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.22}$$

$$q_t^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)] \tag{3.23}$$

4. Reconstrucción de la secuencia de estados más probable

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{3.24}$$

$$t = T-1, T-2, \ldots, 1$$

3.2.2.3.3. Aprendizaje El problema de aprendizaje consiste en el ajuste de los parámetros del modelo $\lambda = (A, B, \pi)$ que mejor expliquen la secuencia de observaciones $O = o_1 o_2 ... o_T$, es decir, $max[P(\lambda|O)]$. Este problema se resuelve a través de un proceso iterativo conocido como *algoritmo de Baum-Welch*.

Se introduce la variable $\xi_t(i,j)$, la cual se define en la Ecuación 3.25:

$$\begin{aligned}
\xi_t(i,j) &= P(q_t = s_i, q_{t+1} = s_j | O, \lambda) \\
&= \frac{P(q_t = s_i, q_{t+1} = s_j, O|\lambda)}{P(O|\lambda)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_t(k) a_{kl} b_l(o_{t+1}) \beta_{t+1}(l)}
\end{aligned} \tag{3.25}$$

Esta variable indica la probabilidad de estar en el estado S_i en el instante t y en el estado S_j en el instante $t+1$, dado una observación O y el modelo λ. Las variables α y β se calculan con los algoritmos de avance y retroceso previamente presentados.

Se define $\gamma_t(i)$ como la probabilidad de estar en el estado s_i en el instante t:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.26}$$

Sumando cada $\gamma_t(i)$ y $\xi_t(i,j)$ en cada instante de tiempo, se tiene:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{ Número esperado de transiciones desde } s_i \tag{3.27}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{ Número esperado de transiciones desde } s_i \text{ a } s_j \tag{3.28}$$

Estos valores se utilizan para el proceso de reestimación de parámetros en el modelo. Si $\lambda = (A, B, \pi)$ es el modelo inicial y $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ es el modelo reestimado, se puede demostrar que $P(O|\overline{\lambda}) > P(O|\lambda)$, con lo cual se ha encontrado un nuevo modelo $\overline{\lambda}$ a partir del cual es más probable que la secuencia de observaciones se haya generado.

Las fórmulas de reestimación son las siguientes:

- Probabilidad de estar en el estado s_i en el tiempo $t = 1$:

$$\overline{\pi}_i = \gamma_1(i) \tag{3.29}$$

$$1 \leq i \leq N$$

- Reestimación de probabilidades de transición:

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.30}$$

$$1 \leq i \leq N, \quad 1 \leq j \leq N$$

- Reestimación de probabilidades de emisión:

$$\overline{b}_j(o_k) = \frac{\sum_{t=1:o_t=o_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.31}$$

$$1 \leq j \leq N, \quad 1 \leq k \leq N$$

La condición de paro está dada por el decremento en el valor de $P(O|\overline{\lambda})$ y por el alcance de un número máximo de iteraciones.

Para realizar el entrenamiento de un modelo utilizando un número R secuencias de observaciones, las ecuaciones 3.29 a 3.31 anteriores se modifican de la siguiente manera:

$$\overline{\pi}_i = \sum_{r=1}^{R} \gamma_1^r(i) \tag{3.32}$$

$$\overline{a}_{ij} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r-1} \xi_t^r(i,j)}{\sum_{r=1}^{R} \sum_{t=1}^{T_r-1} \gamma_t^r(i)} \tag{3.33}$$

$$\overline{b}_j(o_k) = \frac{\sum_{r=1}^{R} \sum_{t=1:o_t=o_k}^{T_r} \gamma_t^r(j)}{\sum_{r=1}^{R} \sum_{t=1}^{T_r} \gamma_t^r(j)} \tag{3.34}$$

Capítulo 4

Desarrollo

En este capítulo se describirá el sistema para el reconocimiento de objetos propuesto el cual es la parte medular de este trabajo de tesis. Este sistema se encuentra dividido en cuatro etapas comunes para los procesos de entrenamiento y reconocimiento: adquisición, detección, pre-procesamiento y representación. Se muestra un diagrama del sistema propuesto en la Figura 4.1.

4.1. Adquisición

En esta etapa se utiliza un sensor *RGB-D* en conjunto con los controladores proporcionados por un *software* adecuado para la adquisición de los datos necesarios para la generación de la nube de puntos de una escena, la cual consiste comunmente de una cantidad máxima de 307,200 puntos cuando se tiene una resolución VGA. Cada uno de estos puntos contiene la información de sus coordenadas X, Y, Z y de sus valores RGB.

4.2. Detección

El propósito de esta etapa es determinar la presencia de objetos y su localización en una escena. Es común establecer hipótesis sobre el estado de los objetos para facilitar su detección ya que una búsqueda exhaustiva de sus modelos en la escena, como el uso de ventanas deslizantes [30], no es opción para el procesamiento en tiempo real. Una de las restricciones más comunes que se emplean es delimitar la búsqueda a objetos que se encuentran sobre un plano, el cual pudiera ser una mesa, un anaquel, el suelo, etc. Esta

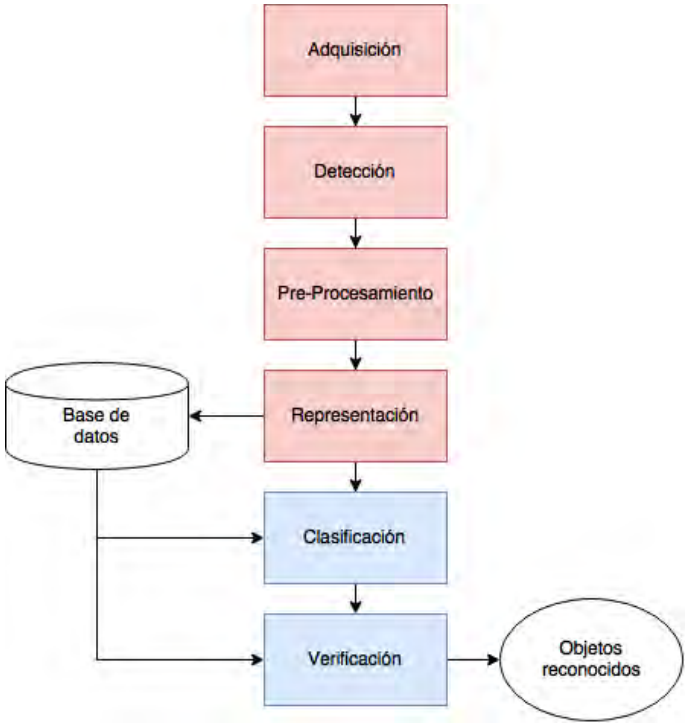

Figura 4.1: Diagrama del sistema de reconocimiento de objetos propuesto. Las etapas en color rojo son compartidas por los procesos de entrenamiento y reconocimiento mientras que las etapas en color azul son exclusivas para el proceso de reconocimiento.

información es aprovechada por varios algoritmos de detección para facilitar el proceso de segmentación de los objetos [31]. La segmentación se refiere a una forma de procesamiento que consiste en separar una imagen o una nube de puntos en agrupamientos de *píxeles* o puntos conocidos como ***clusters***, los cuales podrían pertenecer a objetos en la escena.

4.2.1. Segmentación de un plano

El objetivo de esta etapa es encontrar el plano presente en la escena, lo cual es de interés debido a que, como ya se mencionó, se delimita la búsqueda a los objetos que se encuentran sobre él. Un plano puede ser definido con un modelo paramétrico de cuatro componentes (a, b, c, d) utilizando la ecuación $ax + by + cz + d = 0$, por lo que la obtención de estos parámetros permite determinar los puntos de la nube que pertenecen a él. *Random Sample*

Consencus (RANSAC) [32] es un algoritmo iterativo para la estimación de parámetros de un modelo matemático que lidia con grandes cantidades de valores atípicos. Su empleo en la segmentación de planos en nubes de puntos como apoyo en la tarea de detección de objetos es extenso [31, 33]. La idea de RANSAC para esta tarea consiste en elegir aleatoriamente tres puntos y revisar cuantos de los puntos restantes pertenecen al plano estimado con un margen de error. Se elige el plano calculado con la mayor cantidad de estos puntos y su ecuación se mejora utilizando el método de mínimos cuadrados. El algoritmo RANSAC se describe a detalle a continuación:

1. Se selecciona aleatoriamente el mínimo número de puntos necesarios dentro del total para determinar los parámetros del modelo. En el caso estudiado, son tres puntos los que se utilizan para la obtención de la ecuación del plano.

2. Se calculan los parámetros del modelo con estos puntos. Para el caso del plano, la obtención su ecuación empleando tres puntos $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$ y $P_3 = (x_3, y_3, z_3)$ se realiza de la siguiente manera:

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0 \tag{4.1}$$

3. Se determina la cantidad de puntos del conjunto total que se ajusten al modelo con una tolerancia ϵ, a los que se les denominan ***inliers***. En este caso se utiliza la distancia de un punto al plano:

$$\varepsilon = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{4.2}$$

4. Si la razón de la cantidad de *inliers* sobre el número total de puntos sobrepasa un umbral predefinido, se vuelven a estimar los parámetros del modelo con el resto de los puntos.

