VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DESCRIPCIÓN Y DETECCIÓN DE OBJETOS EN 3D POR MEDIO DE CÁMARAS RGB-D

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRA EN INGENIERÍA (COMPUTACIÓN)

PRESENTA:

CECILIA GÓMEZ CASTAÑEDA

TUTOR:

DR. JESÚS SAVAGE CARMONA

FACULTAD DE INGENIERÍA

COTUTOR:

DR. BORIS ESCALANTE RAMÍREZ

FACULTAD DE INGENIERÍA

MÉXICO, D.F. MAYO 2015

AGRADECIMIENTOS

El finalizar este escrito y con él una etapa definitivamente emocionante e importante en mi vida fue gracias a muchas personas que estuvieron involucradas de maneras inimaginables.

Primero quiero agradecer al Dr. Jesús Savage Carmona por dejarme participar en su equipo y hacer posibles algunas de las cosas que siempre quise realizar. En el mismo sentido muchas gracias a Abel Pacheco por apoyarme, orientarme, y ser amigo en todo momento.

Gracias a mis papas Cecilia y Carlos por su paciencia, confianza, apoyo y amor, cada cosa que realizó es dedicada a ellos. A mi hermano por su cariño demostrado en muy diversas maneras. Gracias a mis abuelos que siempre han estado al pendiente de lo que hago, son mis grandes ejemplos de vida.

Erick muchas gracias por tu entusiasmo y apoyo incondicional, por cuidarme, escucharme, y ser la compañía perfecta, te amo.

Gracias familia y amigos que estuvieron al pendiente y emocionados por el proyecto, su entusiasmo fue un gran motor durante este proceso.

Gracias Dra.Nidiyare Hevia , Dr. Fernado Arámbula y Dr. Boris Escalante por aceptar revisar este trabajo y sus valiosas recomendaciones.

También quiero agradecer a Conacyt por el apoyo económico que me fue otorgado para realizar estudios de maestría.

Se agradece a la DGAPA-UNAM por el apoyo proporcionado para la realización de esta tesis a través del proyecto PAPIIT IN117612, "Robot de Servicio para Asistencia a Adultos Mayores y en Sistemas Hospitalarios".

Se agradece al proyecto UNAM-PAPIIT IG100814.

Finalmente quiero agradecer al Posgrado en Ciencia e Ingeniería de la Computación y muy especialmente a Lulú, Cecilia, Amalia y Diana por su amabilidad y asistencia durante toda la estancia.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

La percepción es el proceso por el cual la información recibida por los sistemas sensoriales es procesada e interpretada por el cerebro humano para obtener una representación del mundo, en este sentido, se puede ver a la visión como el sistema de percepción que inicia con la captura de imágenes, en forma de longitudes de onda con diferentes niveles de luminosidad, por medio de los ojos donde se encuentran millones de células receptoras (bastones y conos) que envían información al cerebro el cual nos devuelve la representación del mundo [1]. El estudio de la visión busca saber cómo se extrae a partir de las imágenes una representación del mundo y cómo es esa representación [2].

Por medio del sentido de la vista el ser humano obtiene entre el ochenta y noventa por ciento de la información que percibe [1][3], esto le permite realizar actividades básicas como localizarse en un lugar, reconocer objetos, identificar personas, etc. Radica en lo anterior los innumerables esfuerzos realizados por imitar la visión humana por medio de programas computacionales que tienen como objetivo generar, a partir de imágenes, descriptores de escenarios y modelos del mundo que coadyuven a tomar decisiones útiles [4].

En el área de visión computacional se han desarrollado diversas técnicas y aplicaciones con las que es posible realizar tareas como control de calidad automatizado en procesos industriales [5][6], navegación en robots [7], procesamiento de imágenes médicas [8][9], procesamiento de imágenes satelitales [10], reconocimiento de objetos, análisis de escenarios particulares tales como determinar el área cubierta de nieve por medio de imágenes satelitales [11], realizar consultas a base de datos de imágenes [12],

extraer caracteres en textos o imágenes [13][14], entre otras.

En robótica, la visión computacional juega un papel muy importante, por ejemplo en robots industriales se busca por medio de imágenes determinar si un producto cumple con estándares de calidad para poder desechar aquellos que tengan defectos físicos, incluso los que el ojo humano sea incapaz de identificar a simple vista [15][16]. Por otro lado, en robots de servicio se pretende que sus sistemas de visión computacional les permitan realizar tareas como: localizarse [17], reconocer y tomar objetos [18], reconocer y/o seguir personas [19] [20], entre otras tareas.

El reconocimiento y manipulación de objetos es uno de los principales problemas a tratar en visión computacional aplicada a robots de servicio. En este trabajo se propone un procedimiento para el entrenamiento de objetos, el cual los describe usando la información tridimensional y de color de estos. Dicho procedimiento permitirá generar modelos de manera semi-automática y mejorará el reconocimiento.

El presente trabajo describe el desarrollo de una aplicación que pretende habilitar a un robot de servicio para realizar sus propios modelos de objetos, utilizando información RGB-D, en una fase de entrenamiento.

Los sistemas de visión 3D para reconocer objetos desarrollados en esta tesis son utilizados por la robot Justina, en las tareas que ella ejecuta para ayudar a adultos mayores y para la atención hospitalaria.

A continuación se detallan los objetivos del trabajo y la motivación.

1.1. Objetivos y alcance

El objetivo general del presente trabajo es desarrollar un sistema de detección, entrenamiento y reconocimiento de objetos basado en la información que proveen las cámaras RGB-D, para lo que se siguen los siguiente objetivos particulares.

1. Implementar un procedimiento para realizar la fase de entrenamiento, que in-

cluye obtener un modelo tridimensional a partir de diversas vistas, así como la generación de descriptores 3D y 2D de los objetos.

2. Implementar un método de empatado con el descriptor de formas, el cual reduzca el espacio de búsqueda para la aplicación de un descriptor 2D con lo que se pretende reducir el tiempo de búsqueda y mejorar la tasa de reconocimiento.

3. Generar el despliegue de la nube de puntos con un método de graficación por computadora.

1.2. Motivación

La principal razón de este proyecto es proveer a un robot la capacidad de adquirir modelos del mundo automáticamente con la finalidad de que tenga un mayor conocimiento sobre su entorno. Lo anterior coadyuvaría a lograr un escenario como el siguiente: un robot de servicio entra en una habitación conocida, por ejemplo la sala, la recorre, en ella encuentra un objeto desconocido; si el robot es capaz de tomarlo y automáticamente generar su modelo podrá asociarlo con características como el lugar donde se encontró, su forma, posiblemente hasta su peso, por lo que sin tener asociada una etiqueta obtiene un conocimiento semántico sobre ese objeto que le permitiría conocer más sobre su entorno.

Hoy en día existe tecnología muy poderosa como las cámaras RGB-D y las bibliotecas como OpenCV y PCL. Estas bibliotecas contienen una colección de algoritmos para el procesamiento de imágenes y de nube de puntos respectivamente que se deben de explotar para mejorar el entrenamiento y reconocimiento de objetos lo cual ayudará a hacer posibles escenarios como el que se presentó.

RoboCup es una organización internacional que busca promover la investigación en temas de robótica e inteligencia artificial proponiendo retos en los cuales equipos de universidades de diferentes países del mundo muestran los resultados de sus esfuerzos en competencias anuales que organiza RoboCup. La sección *@Home* de RoboCup busca el desarrollo de robots de asistencia que en un futuro puedan servir en la vida diaria del

ser humano. Para lograr ese objetivo se involucran varias áreas de conocimiento tales como: Interacción y cooperación humano-robot, navegación en ambientes dinámicos, manipulación de objetos, visión computacional, entre otras [21].

El laboratorio de Bioróbotica perteneciente a la Facultad de Ingeniería de la UNAM, compite en la sección *@Home* de RoboCup con su robot Justina, el presente trabajo busca poder implementar un procedimiento que permita a Justina realizar el entrenamiento de objetos de manera autónoma, para presentarlo en dicha competencia.

1.3. Estructura de la tesis

Este trabajo está dividido en siete capítulos. En el capítulo 2 se describe el marco teórico compuesto principalmente por la definición de imagen, el procedimiento de detección y reconocimiento de objetos, así como la descripción de una arquitectura de un sistema de visión computacional.

El capítulo 3 describe los algoritmos más utilizados para realizar la extracción y descripción de características en imágenes 2D. Mientras en el capítulo 4 se exponen algoritmos utilizados para el procesamiento de información 3D.

En el capítulo 5 se describe el sistema propuesto para el entrenamiento, detección e identificación de objetos. Las pruebas y resultados son mostrados en el capítulo 6.