5. Se repite el algoritmo una cantidad k de veces.

En cada iteración el algoritmo genera un modelo que podría refinarse utilizando todos los *inliers* o que podría descartarse por su reducida cantidad de éstos. Se termina por elegir el modelo con más *inliers*.

Para hacer una estimación del número de iteraciones k necesario para el algoritmo de RANSAC se comienza definiendo w como la probabilidad de seleccionar un *inlier* del total de puntos. Si w_n es la probabilidad de que los n puntos necesarios para la estimación del modelo pertenezcan al modelo, $1 - w_n$ es la probabilidad de que al menos uno de estos puntos seleccionados no sea un *inlier* y $(1 - w_n)^k$, la probabilidad de que nunca se seleccione un conjunto en el que todos sus puntos sean *inlier*, entonces:

$$1 - p = (1 - w_n)^k \tag{4.3}$$

siendo p la probabilidad de seleccionar sólo *inliers*. Entonces, despejando para k:

$$k = \frac{\log(1 - p)}{\log(1 - w_n)} \tag{4.4}$$

Este valor k puede elegirse arbitrariamente, por lo que es posible reducir el número de iteraciones determinado anteriormente con el inconveniente de incrementar la incertidumbre del error. Existen modificaciones al algoritmo para hacerlo más eficiente como la propuesta en [34] en la se cual añade una restricción que indica que sólo puntos con vectores normales que tengan la misma orientación serán considerados para la estimación de parámetros.

La segmentación de un objeto en un plano utilizando este algoritmo se muestra en la Figura 4.2.

4.2.2. Extracción de agrupamientos

La segmentación permite obtener los *clusters* en una escena, siendo la segmentación euclidiana uno de los métodos más sencillos. Básicamente, la idea consiste en calcular las distancias entre pares de puntos y verificar que sus valores no sobrepasen un umbral predefinido para ser considerados parte de un mismo *cluster*. De [35] obtenemos el siguiente algoritmo:

1. Se crea un *árbol kd* (ver Sección 3.2.1.1) para la nube de puntos de entrada P.

2. Se crea una lista vacía de *clusters* C y una cola de puntos Q que necesitan ser verificados.

3. Para cada punto $p_i \in P$, se realiza lo siguiente:

   a) Se añade p_i a la cola Q.

Figura 4.2: Segmentación de un objeto sobre un plano utilizando el algoritmo RANSAC.

b) Para cada punto $p_i \in Q$, se realiza lo siguiente:

1) Se busca el conjunto P_i^k de puntos vecinos a p_i dentro de una esfera con radio r.
2) Para cada vecino $p_i^k \in P_i^k$, se verifica si el punto ya ha sido procesado y si no, se añade a Q.

c) Cuando la lista de todos los punto en Q ha sido procesada, se añade Q a la lista de *clusters* C y se reinicia Q.

4. El algoritmo termina cuando todos los puntos $p_i \in P$ han sido procesados y ahora son parte de la lista de *clusters*.

Aquellos *clusters* cuya cardinalidad exceda un número n_{min} de puntos se consideran candidatos a ser nubes de puntos que definen objetos en la escena.

4.3. Pre-procesamiento

Como ya se ha mencionado, la tecnología actual nos permite obtener representaciones de objetos complejos en forma de nubes de puntos, las cuales suelen componerse por grandes cantidades de datos. Sin embargo, esta información suele tener problemas como la presencia de ruido proveniente de las limitaciones tecnológicas en el proceso de adquisición y el exceso de información que pudiera incrementar los tiempos de procesamiento. Las operaciones de filtrado tienen como objetivo afinar detalles, eliminar el ruido, conservar sólo la información más importante para un análisis más eficiente, entre otros.

4.3.1. Reducción de la resolución

Una nube de puntos generada a partir de un mapa de profundidad con resolución VGA tendrá una cantidad máxima de 307,200 puntos, lo cual requiere de grandes cantidades de memoria y de tiempo de procesamiento. Los *métodos de reducción de resolución basados en mallas de vóxeles* son muy utilizados para disminuir la cantidad de información que pudiera ser redundante en nubes de puntos [36]. Estos métodos emplean estructuras de datos conocidas como *árboles octales*, las cuales se utilizan comúnmente para segmentar el espacio tridimensional. En este tipo de estructuras, cada nodo subdivide al espacio que representa en ocho secciones, tal como se muestra en la Figura 4.3.

La nube de puntos será envuelta por un paralelepípedo, el cual se dividirá recursivamente haciendo uso de los *árboles octales* en ocho vóxeles del mismo tamaño hasta alcanzar una resolución predefinida. Para cada uno de los *vóxeles* generados se calcula el centroide de los puntos contenidos en él. Los centroides obtenidos serán una nueva representación de la nube de puntos original la cual utiliza una cantidad menor de información.

4.3.2. Extracción estadística de valores atípicos

Es común la existencia de información en las nubes de puntos que pudo haber sido generado por errores de medición, por lo cual es conveniente deshacerse de aquellos puntos que puedan generar errores en los cálculos. Se optó por utilizar una técnica de análisis estadístico sobre las vecindades de los puntos para eliminar aquellos que no cumplen ciertos criterios. En este caso, se utilizará el promedio de las distancias que existen entre los puntos analizados y sus vecinos para el cálculo de una distribución que se asumirá gaussiana, con una media μ y una desviación estándar σ. Todos los puntos cuyo promedio calculado se encuentre fuera de un intervalo definido en términos de μ y σ son eliminados.

4.3.3. Reconstrucción de superficies

No siempre es posible eliminar irregularidades en los datos utilizando solamente métodos estadísticos como el descrito anteriormente, es por eso que se emplean métodos para la reconstrucción de superficies con el objetivo de eliminar errores de captura y preservar detalles de los modelos para las siguientes etapas de procesamiento [37]. El método de *mínimos cuadrados móviles* (MLS, por sus siglas en inglés: *Moving Least Squares*) [38] realiza

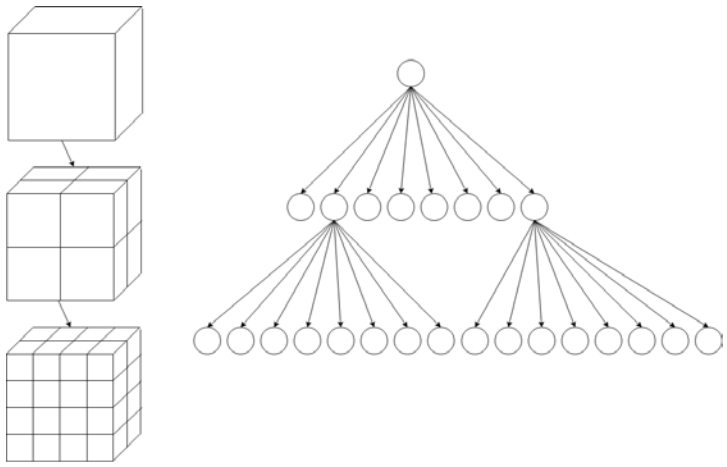

Figura 4.3: a) División recursiva del espacio 3D en octantes b) *Árbol octal* correspondiente.

esta reconstrucción aproximando localmente la superficie con polinomios de alto orden. Este método es una extensión de la popular técnica de mínimos cuadrados para el ajuste de curvas, donde el término "móvil" proviene de las distintas ponderaciones de los puntos para calcular sus contribuciones a la solución en diferentes posiciones.

Se tiene una superficie S con puntos $p_i \in \mathbb{R}^3, i \in 1, ..., N$. El objetivo es proyectar un punto $r \in \mathbb{R}^3$ cerca de S en una nueva superficie que se aproxime al conjunto de puntos p_i. De [39] se obtiene el procedimiento para realizar esta proyección el cual se divide en tres pasos:

1. Se calcula un plano de referencia local en la vecindad de r (ver Figura 4.4). De manera más específica, se encuentra un plano H con un vector normal $a \in \mathbb{R}^3$ que pasa a través de un punto $q = r + ta$, con $t \in \mathbb{R}$, de tal manera que $\|a\| = 1$ y H se calcula minimizando una suma ponderada de las distancias de los puntos p_i a este plano, la cual se presenta en la Ecuación 4.5:

$$\begin{aligned} e_{MLS}(a,t) &= \sum_{p_i \in \wp} \langle a, p_i - q \rangle^2 \theta(\|p_i - q\|) \\ &= \sum_{p_i \in \wp} \langle a, p_i - r - ta \rangle^2 \theta(\|p_i - r - ta\|) \end{aligned} \tag{4.5}$$

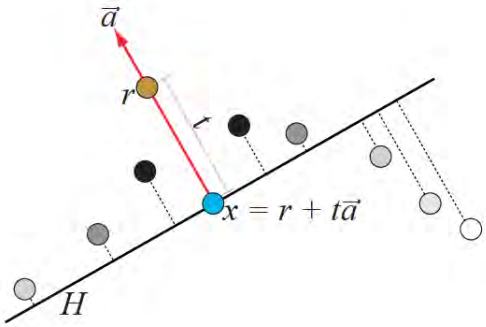

Figura 4.4: Proyección de un punto utilizando MLS. Imagen extraída de [40].

donde $\langle ., . \rangle$ es el producto punto de dos vectores y $\|.\|$ es la magnitud de un vector. En este caso, $\theta\left(\|p_i - q\|\right)$ es una función de ponderación y $\langle a, p_i - q \rangle^2$ es la distancia cuadrática de cada punto p_i al plano H.

2. Se encuentra un polinomio bivariable de aproximación de grado m $\tilde{g} \in \prod_m^2$ que minimize localmente el error alrededor de q. Se consideran las proyecciones ortogonales de los puntos $\{p_i\}$ como los puntos con coordenadas (x_i, y_i) en el plano H con la distancia proyectada $f_i = \langle a, p_i - q \rangle$. Los coeficientes de g se obtienen minimizando la siguiente función:

$$\sum_i \left(g\left(x_i, y_i\right) - f_i\right)^2 \theta\left(\|p_i - q\|\right) \tag{4.6}$$

3. El punto proyectado se define como:

$$proj_{MLS}(r) = q + g(0,0)n = r + (t + g(0,0))n \tag{4.7}$$

4.3.4. Obtención de puntos de interés

La extracción de puntos de interés o *puntos característicos* se realiza para obtener aquellos puntos en la nube que sean representativos y puedan ser encontrados ante variaciones en los puntos de vista. Además, evitan la complejidad computacional que implicaría trabajar con todos los datos. Es común utilizar algún método de reducción de resolución para la obtención de *puntos característicos* en nubes de puntos, sin embargo, se decidió utilizar

un método de sobre-segmentación volumétrica de nubes de puntos conocido como *segmentación de nubes basada en conectividad de vóxeles* (VCCS, por sus siglas en inglés: *Voxel Cloud Connectivity Segmentation*) [41] para la generación de regiones con características similares definidas como ***supervóxeles*** y utilizar sus centroides como *puntos característicos*, tal como se ha realizado con los *superpíxeles* para el caso bidimensional [42], donde se aprovecha el hecho de que las regiones alrededor de sus centroides contienen puntos que comparten información similar y facilitan su codificación.

VCCS utiliza una variante del algoritmo de *k-medias* [43] en el cual la selección inicial de los centroides de los *clusters* se realiza directamente en la nube de puntos mediante la división del espacio tridimensional, con lo cual se evita su expansión más allá de las fronteras que se encuentren desconectadas. Se emplea una conectividad-26, lo que significa que se consideran vecinos aquéllos *vóxeles* que comparten caras, aristas y vértices. Dada una nube de puntos que ha sido filtrada mediante un *método de reducción de resolución basado en mallas de vóxeles* (ver Sección 4.3.1) con una resolución R_{voxel}, se construye una gráfica de adyacencia sobre la nube voxelizada considerando la conectividad anteriormente mencionada. Los centroides de los *supervóxeles* se obtienen creando una nueva malla de *vóxeles* sobre la nube de puntos con resolución R_{seed} (cuyo valor es significativamente mayor a R_{voxel}) y se eligen como los puntos en la nube que se encuentren más cerca de los centros de estos *vóxeles*. Además, los centroides que no cumplan la condición de tener como mínimo un número determinado de *vóxeles* dentro de una distancia R_{search} serán eliminados. En la Figura 4.5 se observan los distintos parámetros involucrados en la selección inicial de estos centroides.

La expansión de los *clusters* se encuentra regida por una distancia que considera la extensión espacial de los puntos, sus colores y sus normales, la cual se define a continuación:

$$D = \sqrt{w_c D_c^2 + \frac{w_s D_s^2}{3R_{seed}^2} + w_n D_n^2} \tag{4.8}$$

donde D_s es la distancia espacial normalizada por R_{seed}, D_c la distancia euclidiana en el espacio RGB normalizado y D_n es la distancia normal que mide el ángulo entre los vectores normales. Los parámetros w_c, w_s y w_n se utilizan para controlar en que medida se desea que cada una de estas características influya en el cálculo de las distancias.

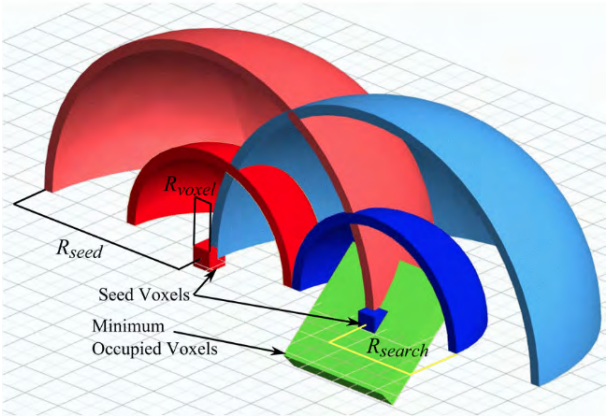

Figura 4.5: Parámetros para la selección inicial de centroides. Imagen extraída de [41].

El proceso para el crecimiento iterativo de los *supervóxeles* se describe a continuación: comenzando con un centroide, la distancia D se calcula para los *vóxeles* adyacentes de acuerdo a la Ecuación 4.8. Si la distancia obtenida es la menor que este *vóxel* ha generado con respecto a otros centroides, éste se clasifica como perteneciente al *supervóxel* asociado a este centroide y los vecinos de este *vóxel* se añaden a la cola de búsqueda para esta clase particular. Se realiza lo mismo para los siguientes centroides de manera que cada nivel hacia fuera del centro es considerado al mismo tiempo para todos ellos. Se continua iterativamente hasta que todos los *vóxeles* quedan clasificados y se concluye obteniendo los nuevos centroides de los *supervóxeles* utilizando los elementos que los constituyen. Para mayor claridad, se muestra la Figura 4.6 que explica el proceso para el caso bidimensional.

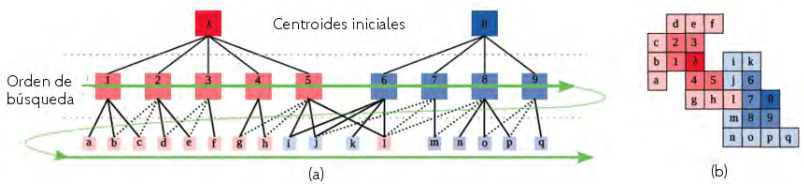

Figura 4.6: (a) Procedimiento de búsqueda para el ejemplo presentado en (b). Imagen extraída de [41].

4.4. Representación

La habilidad de un método para la realizar la representación exclusiva de cada objeto a partir de la información disponible determina su efectividad [44], es por ello que esta etapa es fundamental en el desempeño del sistema. Se decidió utilizar representaciones basadas en *características* que consideran aspectos visuales y geométricos locales.

La representación de los objetos se realizará en forma de modelos estadísticos basados en ***símbolos***, los cuales son elementos de un alfabeto finito generados a partir de las características anteriormente mencionadas y se utilizan para la clasificación de datos. La primer representación consiste en el cálculo de los vectores de distribución de frecuencias normalizados de los *símbolos* asociados a los *puntos característicos* de los objetos mientras que la segunda se basa en la generación de *modelos ocultos de Márkov* utilizando secuencias de observaciones que se crean a partir de los *símbolos* de estos puntos considerando su orden espacial.

Para ordenar en secuencia los *puntos característicos* se comienza rotando y trasladando las nubes de puntos a un sistema de referencia en común, tal como se hace cuando se desea realizar el registro de dos nubes [45]. Lo anterior se lleva a cabo con el propósito de eliminar las variaciones que se existen entre los modelos de los objetos entrenados y los que se observan en una escena debido a que las capturas se obtienen desde diferentes puntos de vista. Para lograr esto se utiliza un método estadístico conocido como Análisis de Componentes Principales (PCA, por sus siglas en ingles: *Principal Components Analysis*), el cual se utiliza para realizar la proyección de datos en una base ortonormal en dirección de la mayor varianza.

El procedimiento para realizar el alineamiento de una nube de puntos por medio de PCA es el siguiente:

1. La nube de puntos P es trasladada de tal forma que su centroide coincida con el origen del sistema de referencia global.