Las conclusiones y trabajo a futuro se encuentran en el capítulo 7. Finalmente se incluye un apéndice en el cual se exponen algunas pruebas de RoboCup que implicaron el desarrollo de funciones que requieren visión computacional.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Imágenes RGB-D

Una imagen es capturada por un dispositivo capaz de sensar ondas ubicadas en cierta región del espectro electromagnético y obtener una representación útil de alguna escena dada. El ejemplo natural de estos dispositivos es el ojo humano el cual percibe longitudes de onda a partir de 400 hz hasta 800 hz, otros ejemplos son las cámaras pinhole y cámaras con sensores CCD (Charge-Coupled Device) o CMOS (Complementary Metal-Oxide Semiconductor).

Los sensores CCD y CMOS transforman la luz en electrones distribuidos en arreglos matriciales que nosotros visualizamos como pixeles para obtener imágenes digitales. Los sensores CCD son analógicos-digitales, cada pixel tiene una salida analógica la cual es convertida a digital por medio de otro circuito, este hecho encarece y hace de mayor tamaño los dispositivos basados en tecnología CCD. Por otro lado, los sensores CMOS digitalizan los pixeles internamente en unos transistores ubicados en cada celda, su precio y tamaño suelen ser mucho menores. Los dispositivos CCD ofrecen mayor calidad de imágenes y robustez ante el ruido, mientras que los CMOS son más veloces [4] [22].

Una imagen digital se describe como un arreglo bidimensional $I[r, c]$ en el cual se almacenan valores discretos llamados pixeles, un pixel es la unidad mínima de información en una imagen. En el caso de imágenes binarias cada pixel toma un valor de 0 o 1 para formar imágenes monocromáticas. En una imagen en escala de grises, a ocho bits, cada pixel almacena valores discretos que van del 0 al 255, los cuales representan

la luminosidad absorbida o reflejada por los objetos contenidos en la escena, se define matemáticamente como una función de dos variables espaciales $I = f(x, y)$ [4] [23].

Una imagen multiespectral es representada por un arreglo n-dimensional en el cual cada pixel es un vector de valores. Por ejemplo, en una imagen RGB cada canal del pixel representa la luz en diferentes longitudes de ondas:

$$f(x, y) = [frojo(x, y), fverde(x, y), fazul(x, y)] \text{ [23]}$$

Una imagen RGB-D proporciona información de una escena tanto de color como de profundidad, es decir, por cada pixel en la imagen se encuentra, además de la información RGB, su ubicación respecto al dispositivo de captura.

Existen diversas técnicas para obtener información 3D de una escena a partir de una serie de imágenes 2D y geometría epipolar. Sin embargo, actualmente se cuenta con dispositivos, como el Kinect Xbox de Microsoft, los cuales, además de capturar información de color, calculan coordenadas tridimensionales de cada pixel en la escena.

2.2. Sistemas de referencia

Para realizar una descripción útil de una escena que contenga posiciones y dimensiones de los objetos de interés que hay en ella, es necesario tomar en cuenta los siguientes sistemas de referencia:

Coordenadas de la imagen: Se refiere al espacio de una imagen digital donde se apunta a cada pixel por medio de su ubicación en un arreglo bidimensional mediante filas y columnas.

Coordenadas del objeto: A cada objeto en la escena se le asigna su propio sistema de referencia, estos son usados para modelar objetos o calcular su posición en aplicaciones tales como visión y graficación por computadora.

Coordenadas de la cámara: Estas coordenadas se usan para el análisis de la escena desde el punto de vista de la cámara. En una escena puede haber una o más cámaras para las cuales cada una tendrá su propio sistema de referencia.

Coordenadas reales de la imagen: Es una representación por la cual se pueden describir la dimensión y ubicación de los objetos por medio de coordenadas del tipo $[x_f, y_f, f]$ donde f es la distancia focal. x_f y y_f hacen referencia al tamaño del pixel y su posición del eje óptico en la imagen.

Coordenadas del mundo: Estas coordenadas relacionan directamente la escena con sus dimensiones en el mundo, son útiles para el modelado 3D de objetos.

Por medio de relaciones geométricas y transformaciones se puede pasar de un sistema de referencia a otro. Szeliski y Shapiro exponen en [22] y [24] información más amplia al respecto. La figura 2.1 ilustra los cinco sistemas de referencia descritos anteriormente.

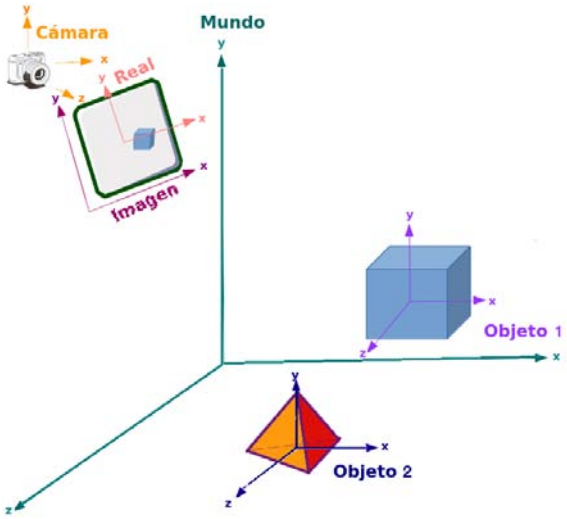


Figura 2.1: Sistemas de referencia [24]

2.3. Detección y reconocimiento de objetos

El reconocimiento de objetos es un campo de gran interés para el desempeño de robots de servicio. Dada una imagen, comúnmente se siguen los siguientes pasos para lograr el reconocimiento:

1. Suponiendo que los objetos se encuentran en mesas o cualquier tipo de superficie plana, se restringe la zona de interés en una escena. Dicha restricción se realiza aplicando un filtro de caja sobre el plano localizado a partir de la información tridimensional de la escena, utilizando algoritmos como RANSAC(Random Sample Consensus) el cual será descrito más adelante.

2. Realizar la búsqueda de características en la zona de interés, para lo cual se utilizan algoritmos de extracción y descripción de características que Szeliski [22] describe en cuatro etapas:

   2.1. **Detección de puntos de interés:** En cada imagen se realiza la búsqueda de puntos de interés, también llamados: puntos o localidades característicos. Estos puntos se obtienen a partir de la información 2D o 3D de una imagen a partir de características tales como color, textura o forma.

   Una propiedad deseable de un punto de interés es que sea repetible, es decir, que se pueda extraer la misma característica con diferentes entradas del mismo modelo.

   La extracción de buenos puntos de interés es fundamental para aplicaciones como reconocimiento de objeto y reconstrucción de escenas, entre otras.

   2.2. **Descripción de puntos de interés:** Por cada punto de interés detectado se genera un descriptor, normalmente utilizando la información de su vecindad. Con el propósito de contar con una buena robustez se desea que los descriptores presenten las siguientes características:

   Unicidad: cada punto de interés debe tener una representación única.

   Robustez: capacidad de mantenerse frente a transformaciones como traslaciones, rotaciones y ante cambios de escalas e iluminación.

   Sensibilidad: capacidad para diferenciar objetos casi iguales.

Abstracción: capacidad para representar los rasgos característicos de los objetos.

Además se busca que la dimensión de los descriptores sea lo más pequeña posible con el propósito de disminuir el tiempo de empatado sin comprometer la capacidad de distinguirse al momento de ser contrastado con otros descriptores.

2.3. **Empatado de descriptores:** Consiste en realizar búsquedas eficientes capaces de encontrar coincidencias en otras imágenes. Normalmente basadas en árboles de búsqueda K-d [25, p. 28].

2.4. **Seguimiento de características**: Este paso es utilizado en el análisis y procesamiento de video. Consiste en seguir un punto característico en diferentes capturas del vídeo con la información de su vecindad.

Existe una gran variedad de detectores y descriptores de características que contribuyen a resolver el problema del reconocimiento de objetos. Algunos ejemplos utilizados para el procesamiento de imágenes 2D son: SIFT (Scale Invariant Fetaure Transform) y SURF (Speeded-Up Robust Features), mientras que para el análisis 3D se utilizan algoritmos como SHOT (Signature of Histograms of OrienTations) y PFH (Point Feature Histogram), los cuales son descritos en los capítulos 3 y 4.

2.4. Arquitecturas de visión por computadora

Muja propone en [26] la arquitectura REIN (REcognition INfrastructure) con el propósito de aumentar la robustez en los sistemas de visión computacional frente a problemas encontrados en ambientes reales, donde se tiende a encontrar diferencias desde texturas y superficies hasta cambios de iluminación y ambientes ocluidos o desordenados.