2. Se calcula la matriz de covarianza C de P y se calculan sus vectores propios $\overline{v}_j$:

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p}) \cdot (p_i - \overline{p})^T \tag{4.9}$$

$$C \cdot \overline{v}_j = \lambda_j \cdot \overline{v}_j, \quad j \in \{0, 1, 2\} \tag{4.10}$$

donde k es el número de puntos de P, $\overline{p}$ el centroide de P, λ_j es el valor propio número j de C y $\overline{v}_j$ el vector propio número j.

3. Se forma una base ortogonal con $\overline{v}_j$ que corresponde a las componentes principales de P_k.

Los *símbolos* son ordenados en secuencias de observaciones de acuerdo a su posición espacial con respecto al nuevo sistema de referencia generado. Para esto, se divide el espacio en *vóxeles* y se acomodan los puntos contenidos en cada uno de ellos tomando en cuenta el orden en el que se analizan cada uno de los vóxeles, el cual se determina de acuerdo a los valores propios asociados a los vectores propios que definen el nuevo sistema de referencia.

4.4.1. Extracción de características

Los *símbolos* que se utilizan para la representación de los objetos en este sistema se generan a partir de información geométrica y visual proveniente de los puntos más cercanos a los *puntos característicos* utilizando el *método de los K-vecinos más cercanos* descrito en la Sección 3.2.1.

4.4.1.1. Información visual

La generación de *símbolos* a partir de la información visual de los *puntos característicos* se realiza utilizando el color de sus puntos adyacentes. Los valores *RGB* de los puntos son transformados al modelo de color *HSV*, el cual es más utilizado en aplicaciones de visión computacional debido a su robustez ante cambios en la iluminación y, además, permite al usuario especificar intuitivamente los colores [46], lo cual facilita la generación de *símbolos*. Este modelo es obtenido mediante la transformación de las coordenadas cartesianas del modelo *RGB* en cilíndricas, las cuales corresponden aproximadamente a tres propiedades perceptuales del color: matiz, saturación y brillo. Las ecuaciones para realizar dicha transformación son las siguientes:

$$R' = \frac{R}{255}, \quad G' = \frac{G}{255}, \quad B' = \frac{B}{255} \tag{4.11}$$

$$C_{max} = \max(R', G', B'), \quad C_{min} = \min(R', G', B') \tag{4.12}$$

$$H = \begin{cases} 0, & C_{max} - C_{min} = 0 \\ 60_{\times}(\frac{G'-B'}{C_{max}-C_{min}}), & C_{max} = R', G' \geq B' \\ 60_{\times}(\frac{G'-B'}{C_{max}-C_{min}} + 6), & C_{max} = R', G' < B' \\ 60_{\times}(\frac{B'-R'}{C_{max}-C_{min}} + 2), & C_{max} = G' \\ 60_{\times}(\frac{R'-G'}{C_{max}-C_{min}} + 4), & C_{max} = B' \end{cases} \tag{4.13}$$

$$S = \begin{cases} 0, & C_{max} = 0 \\ \frac{C_{max}-C_{min}}{C_{max}}, & C_{max} \neq 0 \end{cases} \tag{4.14}$$

$$V = C_{max} \tag{4.15}$$

Los rangos en los cuales se encuentran estos valores son:

$$0 \leq H \leq 360$$
$$0 \leq V \leq 1{,}0$$
$$0 \leq S \leq 1{,}0$$

El espacio HSV es segmentado con la intención de generar regiones con características similares y clasificar los puntos de acuerdo a sus valores en este espacio.

4.4.1.2. Información geométrica

La generación de *símbolos* a partir de información geométrica se realiza con el apoyo de descriptores 3D, los cuales utilizan frecuentemente las normales de los puntos debido a que proporcionan información de la curvaturas de las superficies a las que pertenecen. La obtención de las normales se efectúa utilizando el método propuesto en [35], en donde la estimación de la normal de un punto en la nube se basa en obtener la normal del plano tangente a la superficie a la que pertenece. Esta normal puede ser calculada utilizando los puntos vecinos P_k al punto en cuestión (obtenidos mediante el uso de *árboles kd* descritos en la Sección 3.2.1.1) por medio de un ajuste de mínimos cuadrados tal que la distancia d_i al plano de un punto $p_i \in P_k$

sea cero. Si el plano es representado como un punto x y un vector normal $\overrightarrow{n}$, entonces, para este caso $d_i = 0 = (p_i - x) \cdot \overrightarrow{n}$. Considerando a x como el centroide de P_k, se tiene que la solución a $\overrightarrow{n}$ se puede obtener mediante el uso del método PCA (ver Sección 4.4). Si $\lambda_0, \lambda_1, \lambda_2$ son los valores propios de la matríz de covarianza de P_k y $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, el vector propio correspondiente al valor propio más pequeño λ_0 es una aproximación de la normal $\overrightarrow{n}$.

Se propone utilizar el *descriptor de superficie basado en radio* (*RSD*, por sus siglas en inglés: Radious-based Surface Descriptor) [47] debido a que su alta capacidad descriptiva y su baja dimensionalidad permiten generar *símbolos* eficientemente. Este descriptor se encarga de codificar la geometría local de un punto y su vecindario en términos de sus radios, lo cual genera valores que la describen de una manera intuitiva. El cálculo de este descriptor sobre un *punto característico* consiste en hacer pares con él y con los puntos dentro de su vecindad. Para cada uno de ellos se asume que pertenecen a la superficie de una esfera que además se ajusta a las normales previamente obtenidas como se muestra en la Figura 4.7. La Ecuación 4.16 se obtiene de las relaciones geométricas que se pueden hacer a partir de la formulación anterior:

$$d_{(\alpha)} = \sqrt{2} r \sqrt{1 - \cos \alpha} \tag{4.16}$$

En este mismo artículo se propone hacer la descomposición en serie de Taylor de la ecuación 4.16 obteniéndose la siguiente estimación:

$$d(\alpha) = \sqrt{2} r \sqrt{1 - \cos(\alpha)} \approx r\alpha + \frac{r\alpha^3}{24} + O(\alpha^5) \tag{4.17}$$

Esta relación resulta ser casi lineal para $\alpha \in (0, \frac{\pi}{2})$ por lo que la ecuación 4.17 se puede reducir a $d \approx r\alpha$.

Para estimar de manera precisa los radios máximos y mínimos del punto en cuestión y su vecindad, se realiza una regresión lineal sobre los extremos de los pares (α, r). Una interpretación geométrica intuitiva de estos radios permite clasificar la superficie en la que se encuentra el punto dentro de una categoría, lo cual es aprovechado para la generación de *símbolos*. En el caso que un punto sea parte de una superficie esférica es posible ver que el tamaño de los radios generados serán de la misma longitud; en el caso en el

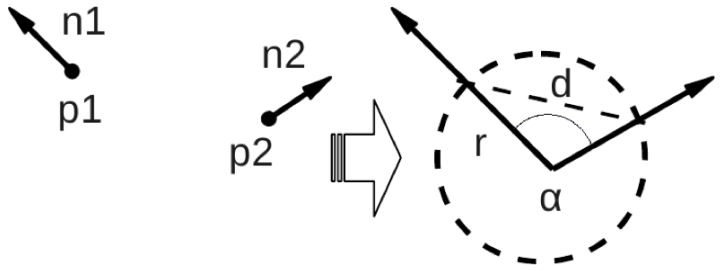

Figura 4.7: Esfera formada a partir de la distancia y el ángulo entre dos normales. Imagen extraída de [47].

que pertenezca a una superficie plana, los radios estimados serán infinitos debido a que sus normales son paralelas.

4.4.1.3. Dimensiones

Las dimensiones de un objeto son una característica básica de ellos que es altamente discriminatoria al utilizarse en conjunto con otro tipo de información. Es por ello que se decidió utilizar las alturas de los objetos, de las cuales se pueden obtener aproximaciones de sus valores a partir de sus nubes de puntos. Lo anterior se realiza con el conocimiento de que los objetos en una escena se encuentran en una cantidad finita de posiciones sobre un plano por lo que los valores estimados de las alturas logran tener una alta repetibilidad.

El procedimiento para la estimación de la altura de un objeto consiste en iterar sobre todos los puntos de su nube de puntos y calcular la distancia de cada uno de ellos al plano utilizando la ecuación 4.2. La altura estimada del objeto en cuestión se considerará como la distancia máxima obtenida en este proceso.

4.5. Entrenamiento y reconocimiento

Hasta ahora, los datos en crudo del sensor han sido convertidos en representaciones individuales de cada objeto. Estas representaciones son utilizadas para los procesos de entrenamiento y reconocimiento descritos en las siguientes secciones. Los clasificadores propuestos se encuentran basados en (1) el *método del vecino más cercano* (1-NN) y (2) el uso de *Modelos Ocultos*

de Markov (HMM), los cuales se describen a detalle en la Sección 3.2.

4.5.1. Entrenamiento

El proceso de entrenamiento en el sistema consiste en la creación de una base de datos de las representaciones de los objetos que se desean reconocer. El usuario obtiene las representaciones de capturas de diferentes vistas de cada objeto con el objetivo de comprender la mayor cantidad de variaciones posibles y les asigna un nombre con el que puedan ser asociadas al objeto al que pertenecen.

El sistema en combinación con el clasificador 1-NN se basa en la representación de los objetos con vectores de distribuciones de frecuencias normalizados de sus *símbolos*, los cuales son generados por cada captura que se realiza a cada objeto a entrenar y son almacenados en un archivo *.xml* con sus respectivos nombres.

El sistema en combinación con el clasificador HMM utiliza las secuencias de observaciones generadas a partir de la obtención de los *símbolos* de distintas capturas de las distintas vistas de cada objeto para obtener los parámetros de distintos modelos asociados a cada vista particular mediante el *algoritmo Baum-Welch* descrito en la Sección 3.2.2.3.3, el cual se encarga de encontrar los parámetros que mejor se ajusten a estas secuencias.

Además, el sistema almacena las alturas estimadas de los objetos de entrenamiento con las que se realiza un análisis estadístico similar al presentado en la Sección 4.3.2: se obtienen los errores de las alturas de estos objetos con respecto a la altura promedio de sus modelos y se asumen que éstos pertenecen a una distribución gaussiana, con lo que es posible generar una varianza. El valor de esta varianza se utiliza para establecer un rango de confiabilidad que se empleará en el proceso de reconocimiento.

4.5.2. Reconocimiento

Al utilizar el sistema para el proceso de reconocimiento se utilizan los modelos generados en la fase de entrenamiento en conjunto con nuevas representaciones obtenidas. Las representaciones que se utilizan dependen del clasificador que se esté empleando.

El sistema en combinación con el clasificador 1-NN reconoce al objeto

en la escena utilizando la distancia euclidiana que existe entre el vector de distribución de frecuencias normalizado de sus *símbolos* y otros vectores almacenados en la base de datos. Se clasifica al objeto como la clase a la que pertenece el vector de entrenamiento asociado a la menor distancia generada.

El sistema en combinación con el clasificador HMM utiliza las secuencias de observaciones generadas a partir de la obtención de los *símbolos* de los objetos en la escena, las cuales alimentan cada uno de los modelos generados previamente en la etapa de entrenamiento, con lo cual se generará una probabilidad para cada una de ellas mediante el algoritmo de avance descrito en la Sección 3.2.2.3.1.1. El sistema clasifica al objeto en la escena cómo el almacenado en la base de datos que genera la mayor probabilidad.

Debido a que los clasificadores reconocen a los objetos como aquellos que generan la menor distancia o mayor probabilidad con respecto a los modelos previamente almacenados en la base de datos, cualquier objeto erróneamente reconocido será considerado un *falso positivo*. Es por eso que se decidió realizar una etapa de verificación que utiliza las alturas estimadas de los objetos, para eliminar las clasificaciones erróneas que el sistema pudiera generar. El objeto detectado es reconocido como aquel que el clasificador determinó si su altura estimada entra dentro del rango de confiabilidad con centro en el valor de la altura promedio determinada para ese modelo; de no ser así, el objeto se considerará como *no reconocido*.

Capítulo 5

Experimentación

5.1. Hardware

- Laptop HP Pavilion dv7
  - Procesador: Intel Core i5-2430M de 2.40 GHz
  - Memoria RAM: 8 GB
- *Kinect*^{TM} para Xbox 360 # 1473

5.1.1. Kinect^{TM}

El *Kinect*^{TM} [26] es un sensor de la compañía Microsoft creado originalmente como controlador de videojuegos para ser utilizado con la videoconsola Xbox, sin embargo, su capacidad para la adquisición de mapas de profundidad a bajo costo, comparado con las cámaras estéreo o de tiempo de vuelo, ha generado interés en su uso para aplicaciones relacionadas con la visión computacional. Este sensor utiliza una técnica conocida como *luz estructurada* para la medición de profundidad, la cual consiste en proyectar un patrón conocido de *píxeles* en la escena para capturar su deformación de proyección, con lo que es posible calcular las distancias de los *píxeles* en una escena. Su sistema de sensado se encuentra compuesto por lo siguientes elementos [48]:

- Cámara *RGB*: entrega imágenes formadas por los tres componentes básicos de color en una resolución *VGA* (640x480 *píxeles* con 8 *bits* de profundidad) a una velocidad de 30 cuadros por segundo.

- Sensor de profundidad: consiste en un proyector y una cámara infrarrojos. Juntos crean un mapa de profundidad en un rango que va de los 0.8 a los 3.5 metros de distancia con respecto al sensor y un ángulo de vista de 57º horizontalmente y 43º verticalmente.

En la Figura 5.1 se pueden observar los componentes anteriormente mencionados.

5.2. Software

- Sistema operativo: Ubuntu 14.04
- Lenguaje de programación: C++11
- IDE: Code::blocks

5.2.1. Librerias

- *Open Source Computer Vision Library* (OpenCV) [49] es una librería de código abierto que sirve para implementar métodos de visión computacional para aplicaciones en tiempo real. Contiene mas de 2500 algoritmos optimizados para su uso en aplicaciones que involucran el reconocimiento de rostros, la detección y el seguimiento de objetos, la clasificación de acciones humanas en videos, entre otros; es además utilizada por compañías privadas, grupos de investigación y organismos gubernamentales.
- *Open Natural Interaction* (OpenNI) [26] es un kit de desarrollo de software que contiene funciones que se encargan de la comunicación con sensores de audio y video. Es ampliamente utilizado para extraer la información *RGB-D* de diversos sensores.
- *Point Cloud Library* (PCL) [50] es una librería que consiste en un conjunto de algoritmos diseñados para el apoyo en trabajos con datos tridimensionales, en particular para nubes de puntos. Puede ser utilizada en aplicaciones de filtrado, estimación de descriptores, reconstrucción de superficies, segmentación, entre otras. Además, es una librería de código abierto, lo cual la hace atractiva para su uso comercial y de investigación.

Para el manejo de los *modelos ocultos de Markov*, se utilizó la libreria *C++ implementation of Hidden Markov Model* [51], por lo que se adoptó el

Figura 5.1: Sensor $Kinect^{TM}$. Imagen extraída de [26].

mismo esquema de trabajo utilizado en los ejemplos proporcionados para la generación de los archivos de entrenamiento, el cual es descrito a continuación: una HMM se encontrará especificada en dos archivos: NAME.trans y NAME.emit donde "NAME" es el nombre del HMM. El archivo NAME.trans contiene las probabilidades de transición entre estados, mientras que el archivo NAME.emit contiene las probabilidades de emisión. Las probabilidades de estado iniciales se incluyen al considerar un estado inicial cuya probabilidad de permanecer en él es nula y la de cambiar a los otros estados corresponde a dichas probabilidades. En la Tabla 5.1 se muestra un ejemplo de estos archivos, el cual corresponde al modelo presentado previamente en la Figura 3.5.

El nombre de los archivos en la base de datos se encuentra compuesto por el nombre del objeto entrenado y un subíndice que indicará el número de vista entrenada (ej. caja_3.emit).

Tabla 5.1: Probabilidades asociadas al *modelo oculto de Márkov* presentado en la Figura 3.5 (a) Probabilidades de transmisión almacenadas en el archivo *phone.trans* (b) Probabilidades de emisión almacenadas en el archivo.

(a)

INIT		
INIT	Onset	1
Onset	Onset	0.3
Onset	Mid	0.7
Mid	Mid	0.9
Mid	End	0.1
End	End	0.4
End	FINAL	0.6

(b)

Onset	C1	0.5
Onset	C2	0.2
Onset	C3	0.3
Mid	C3	0.2
Mid	C4	0.7
Mid	C5	0.1
End	C4	0.1
End	C6	0.5
End	C7	0.4

5.3. Parámetros

La elección de los distintos parámetros en el sistema se seleccionaron de acuerdo a una serie de pruebas que consistieron en la modificación del número de observaciones y en su secuencialización para el caso en el que se utiliza el clasificador HMM. Se eligieron aquellos parámetros que generaron la mayor tasa de reconocimiento considerando a la par los tiempos de procesamiento. En esta sección se presentan algunos de estos parámetros y la manera en que fueron elegidos.

Para la generación de los *símbolos* que codifican la información geométrica se decidió utilizar la división del espacio del descriptor *RSD* en regiones que representen geometrías básicas, tal como se propone en [47] y se muestra en la Figura 5.2. Los parámetros involucrados en la división de este espacio se modificáron utilizando una imagen de prueba que nos permitiera ajustarlos visualmente a conveniencia. Los *símbolos* pertenecen a una de las siguientes cinco categorías propuestas: *plano*, *cilindro* o *anillo*, *esfera*, *borde* y *esquina*.