REIN propone implementar unas serie de detectores, estimadores de pose y filtros con la finalidad de utilizar la combinación de métodos según convenga para lograr mejores resultados según el ambiente, esto se logra manejando cada algoritmo como un

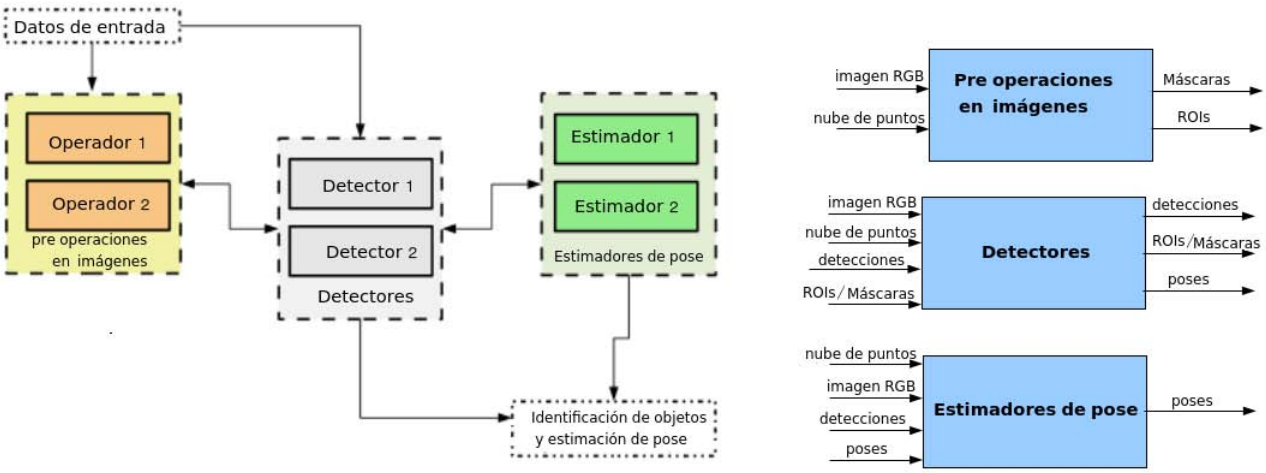


Figura 2.2: Arquitectura REIN [26]

plug-in, esta arquitectura se encuentra disponible para su uso en ROS (Robot Operating System).

Finalmente en este artículo se demuestra que usar una combinación de algoritmos enfocados al reconocimiento de objetos utilizando información 2D y 3D de manera híbrida aumenta la tasa de reconocimiento. Basándose en esta idea, se propone un sistema que permita mezclar varias técnicas fácilmente con el fin de obtener mejores resultados.

CAPÍTULO 3

DESCRIPCIÓN DE OBJETOS CON IMÁGENES RGB

La descripción de objetos en imágenes ha sido muy estudiada desde los inicios de visión por computadora. Existen algoritmos para este fin basados en histogramas de color, textura, extracción y descripción de características. En este capitulo se describen los algoritmos SIFT (Scale Invariant Feature Transform) y SURF (Speed-Up Robust Features) basados en la extracción y descripción de características.

3.1. SIFT

SIFT fue presentado por Lowe en [27] [28] como una técnica que transforma una imagen en una colección de vectores de características locales las cuales son invariantes a traslaciones, escalamientos, rotaciones y parcialmente a cambios de iluminación y puntos de vista. Cabe mencionar que SIFT es uno de los detectores y descriptores más usados para realizar tareas relacionadas con visión computacional, el éxito de SIFT recae en extraer características que se pueden localizar tanto en el dominio espacial como en él de la frecuencia aumentando su robustez ante oclusiones, ruido y ambientes desordenados.

SIFT consta de cuatro etapas, las cuales se explican a continuación con base en [28]:

1. Detección de extremos en el espacio-escala

El objetivo de esta etapa es encontrar posibles puntos de interés identificando localidades y escalas que puedan ser repetiblemente asignadas bajo diferentes vistas. Se

detectan localidades invariantes a cambios de escala buscando características estables a través de todas las escalas posibles, usando una función continua conocida como espacio-escala (espacio de escalas).

Las localidades invariantes al escalamiento son buscadas en el espacio-escala gaussiano de una imagen $L(x, y, \sigma)$, el cual se define como la convolución entre la imagen y una función gaussiana.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.1}$$

donde:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3.2}$$

Los puntos de interés se seleccionan usando los extremos del espacio-escala en la función DoG convulsionada con la imagen, $D(x, y, \sigma)$, lo cual es calculado a partir de la diferencia de dos escalas cercanas separadas por un factor k.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{3.3}$$

Un ejemplo para la construcción de $D(x, y, \sigma)$ se muestra en la figura 3.1. La imagen inicial es convulsionada con una función gaussiana para producir imágenes separadas por un factor k en el espacio-escala (columna izquierda). Se divide cada octava en un número entero s de intervalos, tal que $k = 2^{1/s}$. Se producen $s+3$ imágenes en la pila de imágenes suavizadas por cada octava para asegurar que la detección de extremos cubra la octava completa. Las imágenes adyacentes escaladas son restadas para producir DoG (columna de la derecha). Una vez que la octava es procesada, se vuelve a muestrear la

imagen gaussiana, la cual tiene dos veces el valor inicial de σ, tomando cada segundo pixel en cada fila y columna.

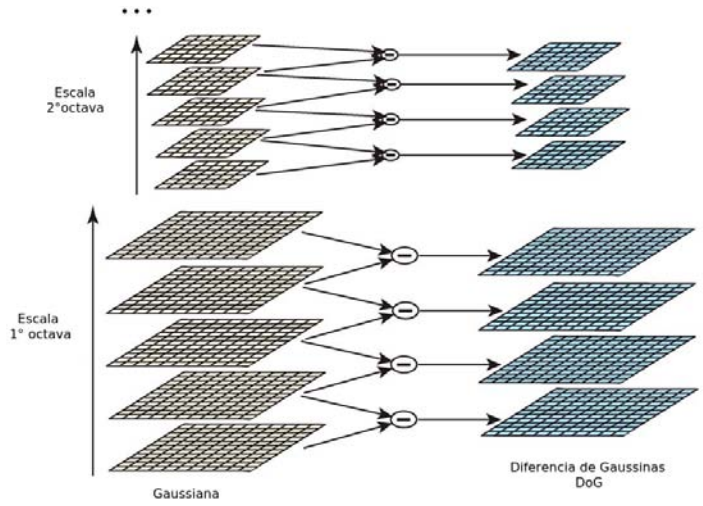


Figura 3.1: SIFT: Detección de extremos en escala-espacio [28]

2. Localización de puntos de interés

Para detectar el máximo y mínimos local de $D(x, y, \sigma)$ cada punto es comparado con sus 8 vecinos de la imagen actual y los nueve vecinos tanto de las imágenes escaladas arriba y abajo, respectivamente. El punto es seleccionado si es el mayor de sus vecinos o el menor.

Una vez seleccionados los máximos y mínimos, se realiza un estudio de estabilidad para rechazar aquellos puntos que son pobremente localizados en bordes o con bajo contraste.

Los puntos con bajo contraste se eliminan bajo un proceso de umbralización. Utilizando la expansión de la serie de Taylor de la función escala espacio $D(x, y, \sigma)$ desfasada para que el punto de origen sea el punto que se está analizando.

$$D(\mathbf{x}) = D + \frac{\delta D^T}{\delta \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x^T}\frac{\delta^2 D}{\delta \mathbf{x}^2}\mathbf{x} \tag{3.4}$$

donde D y sus derivadas son evaluadas en el punto analizado y $\mathbf{x} = (x, y, \sigma)^T$ es el desplazamiento del punto. La localidad del extremo $\hat{\mathbf{x}}$ es calculada a partir la derivada de esta función con respecto a $\mathbf{x}$ mandándolo a 0, dando

$$\hat{\mathbf{x}} = -\frac{\delta^2 D^{-1}}{\sigma \mathbf{x}^2}\frac{\delta D}{\delta \mathbf{x}} \tag{3.5}$$

La función $D(\hat{\mathbf{x}})$ valuada en el extremo $\hat{\mathbf{x}}$ es usada para rechazar extremos inestables con bajo contraste, quedando

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\delta D^T}{\delta \mathbf{x}}\hat{\mathbf{x}} \tag{3.6}$$

Lowe sugiere en su artículo rechazar aquellos puntos extremos donde $D(\hat{\mathbf{x}}) < .003$.

El siguiente paso es eliminar aquellos puntos que pertenecen a bordes que no son lo suficientemente localizables. Un pico pobremente definido por DoG podría tener una gran curvatura principal a través del borde pero una pequeña en la dirección perpendicular. La curvatura principal puede ser calculada con una matriz hessiana de $2x2$ evaluada en la posición y escala del punto de interés.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Las derivadas son estimadas tomando diferencias entre vecinos del punto de interés.

Los eigen valores de H son proporcionales a sus curvaturas principales por lo que se evade el cálculo de los eigen valores y se pone atención en su radio.

Definiendo α como el eigen valor de mayor magnitud y β como el de menor. Entonces, se puede calcular la suma de los eigen valores del rasgo de H y su producto a partir del determinante:

$$\begin{aligned} Tr(H) &= D_{xx} + D_{yy} = \alpha + \beta, \\ Det(H) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta; \end{aligned} \tag{3.7}$$

Definiendo r como el radio entre el eigen valor de mayor magnitud y el de menor entonces $\sigma = r\beta$ por lo que

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{3.8}$$

$\frac{(r+1)^2}{r}$ toma valores mínimos cuando los dos eigen valores son iguales e incrementa con r. Para evaluar si el radio está por debajo de un umbral r se realiza la siguiente comparación, rechazando los puntos que no satisfagan la desigualdad.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{3.9}$$

3. Asignación de orientaciones

Para cada punto de interés se asigna una o más orientaciones basadas en direcciones locales de gradientes en la imagen. Con esta asignación se pretende mantener la invarianza a rotaciones.

La escala del punto seleccionado es usada para seleccionar la imagen suavizada L con la escala más cercana. En esta imagen para cada pixel L(x,y) se calcula un gradiente de magnitud y otro de orientación usando sus vecinos:

$$m(x,y) = \sqrt{L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{3.10}$$

$$\theta(x,y) = tan^{-1}((L(x,y+1) - (L(x,y-1))/(L(x+1,y) - L(x-1,y))) \qquad (3.11)$$

Se crea un histograma de orientaciones de 36 casilleros que cubre los 360 grados. Cada muestra agregada al histograma es ponderada por su gradiente de orientación y por una ventana circular gaussiana ponderada con σ que es 1.5 veces la escala del punto seleccionado.

El casillero con mayor concurrencia es asignado como una dirección dominante, posteriormente los casilleros con un valor de más del 80 % del valor de la magnitud principal se considerará como dirección dominante. El tener más de un valor de orientación hace que el punto de interés tenga mayor estabilidad.

4. Descripción de puntos de interés

En este punto se tiene una colección de puntos de interés los cuales están formados por una localidad, una escala y una orientación. Así que el siguiente paso es crear un descriptor para cada punto que sea lo suficientemente distintivo.

Cada punto es ponderado por su magnitud de gradiente y una función 3D gaussiana para evitar cambios bruscos en el descriptor y asignar menos énfasis a los puntos más alejados del punto de interés. El valor σ de la función gaussiana se fija como 1.5 veces el tamaño de la región de cálculo para el punto de interés. Se analizan las muestras de cada región de 16×16 formando histogramas de orientaciones de 4×4 como se ilustra en la figura 3.2:

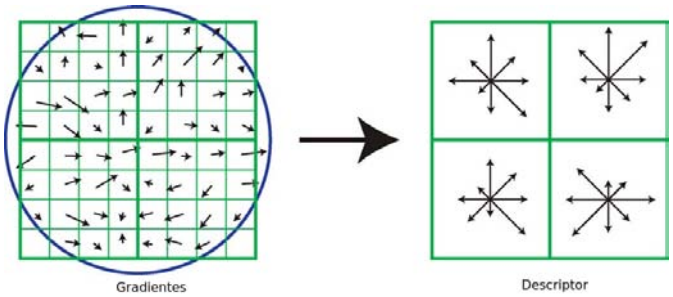


Figura 3.2: SIFT: Generación del descriptor [28]

Los histogramas están compuestos por 8 casilleros que almacenan las orientaciones posibles proporcionales a 45° donde la magnitud de cada flecha es proporcional al valor acumulado por cada casillero. Así se obtienen 16 histogramas respecto de las orientaciones de los puntos de cada región para cada uno de los puntos de interés.

Finalmente el descriptor de cada punto de interés está formado por un vector que contiene los valores de las 8 orientaciones de los 4×4 histogramas componiendo un vector de características de $4 \times 4 \times 8 = 128$ elementos.

3.2. SURF

SURF propuesto en [29] por Bay en 2006, es un detector y descriptor invariante a escalas y rotaciones, el cual busca diferenciarse de los demás por encontrar, describir y empatar puntos distintivos con mayor rapidez, tomando en consideración el hecho de que el tamaño del descriptor tiene un impacto directo en esta tarea y sin perder de vista que los descriptores de menor tamaño suelen ser menos distintivos.

A continuación se describe el algoritmo en base a lo publicado por Bay en [29]:

1. Detección basada en matriz hessiana

El detector SURF está basado en la matriz hessiana debido a su buen rendimiento en términos de tiempo de cómputo y exactitud. En SURF la búsqueda de localidades y escalas está determinada por el determinante de la matriz hessiana, la cual dado un punto $\mathbf{x} = (x, y)$ en una imagen I se define como la matriz hessiana, $H(x, \sigma)$, en el punto $\mathbf{x}$ en el escala σ:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{3.12}$$

Donde $L_{xx}(\mathbf{x}, \sigma)$ es la convolución de la derivada de segundo orden de la función

gaussiana $\frac{\delta^2}{\delta x^2} g(\sigma)$ con la imagen I en el punto $\mathbf{x}$. Lo mismo se aplica para $L_{xy}(\mathbf{x}, \sigma)$ y $L_{yy}(\mathbf{x}, \sigma)$.

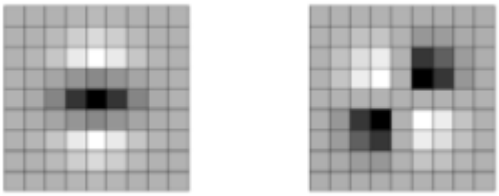
Partiendo del hecho de que el análisis de imágenes bajo filtros gaussionos resulta óptimo para detectar puntos de interés en el espacio-escala, SURF propone aproximar un filtro gaussiano mediante filtros de caja, los cuales se aproximan a la derivada gaussiana de segundo orden. Los filtros de caja tienen la ventaja de poder ser evaluados muy rápidamente usando imágenes integrales independientemente del tamaño.

Las imágenes integrales coadyuvan a realizar una computación más rápida para el cálculo de convoluciones de filtros tipo caja. La entrada de una imagen integral $I_{\sum}(\mathbf{x})$ en la localidad $\mathbf{x} = (x, y)^T$ representa la suma de todos los pixeles en la imagen de entrada dentro de una región rectangular formada por el punto $\mathbf{x}$ y el origen.

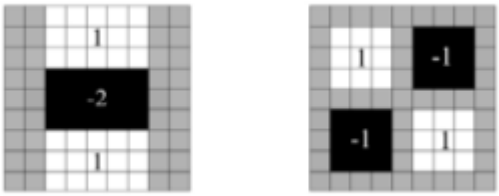
$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{3.13}$$

Con I_{Σ} calculada sólo se necesitan cuatro sumas para calcular la suma de intensidades sobre cualquier área vertical o rectangular, independientemente del tamaño.

Los cuadros de 9×9 que se muestran en la figura 3.3 son aproximaciones a la derivada parcial de segundo orden de una gaussiana con valor de $\sigma = 1.2$ que es considerado como la mínima resolución espacial para obtener los bloques de respuesta, los cuales son denotadas por D_{xx}, D_{yy} y D_{xy}. Dichas respuestas son almacenadas en un mapa de respuestas sobre diferentes escalas, donde las localidades máximas son detectadas.



(a) ***Derivada parcial de la función gaussiana de segundo orden en las direcciones*** $y-$ (L_{yy}) ***y*** xy (L_{xy}) ***respectivamente***



(b) ***Aproximación de la derivada gaussiana utilizando la matriz hessiana con los filtros de caja en las direcciones*** $y-$ (D_{yy}) ***y*** xy (D_{xy}) ***respectivamente***

Figura 3.3: SURF: Detector Hessiano.[29]

Los pesos aplicados a las regiones rectangulares son mantenidos con el fin de hacer la computación más eficiente, sin embargo, para obtener la determinante de la matriz hessiana los pesos relativos son balanceados, $w = \frac{|L_{xy}(1.2)|_F|D_{yy}(9)|_F}{|L_{yy}(1.2)|_F|D_{xy}(9)|_F} = 0.912... \simeq 0.9$, donde $|x|_F$ es la norma de Frobenious quedando:

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{3.14}$$

A la vez los resultados del filtro son normalizados con respecto al tamaño de la máscara para garantizar una norma constante de Frobenius para cualquier tamaño de filtro.