Los *símbolos* que codifican la información visual son generadas a partir de la segmentación del modelo *HSV* en el canal de matiz cada 60 grados, tal como se muestra en la Figura 5.3. Se producen seis regiones que generan las siguientes clases: *rojo*, *azul*, *verde*, *cian*, *magenta* y *amarillo*. Además, se añaden las clases *blanco* y *negro* para puntos con valores bajos en el canal de saturación y cuya clasificación depende de su valor en el canal de brillo. Se obtienen un total de cuarenta *símbolos* al combinar la información geométrica con la visual.

Se decidió trabajar con tres topologías para el clasificador HMM (ver Figura 5.4): modelo lineal, modelo de Bakis y modelo izquierda-derecha. Además, se trabajó con una cantidad variable de estados con la intención de observar como la tasa de reconocimiento se modifica en función de este número.

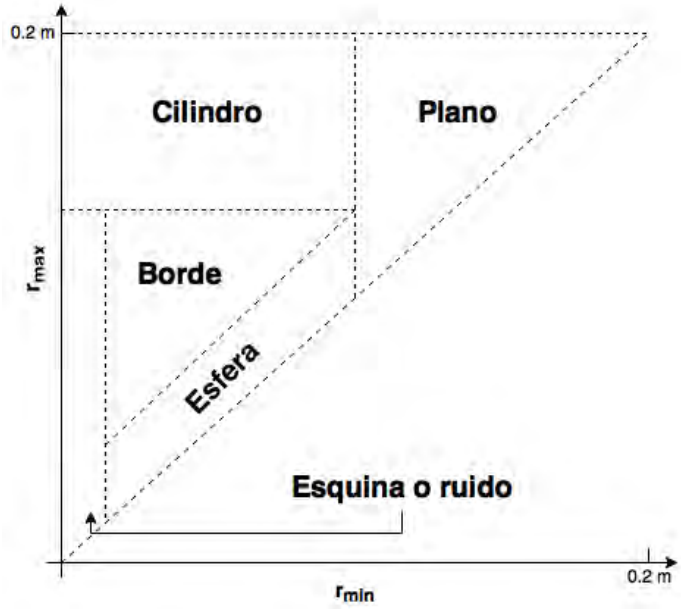

Figura 5.2: División del espacio característico de r_{min} y r_{max} en regiones cuyos valores representan geometrías básicas.

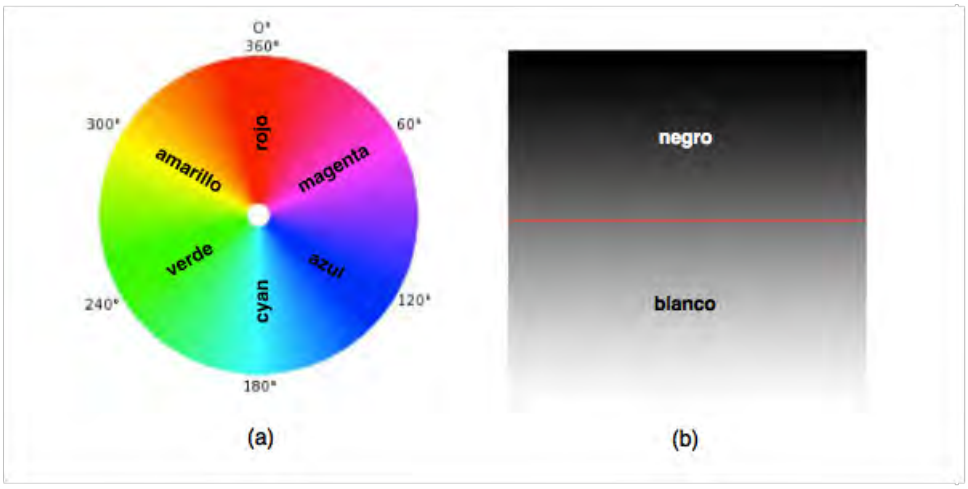

Figura 5.3: Modelo de color *HSV* (a) Representación del canal de matiz por una región circular (b) Representación del canal de brillo con un valor de saturación cero.

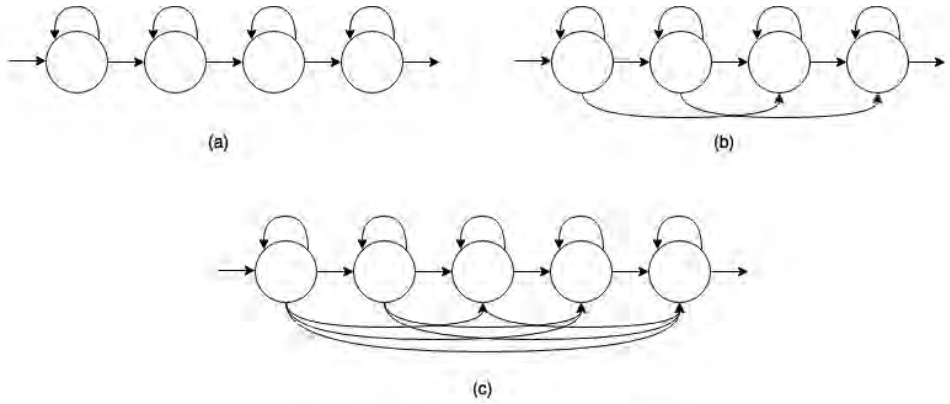

Figura 5.4: Diferentes topologías para los modelos ocultos de Márkov extraídos de [52]. (a) Modelo lineal (b) Modelo de Bakis (c) Modelo de izquiera a derecha.

5.4. Pruebas

5.4.1. Nubes de puntos sintéticas

Se decidió comenzar a realizar pruebas con nubes de puntos creadas artificialmente con el propósito específico de evaluar casos en los que fueran evidentes las ventajas del clasificador HMM en comparación con el clasificador 1-NN. Para ello, se utilizaron los modelos mostrados en la Figura 5.5 a las cuales se les añadió ruido aleatorio en sus valores *HSV* y en sus coordenadas espaciales para simular las distorsiones que ocurren en el proceso de adquisición en diferentes capturas generando un total de diez instancias por modelo. De una manera intuitiva se puede obtener el número de estados que sería adecuado para generar el HMM de estos modelos. Los resultados de esta prueba pueden observarse en las matriz de confusión mostrada en la Tabla 5.2.

5.4.2. Nubes de puntos de objetos

El sistema se evalúa con un conjunto de objetos prevenientes de una base de datos *RGB-D* de objetos presentada en [53], la cual consiste en nubes de puntos de diferentes capturas de los objetos obtenidas al rotarlos de una manera aproximadamente uniforme a diferentes elevaciones y a una distancia de un metro de un total de 300 objetos domésticos y generadas por un sensor $Kinect^{TM}$.

Se limitó la cantidad de objetos de entrenamiento a veinte tomando en cuenta la cantidad de ellos que se requería reconocer en la competencia RockIn (ver Sección 1.3). En la Figura 5.6 se muestra una captura de cada objeto utilizado para realizar estas pruebas.

Tabla 5.2: Matrices de confusión de las nubes de puntos sintéticas con (a) el clasificador 1-NN (b) el clasificador HMM.

(a)

	01	**02**	**03**	**NR**
01	7	0	3	0
02	5	1	4	0
03	2	2	6	0

(b)

	01	**02**	**03**	**NR**
01	10	0	0	0
02	0	10	0	0
03	0	0	10	0

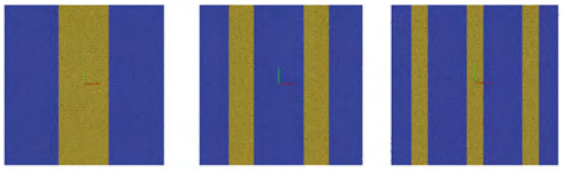

Figura 5.5: Nubes de puntos sintéticas

Figura 5.6: Objetos utilizados en las pruebas.

Para la etapa de entrenamiento se generaron ocho representaciones por objeto, lo cual genera un total de ciento sesenta modelos. Estas representaciones corresponden a ocho vistas distintas y uniformemente espaciadas con las que se tiene la intención de capturar la información completa de cada objeto. En la Figura 5.7 se muestra un conjunto de capturas que corresponden a las vistas del *objeto 09* utilizadas para el entrenamiento. Para las pruebas de reconocimiento se utilizaron diez capturas de cada objeto de la base de datos distintas a las que se utilizaron en la etapa de entrenamiento. El rango de confiabilidad generado a partir de la distribución del error de las alturas de los objetos de entrenamiento con respecto a sus alturas promedio y considerando dos y media desviaciónes estandar fue de $\pm$11,6 mm.

Se decidió comenzar a realizar pruebas utilizando *símbolos* generados a partir de información exclusivamente geométrica y depués realizarlo para *símbolos* generados utilizando ambas informaciones. Las matrices de confusión correspondientes a estas pruebas se muestran en las Tablas 5.3 y 5.4.

Figura 5.7: Conjunto de capturas del *objeto 09* utilizadas para la obtención de las representaciones en la etapa de entrenamiento.

Tabla 5.3: Matriz de confusión de los objetos con el sistema propuesto utilizando *símbolos* generados a partir de información geométrica en conjunto con el clasificador 1-NN sin la inclusión de la etapa de verificación.

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**NR**
01	7	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
02	3	6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	1	0	4	1	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0
04	1	0	2	2	0	0	0	0	0	2	0	0	0	0	2	0	0	0	1	0	0
05	0	0	0	0	6	0	0	0	3	0	0	1	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	1	4	1	0	0	2	1	0	0	0	0	1	0	0	0	0
08	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0	0	2	2	0	0	0
09	0	0	0	0	2	2	0	0	5	0	0	0	0	0	0	0	0	0	0	1	0
10	2	2	0	0	0	0	0	0	0	3	0	0	0	2	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	5	0	0	0	0	0
13	6	0	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0
14	1	0	0	1	0	0	0	0	0	3	0	0	0	3	1	0	1	0	0	0	0
15	0	2	3	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	4	0	0	0	0	0
17	0	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	3	2	0	0	0
18	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	2	4	0	0	0
19	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
20	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	1	0	0	0	4	0

Tabla 5.4: Matriz de confusión de los objetos con el sistema propuesto en conjunto con el clasificador 1-NN utilizando *símbolos* generados a partir de información visual y geométrica sin la inclusión de la etapa de verificación.

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**NR**
01	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	2	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	8	0	0	0	0	0	0	2	0	0	0	0	0	0	0
08	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	1	0	0	0	0
09	0	0	0	0	0	0	0	0	7	0	0	1	0	0	0	0	0	2	0	0	0
10	0	1	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
18	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	9	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0

Tabla 5.5: Matriz de confusión de los objetos con el sistema propuesto utilizando únicamente información sobre sus dimensiones

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**NR**
01	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	8	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	3	0	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
05	0	4	0	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	5	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
07	0	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	5	0
08	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	1	0	0	0
09	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	7	0	3	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	3	0	3	0	3	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	2	0	6	0	0	0	0	0	0	2	0	0
13	0	0	0	0	0	7	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	3	0	0	1	0	0
15	3	2	0	0	0	2	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
16	1	0	0	2	0	0	0	0	0	0	0	0	0	3	0	4	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0
19	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	8	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0

Tabla 5.6: Matriz de confusión de los objetos con el sistema propuesto en conjunto con el clasificador 1-NN incluyendo la etapa de verificación.

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**NR**
01	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	2	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	2
08	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	1
09	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	3
10	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1

Tabla 5.7: Tasa de reconocimiento para el método en conjunto con el clasificador HMM utilizando como topología (a) Modelo lineal (b) Modelo de Bakis (c) Modelo de izquiera a derecha.

(a)

Número de estados	Porcentaje de reconocimiento
2	92.5
3	96.5
4	95.4
5	96
6	96

(b)

Número de estados	Porcentaje de reconocimiento
2	92.5
3	96.5
4	95.5
5	96
6	94

(c)

Número de estados	Porcentaje de reconocimiento
2	92.5
3	96.5
4	96.5
5	95.5
6	94.5

Tabla 5.8: Matriz de confusión de los objetos con el sistema propuesto en conjunto con el clasificador HMM incluyendo la etapa de verificación y utilizando los parámatros que generaron la mayor tasa de reconocimiento

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	NR
01	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	2	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	1
08	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
09	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0

Tabla 5.9: Cantidad de observaciones y tiempos de procesamiento promedio por objeto del sistema.

Objeto	**Número de observaciones**	**Tiempo (ms)**	
		Procesamiento	**Clasificador HMM**
01	180	149	421
02	63	46	145
03	169	110	375
04	229	189	489
05	1393	951	3239
06	657	482	1472
07	133	72	307
08	810	733	1802
09	1130	1159	2557
10	288	221	618
11	878	695	2052
12	304	257	721
13	98	67	223
14	125	81	288
15	296	279	663
16	564	493	1342
17	806	752	1884
18	304	261	728
19	168	137	394
20	226	144	520

Una *tabla de confusión* es aquella que reporta el número de *falsos positivos* (FP), *falsos negativos* (FN), *verdaderos positivos* (VP) y *verdaderos negativos* (VN). En la Tabla 5.8 se muestran las *tablas de confusión* para algunas de las pruebas realizadas. Es posible utilizar medidas de evaluación con estos datos que permiten realizar un análisis más detallado.

La ***precisión*** mide el porcentaje de objetos clasificados correctamente y se determina de la siguiente manera:

$$Precision = \frac{VP}{VP + VN + FP + FN} \tag{5.1}$$

El ***recuerdo*** calcula la cantidad de elementos que se clasificaron correctamente de todos los que debería haber encontrado y se determina de la siguiente manera:

$$Recuerdo = \frac{VP}{VP + FN} \tag{5.2}$$

A continuación se muestra la *precisión* y el *recuerdo* del método utilizando el clasificador HMM sin hacer uso de la etapa de verificación:

$$Precision = \frac{193}{193 + 0 + 2 + 5} = 96{,}5\,\%$$

$$Recuerdo = \frac{193}{193 + 7} = 96{,}5\,\%$$

Se presenta además la *precisión* y el *recuerdo* del método con el mismo clasificador haciendo uso de la etapa de verificación:

$$Precision = \frac{193}{193 + 0 + 7 + 0} = 96{,}5\,\%$$

$$Recuerdo = \frac{193}{193 + 2} = 98{,}9\,\%$$

La *precisión* del método utilizando la etapa de verificación propuesta es la misma que la del método haciendo omisión de ella. Sin embargo, el segundo obtiene un peor *recuerdo* debido a la mayor cantidad de falsos positivos generados.

Tabla 5.10: Tablas de confusión para el sistema utilizando el clasificador (a) 1-NN con etapa de verificación (b) HMM con etapa de verificación (c) HMM sin etapa de verificación.

(a)

193 Verdaderos Positivos	5 Falsos Negativos
2 Falsos Positivos	0 Verdaderos Negativos

(b)

193 Verdaderos Positivos	5 Falsos Negativos
2 Falsos Positivos	0 Verdaderos Negativos

(c)

193 Verdaderos Positivos	0 Falsos Negativos
7 Falsos Positivos	0 Verdaderos Negativos

5.5. Análisis de resultados

En las pruebas que se utilizaron nubes de puntos sintéticas se obtuvieron tasas de reconocimiento del 46.6 % con el clasificador 1-NN y del 100 % con el clasificador HMM. Esto es evidencia de que la distribución espacial de las características de los objetos brinda información complementaría que puede ayudar a discriminar entre objetos cuyas representaciones generen distribuciones de frecuencias de características similares.

En las pruebas en las que se utilizaron nubes de puntos de objetos en las que se generaron *símbolos* cuya información era exclusivamente geométrica se obtuvo una tasa de reconocimiento del 47.5 %. Al generar *símbolos* a partir de la combinación de ambas informaciones se obtuvieron tasas de reconocimiento del 94 % con el clasificador 1-NN y del 96.5 % con el clasificador HMM para el caso en el que se obtuvo la mayor tasa de reconocimiento. La topología izquierda-derecha en conjunto con una cantidad de tres o cuatro estados son los parámetros con los que se generó la mayor tasa de reconocimiento para el conjunto de objetos utilizados.

Al considerar únicamente la información que brindan las dimensiones de los objetos, se obtiene una tasa de reconocimiento del 66 %. El uso de esta

misma información en la etapa de verificación en conjunto con la visual y la geométrica se obtienen las mismas tasas de reconocimiento que se lograron anteriormente con la diferencia de que se logra eliminar la mayoría de los *falsos positivos* para el conjunto de objetos utilizados empleando el rango de confiabilidad propuesto.

Cabe destacar que los tiempos de procesamiento y de clasificación para el clasificador HMM son proporcionales al número de observaciones generadas por cada objeto, tal como se puede observar en la Tabla 5.9.

Capítulo 6

Conclusiones y trabajo a futuro

6.1. Conclusiones

Las tasas de reconocimiento obtenidas en las pruebas muestran que las observaciones generadas son adecuadas para su uso en conjunto con los clasificadores propuestos. Sin embargo, los tiempos de procesamiento en los que éstas se obtienen impiden que el reconocimiento se realice en tiempo real, siendo el proceso de reconstrucción de la nube el que consume más tiempo.

La consideración de la distribución espacial de las características puede incrementar la tasa de reconocimiento en algunos casos, lo cual es más evidente en las pruebas que se realizaron en la Sección 5.4.1 en donde el clasificador HMM es superior al clasificador 1-NN debido a que el segundo sólo considera la distribución de frecuencias de las características, por lo cual carece de la información necesaria para poder discriminar entre las nubes de puntos utilizadas. En el caso en el que se trabaja con nubes de puntos de objetos, la tasa de reconocimiento es similar debido a que la propiedad mencionada no es bien aprovechada al tener objetos con distribuciones de frecuencias de las observaciones distintas, incluso cuando se modifican la cantidad de estados y las topología utilizadas. A pesar del incremento en los tiempos de reconocimiento debido al procesamiento adicional necesario para la secuencialización de las observaciones junto con el clasificador HMM, las pruebas indican que el uso de información sobre la distribución espacial de las características generan un incremento en las tasas de reconocimiento con un ajuste adecuado de sus parámetros.