El espacio-escala es usualmente implementado como pirámides de imágenes donde las imágenes son repetidamente suavizadas con una función Gaussiana para construir cada nivel hacia arriba. Debido al uso de filtros de caja e imágenes integrales en SURF no se tiene que aplicar el mismo filtro a la imagen de salida del filtrado anterior, si no que, se aplican dichos filtros de cualquier tamaño a la misma velocidad directamente en la imagen original. Además la escala espacio es analizada escalando el tamaño del filtro hacia arriba en vez de reducir iterativamente el tamaño de la imagen.

La salida del filtro aplicado de 9×9 es considerada la capa de la escala inicial, equivalente a la derivada gausiana con $\sigma = 1.2$. Las siguientes capas son obtenidas filtrando la imagen con filtros gradualmente de mayores. Los filtros usados son de $9 \times 9, 15 \times 15, 21 \times 21, 27 \times 27$, etc. En cada octava, el tamaño del filtro es doblado (quedando de 6 a 12 a 24). Simultáneamente, los intervalos de muestreo para la extracción de puntos de interés pueden ser doblados.

Debido a que el radio del filtro base permanece constante después de ser escalado, la derivada gaussiana escala de manera equivalente. Por ejemplo, el filtro de 27×27 corresponde $\sigma = 3.6$.

Por otra parte la norma de Frobeniuis se mantiene constante para los diferentes tamaños de filtros, por lo que estos se mantienen normalizados respecto a la escala.

Para localizar los puntos de interés en la imagen y sobre todas las escalas, se realiza una supresión de no máximos en una vecindad de $3 \times 3 \times 3$. Los máximos del determinante de la matriz Hessiana son interpolados en el espacio escala de la imagen con el método propuesto por Brown. La interpolación es importante debido a que la diferencia en escalas entre las primeras capas de cada octava es relativamente grande.

Descriptor SURF

El primer paso consiste en encontrar una orientación basada en la información de la vecindad del punto de interés, donde posteriormente se extrae una región cuadrada en base a la orientación elegida para extraer el descriptor.

La orientación se calcula mediante respuestas en las direcciones x e y de una ventana

de Haar en una vecindad circular con radio igual a $6s$ del punto de interés, donde s es la escala donde el punto de interés fue detectado. Las respuestas de la ventana también son escaladas con s, de esta manera las ventanas son más grandes a mayores s. Se utilizan imágenes integrales para calcular el filtro rápidamente.

Una vez que las respuestas son calculadas, éstas son representadas mediante un espacio de vectores donde la respuesta horizontal es asignada a la abscisa y la respuesta vertical corresponde a la ordenada. Finalmente la orientación es estimada calculando la suma de todas las respuestas dentro de una ventana deslizante de orientación la cual cubre un angulo de $\frac{\pi}{3}$. Una vez que las respuestas horizontales y verticales son sumadas se obtiene un nuevo vector, la orientación es asignada donde se encuentra el vector de mayor dimensión. El descriptor está basado en una región cuadrada, a la que se le asigna la orientacion obtenida en el paso previo y es centrada en el punto de interés.

Se selecciona la región de tamaño $20s$ la cual es dividida en 4×4 sub-regiones, para cada sub-región se calculan un número reducido de características separadas por espacios de 5×5. Llamando d_x a las respuestas de la ventana de Haar en dirección horizontal y d_y a las respuestas en dirección vertical, donde horizontal o vertical se define según la orientación asignada al punto de interés. Para incrementar la robustez ante deformaciones geométricas y errores de localización, las respuestas d_x y d_y son ponderadas con una gaussiana ($\sigma = 3.3.s$) centrada en el punto de interés.

Finalmente las respuestas d_x y d_y son sumadas para cada sub-región formando un primer conjunto del vector de características. Los valores absolutos de las respuestas $|d_x|$ y $|d_y|$ agregan robustez al descriptor ante cambios de intensidad y polaridad. Por último cada sub-región tiene un vector v cuatro-dimensional $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, formando el descriptor para las 4×4 sub-regiones.

CAPÍTULO 4

DESCRIPCIÓN DE OBJETOS 3D

La extracción de características a partir de información tridimensional permite obtener ventajas tanto en el reconocimiento de objetos como en la planeación de toma de objetos. En esta sección se exponen conceptos relacionados con la información tridimensional de un objeto, en seguida se explican algunos métodos utilizados para realizar la descripción de puntos de interés en nubes de puntos (PHF, FPFH ,RBD, SHOT) y finalmente se describe un procedimiento comúnmente seguido para realizar la extracción y empatado de características en nubes de puntos.

Tomando la notación expuesta en [30] una nube de puntos es un conjunto de n puntos $P = \{p_1, p_2, ..., p_n\}$. Cada p_i por su parte contiene un conjunto de m valores característicos $p_i = \{f_1, f_2, ...f_m\}$, por ejemplo $p_i = \{\mathbf{p}_i, \mathbf{n}_i\}$ donde $\mathbf{p}_i = [x_i, y_i, z_i]$ representa la posición 3D del punto y $\mathbf{n}_i = [n_{xi}, n_{yi}, n_{zi}]$ su vector normal.

Los k puntos más cercanos (vecindad) de un punto p_i se define como el subconjunto P^K de puntos p_j, $(j\epsilon\{1...k\})$, donde $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq r$, leyéndose $\|\cdot\|_2$ como la distancia euclidiana y r es un radio dado de una esfera.

Los descriptores de nubes de puntos se pueden clasificar en locales y globales:

Los **descriptores locales** asocian cada punto de interés identificado en la nube de puntos con una descripción que involucra la geometría local del punto respecto a sus vecinos [31]. Estos descriptores son utilizados en aplicaciones tales como registro y reconocimiento de objetos, así como para realizar clasificación local de superficies. Algunos ejemplos son descritos en la tabla 4.1 [32]:

Descriptor	Tamaño	Entradas	Notas
PHF: Point Feature Histogram [33] [34].	16 y 125 en implementación de PCL [1].	Nube de puntos de interés, normales, método de búsqueda y radio de búsqueda.	Considera las relaciones angulares entre pares de puntos y también toma en cuenta la vecindad.
FPFH: Fast Point Feature Histogram [36].	125 y 33 en la implementación de PCL	Nube de puntos de interés, normales, método de búsqueda y radio de búsqueda.	Simplifica PFH no tomando en cuenta las relaciones entre vecinos.
SHOT: Signature of Histograms of Orientations [37]	352 en implementación de PCL	Nube de puntos de interés, normales y radio de búsqueda.	Representa rasgos topológicos tomando en cuenta puntos que se encuentran en una estructura de soporte esférico. Es invariante a rotaciones y traslaciones, además de presentar robustez ante ruido y desorden.
3DSC: 3-D Shape Context [38]	1980	Nube de puntos de interés, normales, método de búsqueda, radio de búsqueda, radio mínimo y densidad de puntos en el radio.	Es una extensión del descriptor en 2D, utiliza una malla esférica dividida por sectores al rededor de cada dimensión. Su desventaja principal es que la dimensión del descriptor es muy grande.
USC: Unique Shape Context [39]	1960	Nube de puntos de interés, radio de búsqueda, radio mínimo, densidad de puntos en el radio y radios locales.	Extiende 3DSC, provee invarianza ante rotaciones dando una orientacion única a cada descriptor.
RSD: Radius-Based Surface Descriptor [40].	289 en implementación de PCL	Nube de puntos de interés, normales, método de búsqueda, radio de búsqueda y radio máximo.	Estima diferencia de radios entre pares de puntos, suponiendo que cada par de puntos se encuentra en una superficie esférica.
SI: Spin Images [41]	153	Nube de puntos de interés, normales, radio de búsqueda y resolución de imagen.	Usa un soporte cilíndrico basado en una alineación de normales dividido en volúmenes.

Tabla 4.1: Descriptores locales de nubes de puntos

Los **descriptores globales** son representaciones que extraen características geométricas de objetos completos por lo que requieren un proceso previo de segmentación para extraer de una escena la nube de puntos que representa al objeto de interés [31].