Las pruebas en las que se utiliza únicamente la información sobre las dimensiones de los objetos para el reconociento muestran que, a pesar de la repetibilidad del método para la obtención de las alturas estimadas, el uso exclusivo de esta información puede no ser suficiente para generar buenos resultados debido a que algunos de los objetos generan datos similares. Sin embargo, esta información utilizada en la etapa de verificación logra eliminar efectivamente una gran cantidad de *falsos positivos* con la desventaja de que se pudieran generan además *falsos negativos*. Lo anterior reduciría la tasa de reconocimiento, no obstante, al generar *falsos negativos* es posible realizar diferentes acciones que mejoran la calidad del reconocimiento, entre ellas volver a realizar el proceso con nuevas capturas, clasificar al objeto como desconocido, realizar la verificación con los siguientes modelos que hayan generado la mayor probabilidad, etc.

Las pruebas presentadas son evidencia de que las características visuales y geométricas, la distribución espacial de éstas y las dimensiones de los objetos son atributos que, utilizados en conjunto, generan buenos resultados al usarse en un sistema para el reconocimiento de objetos.

6.2. Trabajo a futuro

- Utilizar sensores que generen datos de mejor calidad, como la segunda versión del $Kinect^{TM}$, para poder prescindir de algunos de los métodos de filtrado y de este manera reducir los tiempos de procesamiento.
- Investigar métodos para la segmentación de los objetos en la escena que no requieran la restricción de que estos se encuentren sobre superficies planas.
- Evaluar el desempeño de otros descriptores geométricos para la generación de símbolos.
- Experimentar con otras maneras de secuencializar la información como el uso de códigos cadena 3D.
- Automatizar el proceso de la búsqueda de los mejores parámetros para el clasificador HMM utilizando alguna técnica para la evaluación de resultados como la validación cruzada.

- Investigar variantes de los modelos ocultos de Márkov como los que utilizan estados silenciosos para evaluar la posibilidad de trabajar con objetos ocluidos.
- Obtener información sobre la posición de los objetos a partir de la información obtenida para su posible manipulación por parte de un robot.
- Integrar la información de las dimensiones de los objetos en los *símbolos* y evaluar otras alternativas que se pudieran utilizar en la etapa de verificación.
- Evaluar la posibilidad de utilizar *cómputos de propósito general en unidades de procesamiento de gráficos* (GPGPU, por sus siglas en inglés: *General-Purpose Computing on Graphics Processing Units*) para realizar procesamiento en paralelo y así poder reducir los tiempos de ejecución.
- Implementar el sistema propuesto en un robot de servicio.

Bibliografía

[1] Hebert Montegranario and Jairo Espinosa. 3d data in computer vision and technology. In *Variational Regularization of 3D Data*, pages 1–4. Springer, 2014.

[2] Scott E Umbaugh. *Digital image processing and analysis: human and computer vision applications with CVIPtools*. CRC press, 2016.

[3] Mohammed Bennamoun and George J Mamic. *Object recognition: fundamentals and case studies*. Springer Science & Business Media, 2012.

[4] Tim Niemueller, Daniel Ewert, Sebastian Reuter, Ulrich Karras, Alexander Ferrein, Sabina Jeschke, and Gerhard Lakemeyer. Towards benchmarking cyber-physical systems in factory automation scenarios. In *Automation, Communication and Cybernetics in Science and Engineering 2013/2014*, pages 665–669. Springer, 2014.

[5] Horst Bunke and Terry Caelli. *Hidden Markov models: applications in computer vision*, volume 45. World Scientific, 2001.

[6] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.

[7] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11):2270–2287, 2014.

[8] Joachim Hornegger and Heinrich Niemann. Probabilistic modeling and recognition of 3-d objects. *International Journal of Computer Vision*, 39(3):229–251, 2000.

[9] Peter M Roth and Martin Winter. Survey of appearance-based methods for object recognition. *Inst. for Computer Graphics and Vision, Graz*

University of Technology, Austria, Technical Report ICGTR0108 (ICG-TR-01/08), 2008.

[10] Jiri Matas and Stepan Obdrzalek. Object recognition methods based on transformation covariant features. In *Signal Processing Conference, 2004 12th European*, pages 1721–1728. IEEE, 2004.

[11] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[12] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[13] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[14] Luís A Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*, volume 1, page 7. Citeseer, 2012.

[15] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[16] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010.

[17] Zoltan-csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3d modelling of novel objects from a single view. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3700–3705. IEEE, 2010.

[18] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010.

[19] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992. IEEE, 2011.

[20] Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. 3d object recognition from range images using local feature histograms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–394. IEEE, 2001.

[21] Joachim Hornegger, Heinrich Niemann, Dietrich Paulus, and Gero Schlottke. Object recognition using hidden markov models. *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, 16:37–44, 1994.

[22] Manuele Bicego and Vittorio Murino. Investigating hidden markov models' capabilities in 2d shape classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):281–286, 2004.

[23] Manuele Bicego, Umberto Castellani, and Vittorio Murino. A hidden markov model approach for appearance-based 3d object recognition. *Pattern Recognition Letters*, 26(16):2588–2599, 2005.

[24] Young Kug Ham and Rae-Hong Park. 3d object recognition in range images using hidden markov models and neural networks. *Pattern recognition*, 32(5):729–742, 1999.

[25] Carmen de Trazegnies, Cristina Urdiales, Antonio Bandera, and F Sandoval. 3d object recognition based on curvature information of planar views. *Pattern Recognition*, 36(11):2571–2584, 2003.

[26] Jeff Kramer, Nicolas Burrus, Florian Echtler, Herrera C Daniel, and Matt Parker. *Hacking the Kinect*, volume 268. Springer, 2012.

[27] Cliff Shaffer. Spatial data structures. `http://algoviz.org/OpenDSA/Books/Everything/html/KDtree.html`. [Online; accesado el 17-Junio-2016].

[28] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[29] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[30] Du Tran and Junsong Yuan. Optimal spatio-temporal path discovery for video event detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3321–3328. IEEE, 2011.

[31] Antonio JR Neves, Rui Garcia, Paulo Dias, and Alina Trifan. Object detection based on plane segmentation and features matching for a service robot. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 10(4):638–645, 2016.

[32] 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*, 24(6):381–395, 1981.

[33] Luz Martínez, Patricio Loncomilla, and Javier Ruiz-del Solar. Object recognition for manipulation tasks in real domestic settings: A comparative study. In *Robot Soccer World Cup*, pages 207–219. Springer, 2014.

[34] Frederic Bretar and Michel Roux. Hybrid image segmentation using lidar 3d planar primitives. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W19):72–78, 2005.

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

[36] KH Lee, H Woo, and T Suk. Point data reduction using 3d grids. *The International Journal of Advanced Manufacturing Technology*, 18(3): 201–210, 2001.

[37] Jianning Wang and Manuel M Oliveira. A hole-filling strategy for reconstruction of smooth surfaces in range images. In *Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on*, pages 11–18. IEEE, 2003.

[38] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158, 1981.

[39] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9 (1):3–15, 2003.

[40] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 264–270. ACM, 2004.

[41] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[42] Samir Sahli, Pierre-Luc Duval, Yunlong Sheng, and Daniel A Lavigne. Robust vehicle detection in aerial images based on salient region selection and superpixel classification. In *SPIE Defense, Security, and Sensing*, pages 80200L–80200L. International Society for Optics and Photonics, 2011.

[43] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[44] Shilpa Bane and DR Pawar. Survey on feature extraction methods in object recognition. pages 3224–3226, 2014.

[45] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, Rudi Penne, and Maarten Weyn. A benchmark survey of rigid 3d point cloud registration algorithms.

[46] Adrian Ford and Alan Roberts. Colour space conversions. *Westminster University, London*, 1998:1–31, 1998.

[47] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2d–3d categorization and classification for multimodal perception systems. *The International Journal of Robotics Research*, 30 (11):1378–1402, 2011.

[48] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on*, 43(5):1318–1334, 2013.

[49] Gary Bradski. The opencv library (2000). *Dr. Dobb's Journal of Software Tools*, 2000.

[50] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[51] Dekang Lin. A c++ implementation of hidden markov model. `http://stp.lingfil.uu.se/ starback/hmm/`. [Online; accesado el 17-Junio-2016].

[52] Gernot A Fink. *Markov models for pattern recognition: from theory to applications*. Springer Science & Business Media, 2014.

[53] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter 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.