Estos descriptores son utilizados en aplicaciones tales como reconocimiento de objetos, clasificación geométrica y recuperación de formas, algunos ejemplos se mencionan en la siguiente tabla 4.2 [32]:

[1] Point Cloud Library[35]

Descriptor	Tamaño	Entradas	Notas
VFH: Viewpoint Feature Histogram [42]	263 y 308 en implementación de PCL	Nube de puntos de interés, normales y método de búsqueda.	Considera el ángulo que se forma entre la normal de un punto y la del punto central de la nube.
CVFH: Clustered Viewpoint Feature Histogram [43]	308	Nube de puntos de interés, normales, método de búsqueda y umbrales de ángulo y curvatura.	Calcula descriptores VFH en N regiones disjuntas del objeto lo que le da robustez ante oclusiones.
ESF: Ensemble of Shape Functions [44]	640	Nube de puntos	Describe la nube de puntos mediante distancias, ángulos, áreas. Robusto ante el ruido.

Tabla 4.2: Descriptores globales de nubes de puntos

4.1. Histograma de Características de Punto

El Descriptor de Histograma de Características de Punto (PFH por sus siglas en inglés) es un descriptor local de nubes de puntos propuesto por Rusu en los artículos [33] y [34]. PFH está compuesto por restricciones geométricas, además de asociar a cada punto etiquetas informativas tales como: punto en borde, punto en esfera, punto en plano con el objetivo de generar descriptores que disminuyan el número de falsos empates cuando se buscan correspondencias punto a punto.

PFH genera un histograma de valores en el cual se codifican las propiedades geométricas de la vecindad de cada punto característico, dicho histograma provee invariaza ante cambios de escala y pose. Para formar el histograma se calcula la curvatura promedio para cada punto p.

1. Para cada punto p, se seleccionan todos los P_k vecinos que se encuentran dentro de una esfera de radio r y se obtiene su normal n_i, si no existe n_i se aproxima con la normal del plano que mejor se ajuste a la superficie de la vecindad del punto p utilizando análisis de componentes principales:

$$n_i = V_0, \lambda_0 \leq \lambda_1 \leq \lambda_2$$

donde V_0 es el eigenvector correspondiente al eigenvalor más pequeño λ_0.

2. Con la información del punto de vista v se orienta n_i:

$$\text{Si } \frac{\langle v-p_i, n_i\rangle}{\|v-p_i\|} < 0, \text{ entonces } n_i = -n_i.$$

3. Para cada par de puntos p_i y p_j $(i \neq j, j < i)$ en la vecindad del punto p, y sus normales estimadas n_i y n_j, se nombra como punto origen p_s al punto que tiene un menor ángulo entre su normal asociada y la linea que conecta ambos puntos, mientras que al otro punto se le llama punto modelo p_t:

$$\begin{gathered}\text{Si } \langle n_i, p_j - p_i\rangle \leq \langle n_j, p_i - p_j\rangle \\ \text{entonces } p_s = p_i, p_t = p_j \\ \text{de lo contrario } p_s = p_j, p_t = p_i\end{gathered}$$

con lo que se construye un triedro de Darboux, como el que se muestra en la figura 4.1, con el origen en p_s:

$$u = n_s, v = (p_t - p_s) \times u, w = u \times v$$

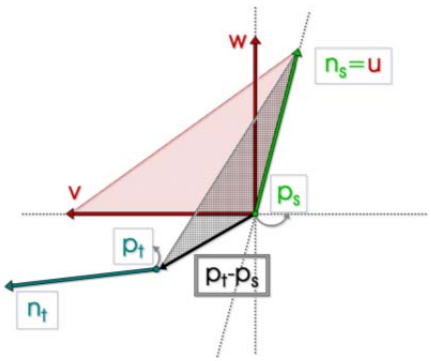


Figura 4.1: Triedro de Darboux(vectores u, v,w) colocados en p_s [34]

4. A partir de p_s, p_t, n_s, y n_t se calculan cuatro características que son categorizadas usando un histograma de 16 casilleros, cada casillero en el índice idx contiene el porcentaje de los puntos origen que caen dentro de su umbral.

$$\left.\begin{array}{rcc} f_1 & = & \langle v, n_t \rangle \\ f_2 & = & ||p_t - p_s|| \\ f_3 & = & \langle u, p_t - p_s \rangle / f_2 \\ f_4 & = & atan(\langle w, n_t \rangle, \langle u, n_t \rangle) \end{array}\right\} idx = \sum_{i=1}^{i \leq 4} step(s_i, f_i) \cdot 2^{i-1}$$

donde $step(s, f)$ se define como 0 si $f < s$ y 1 de lo contrario.

Estableciendo s_i en el centro del intervalo de f_i, es decir 0 para f_1, f_3, f_4 y r para f_2 el algoritmo clasifica cada par de puntos en la vecindad $\{p_i, p_j\}$ en dos categorías, y guarda el porcentaje de pares de puntos que tienen la misma categoría para las cuatro características.

Las características propuestas son medidas de los ángulos entre las normales de los puntos y el vector de distancia que se forma entre ellos. El número de casilleros en el histograma, usando estas cuatro características geométricas, es de div^4 donde div es el número de subdivisiones dado por los rangos de cada característica. En [33] menciona que usando un número de 2 subdivisiones obtuvieron buenos resultados con $2^4 = 16$ casilleros. Mientras en [34] se notan buenos resultados dividiendo los valores característicos en tres partes, obteniendo un total de $3^4 = 81$ casilleros. Como se observa aumentar el valor de div incrementa exponencialmente el número de casilleros por lo que se busca encontrar buenos resultados con el valor de div lo más pequeño posible.

La complejidad computacional de PFH dada una nube de puntos P con n puntos es de $O(n \cdot k^2)$, donde k es el número de vecinos para cada punto p_i en P [36].

4.2. Histograma de Características de Punto Rápido

Histograma de características de punto rápido (FPFH por sus siglas en inglés) es presentado en [36] por Rusu como una optimización de PFH que reduce el costo computacional y guarda descriptores más discriminatorios.

Con el fin de simplificar el cálculo del histograma se sigue el siguiente procedimiento:

Para cada punto p_i sólo se calculan las relaciones entre éste y sus vecinos obteniendo un Histograma Característico Simplificado (SPFH).

Posteriormente para cada uno se vuelven a evaluar sus k vecinos tomando los valores de SPFH para asignarle un peso al histograma final FPFH de p_i:

$$FPFH(p) = SPF(p) + \frac{i}{k} \sum_{i=1}^{k} \frac{1}{w_k} \cdot SPF(p_k) \tag{4.1}$$

donde el peso w_k representa la distancia entre el punto p_i y uno de sus vecinos p_k en un espacio dado.

La figura 4.2 muestra la región de influencia de un punto p_q (punto central rojo). Dado un p_q se calcula SPFH creando pares entre p_q y sus k-vecinos, una vez que todos los puntos son evaluados se crea FPFH.

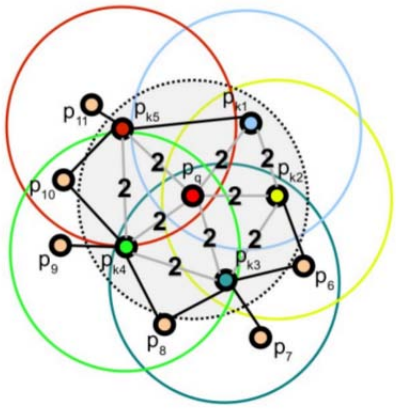


Figura 4.2: Diagrama de región de influencia de p_q en P para FPFH [36]

La complejidad computacional para FPFH es de $O(n \cdot k)$ por lo que es más rápido que su antecesor PFH, lo cual no compromete la distinción entre descriptores.

4.3. Descriptor de Superficies Basado en Radios

El descriptor de superficies basado en radios (RSD, por sus siglas en inglés) es presentado en [40] por Zoltan-Csaba. Dicho artículo aborda el problema de estimar modelos 3D de objetos, para la reconstrucción y planeación de toma de objetos. RSD toma en consideración el hecho de que la mayoría de objetos utilizados diariamente son simétricos.

RSD [40] [45], describe la geometría de un punto p_i estimando el radio de las curvaturas que caen en su vecindad. El valor característico de cada punto p_i es conformado por las curvaturas máxima y mínima, las cuales se obtienen calculando la distancia entre el punto y cada punto de su vecindad formada por la siguiente relación:

$$d(\alpha) = \sqrt{2r}\sqrt{1 - cos\alpha} \tag{4.2}$$

Dada una distancia y el ángulo formado entre dos normales de puntos, se puede aproximar el radio de la superficie donde se encuentra en punto.

En el caso de una esfera esta distancia siempre es igual a r, en el caso de un plano la estimación del radio siempre es infinito para todos los puntos debido a que se tienen sólo normales paralelas. Un punto en un cilindro estará en muchos círculos por lo que el radio estimado varía. Para esquinas y bordes el radio estimando cambia de manera similar que esferas y cilindros.

En la figura 4.3 se observan los histogramas 2D de puntos en diferentes superficies que representan las diferencias de ángulos y distancia de dichos puntos a sus vecinos. De izquierda a derecha corresponden a punto en plano, esfera, esquina, borde y tres cilindros respectivamente.

Dado un punto en una superficie y sus vecinos, el radio mínimo y máximo pueden ser estimados usando el modelo en la ecuación 4.2 resolviendo el sistema de ecuaciones para r a diferentes intervalos de distancia.

Figura 4.3: Histogramas 2D de descriptores RSD, en diferentes superficies [40]

Esta característica es fácil de computar, es bastante descriptiva y no requiere normales orientadas consistentemente.

La imagen 4.4 muestra el resultado de la estimación de los radios en diferentes nubes de puntos. Ilustrando los radios en un código de color que va del rojo a azul, donde el valor menor se asocia a un tono rojo y el mayor a uno azul.

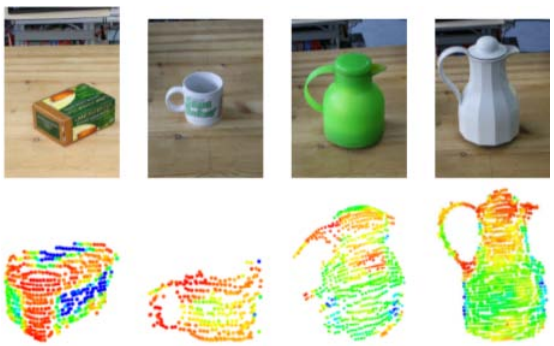


Figura 4.4: Descriptores RSD, evaluados en diferentes objetos [40]

4.4. Firma de histogramas para la descripción local de superficies

La firma de histogramas para la descripción local de superficies propuesto en [37] es un descriptor 3D basado en histogramas, los cuales proveen de robustez ante el ruido e información geométrica.

El descriptor se crea obteniendo la firma de la agrupación de un conjunto de histogramas locales que son calculados para volúmenes 3D, los cuales son definidos por una malla sobrepuesta en la superficie evaluada. La vecindad de cada punto p_i es definida

por los puntos que caen dentro de su división en la malla, para cada p_i el histograma local acumula en sus casilleros valores obtenidos a partir de la función $\cos\theta_i = n_u \cdot n_{v_i}$ del ángulo formado entre la normal de cada punto perteneciente a la división de la malla, n_{v_i}, y la normal en el punto característico n_u.

Para formar la estructura de la signatura se utiliza una rejilla esférica isotrópica que contiene particiones a lo largo de los ejes radial, azimuth y altitud ilustrados en la figura 4.5. Dado que cada volumen de la rejilla codifica una entidad muy descriptiva representada por el histograma local se puede utilizar una partición gruesa de la cuadrícula espacial y una pequeña cardinalidad del descriptor. Los experimentos expuestos en [37] muestran que 32 contenedores resultantes de ocho divisiones en azimuth, dos en altitud y dos en radial son suficientes para obtener buenos resultados.

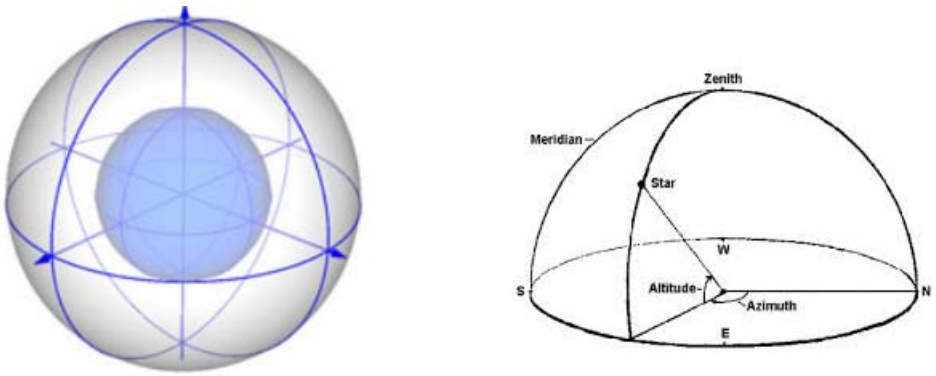


Figura 4.5: Malla de estructura para descriptor SHOT [37]

4.5. Proceso de reconocimiento de objetos con nubes de puntos

En esta sección se describen dos procedimientos para realizar el reconocimiento de objetos a partir de información 3D presentado en [31], los cuales se ilustran en la figura 4.6. El primer proceso es usando descriptores locales y el segundo es utilizando descriptores globales.

Figura 4.6: Reconocimiento de objetos a partir de información 3D [31]

A continuación se describen los pasos planteados para la utilización de descriptores locales:

Extracción de puntos de interés: Las principales características que deben de tener los puntos de interés seleccionados son: ser repetible, es decir, que tenga la capacidad de extraer el mismo punto en una variedad de vistas del objeto. Estos puntos deben de ser distintivos, caracterizar una superficie y ser fácilmente extraídos, empatados y clasificados.

Descripción: Una vez que los puntos de interés son extraídos, cada uno de ellos es asociado con un descriptor local. Este descriptor es generado utilizando la información de los puntos vecinos que están dentro de un determinado radio de distancia del punto de interés.

Empatado: Dados los descriptores de una escena y de cada uno de los modelos con los que serán contrastados se buscan correspondencias punto a punto. Éstas son detectadas en cada descriptor de la escena y su vecino más cercano respecto a los modelos, se utilizan técnicas que han sido desarrolladas para acelerar la búsqueda del vecino más cercano (NN) en FLANN (Fast Library for Approximate Nearest Neighbors)[46].

Finalmente, se consideran únicamente las correspondencias más cercanas que pasan por un umbral basado en la distancia euclidiana.

Agrupación de correspondencias: Una vez realizado el empatado punto a punto, basado en la cercanía entre descriptores, se realiza la agrupación de correspondencias con el propósito de eliminar aquellas correspondencias que no son consistentes geográficamente entre ellas.

Asumiendo que la transformación entre el modelo y su instancia en la escena es rígida, el conjunto de correspondencias a fines a cada modelo es agrupado en subconjuntos basados en una rotación y traslación específica del modelo en la escena. Los conjuntos con menor número de concurrencias respecto a un umbral son eliminados.

Un ejemplo de este tipo de verificación geométrica es presentado en [47] por Hui Chen. Dado un par de puntos del empatado este, es consistente si cumple con la siguiente restricción:

$$|d_{s1s2} - d_{m1m2}| < \epsilon \tag{4.3}$$

donde d_{s1s2} y d_{m1m2} son las distancias euclidianas del punto evaluado al centroide de sus respectivas nubes.

Estimación de orientación: Finalmente se utilizan esquemas eficientes de empatado para aproximar las vecindades cercanas. El estado anterior no garantiza que las correspondencias para cada grupo sean consistentes con una pose en una única configuración con seis grados de libertad. Por esta razón, en este paso se aplican algoritmos de verificación basados en RANSAC.

Los siguientes pasos corresponden a descriptores globales:

Segmentación: Este proceso implica separar la superficie del objeto de interés de una escena dada.

Descripción: Una vez que se tiene un conjunto de puntos que representan al objeto, se describe su geometría con algún descriptor global el cual normalmente es representado por un histograma.

Empatado: Los histogramas resultantes son comparados con los que fueron previamente extraídos en una etapa de entrenamiento buscando el mejor vecino más cercano. El empatado de histogramas se realiza usando la biblioteca FLANN mediante una búsqueda de fuerza bruta.

Alineación, orientación y refinamiento: Una vez empatados los descriptores se realizan procedimientos para eliminar falsas correspondencias basados normalmente en RANSAC, lo cual también permite calcular una transformación para alinear el objeto evaluado con el objeto identificado de la base de datos. Posteriormente esta primera transformación es refinada con ICP (Interative Closest Point)[1].

Verificación de hipótesis: se realiza mediante comparaciones de propiedades geométricas que se calculan de manera eficiente una vez que el modelo de la escena se ha alineado con el modelo de la base de datos identificado.

Finalmente cabe mencionar el trabajo de Martínez [48] presentado en el simposio de RoboCup 2014, donde diferentes descriptores 2D y 3D son evaluados en ambientes reales.

A continuación se describen brevemente los descriptores que se utilizaron en [48] por Martínez y que no han sido descritos anteriormente en el texto.

L&R SIFT, L&R SURF [49] son adaptaciones de SIFT y SURF respectivamente que pretenden ser más robustos ante distorsiones, diferencias de intensidad y transformaciones mediante un proceso de verificación que incluye supresión de no máximos, correccional lineal, análisis probabilísticos, verificación de distorsiones afines y pruebas basada en RANSAC.

OUR-CVFH [43] es un descriptor semi-global que está basado en los descriptores SGURFs [43] y CVFH [50]. OUR-CVFH utiliza la orientacion obtenida por SGURFs para extraer características geométricas del objeto. Dada una superficie S, se obtienen los tres primeros componentes y el de puntos de vista de CVFH. En este método el componte del punto de vista es codificado con 64 casilleros. Para cada agrupación c_i es creado un marco de referencia y posteriormente se aplica una transformación a la superficie S tal que los puntos pueden ser divididos en octavas definidas por los ejes $(x-, y-, z-)...(x+, y-, z-)...(x+, y+, z+)$. A cada octava se le asocia un histograma que es usado para representar la superficie.

[1] RANSAC e ICP son descritos más adelante

El descriptor **histograma de color** [48] se basa en el espacio de color HSV, donde la matriz tiene un tamaño de 30 en la dimensión H y de 32 en la dimensión S. El empatado es realizado utilizando la distancia de Hellinger.

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_{1} \sqrt{H_1(I) H_2(I)}} \tag{4.4}$$

Finalmente estos histogramas son formados únicamente con el objeto segmentado, debido a que el entorno afecta directamente en el descriptor resultante.

SIFT-VFH [48] en un método híbrido que combina la información de profundidad de una imagen con la de color. El algoritmo SIFT es aplicado antes que VFH debido a su alta precisión.

En la tabla 4.3 se exponen los resultados obtenidos en [48] donde evalúa el desempeño de los diferentes descriptores con seis objetos en una mesa bajo los siguientes escenarios:

Escenario 1: fondo blanco, iluminación normal y sin oclusión.

Escenario 2: fondo blanco, iluminación baja y sin oclusión.

Escenario 3: fondo café, iluminación normal y sin oclusión.

Escenario 4: fondo variante, iluminación normal y sin oclusión.

escenario	s1	s2	s3	s4	promedio
L&R SIFT	0.36 / 0.97	0.1 / 0.99	0.15 / 0.99	0.10/0.94	0.18 / 0.97
L&R SURF	0.18 / 1	0.4 / 1	0.05 / 1	0.03/1	0.07 / 1
L&R SIFT seg	0.36 / 0.97	0.08 / 0.96	0.14 / 0.98	0.09/0.99	0.17 / 0.97
L&R SURF seg	0.13 / 1	0.02 / 1	0.02 / 1	0.01/0.12	0.04 / 0.78
SURF	0.36 / 0.93	0.7 / 0.90	0.09 / 0.85	0.09/0.96	0.15 / 0.91
SURF seg	0.50 / 0.52	0.12 / 0.45	0.15 / 0.47	0.15/0.69	0.23 / 0.53
VFH	0.47 / 0.60	0.36 / 0.58	0.45 / 0.63	0.44 / 0.63	0.43 / 0.61
OUR-CVFH	0.50 / 0.69	0.43 / 0.68	0.50 / 0.69	0.53 / 0.69	0.49 / 0.69
Hist	0.25 / 0.65	0.10 / 0.42	0.04 / 0.39	0.01 / 1	0.10 / 0.61
SIFT-VFH	0.60 / 0.88	0.41 / 0.80	0.48 / 0.80	0.36 / 0.77	0.47 / 0.81

Tabla 4.3: Comparación de descriptores para el reconocimiento de objetos. El primer valor se refiere a la probabilidad de reconocer un objeto dado que fue detectado y el segundo se refiere a la probabilidad de deteccion [48].

4.6. Construcción de modelos 3D

La adquisición de modelos 3D o registro se ha estudiado con diversos motivos, tales como: toma de objetos, estimación de pose, adquisición de nuevos objetos [51] [52] [53]. El objetivo es alinear un conjunto de vistas del objeto en un único modelo tridimensional.

Dada una vista base de un objeto se busca agregar a ésta nueva información a partir de otra vista del mismo objeto encontrando la transformación que mejor alinee este par de vistas. Los algoritmos normalmente utilizados para realizar el registro de un objeto son RANSAC (alinear) e ICP (afinar).

La imagen 4.7 ilustra el procedimiento propuesto por PCL para realizar el registro de pares de nubes de puntos.

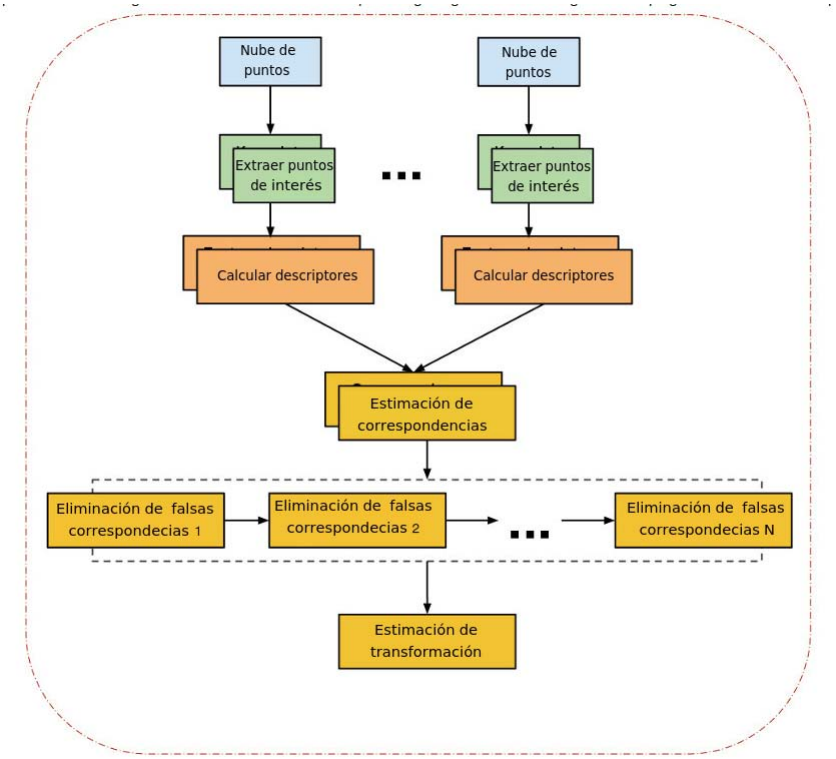


Figura 4.7: Registro de objetos [54]

Para cada nube de puntos se identifican los puntos de interés para los cuales se generan sus descriptores. Estos descriptores junto con su correspondiente información espacial (sus coordenadas x,y,z) son empatados para encontrar un conjunto de correspondencias basado en similitudes entre descriptores y posiciones. No todas las correspondencias son válidas, entre otras causas debido al ruido de los datos, por lo cual se debe realizar un procedimiento de eliminación de falsas correspondencias basado normalmente en RANSAC. Finalmente con el conjunto de correspondencias válidas se estima una transformación que se espera alinee estas nubes de puntos.

Una vez ejecutada la transformación se afina la alineación obtenida mediante métodos basados en ICP.

4.6.1. RANSAC

RANSAC fue introducido por Fisher y Boles en 1981 en [55], es una algoritmo iterativo para estimar los parámetros de un modelo matemático en un conjunto de datos el cual contiene datos que no pertenecen al modelo.

RANSAC comienza seleccionando de manera aleatoria un subconjunto de k correspondencias, el cual es usado para calcular una estimación inicial de puntos que pertenecen al modelo (inliers). Los residuos del conjunto completo de correspondencias es calculado como:

$$r_i = x_i(si; p) - x_i \tag{4.5}$$

donde x_i' son las localidades estimadas y x_i son aquellas capturadas.

Posteriormente RANSAC cuenta los inliers que se encuentran dentro del umbral de error de las locaciones precedidas $||ri|| < -e$. Este proceso es repetido S veces y el modelo con mayor número de inliers pasa a la siguiente iteración.

Para asegurarse de que se ha seleccionado el conjunto de inliers verdadero se debe seleccionar el número de S que resulta suficiente para encontrar el modelo. Si se define p como la probabilidad de éxito después de S pasadas. La probabilidad en una pasada de que los k datos aleatorios tomados inicialmente sean inliers es p_k. Además la probabilidad de que todos los S sean inliers tales que $1 - P - (1 - pK)s)$, por lo que el número requerido de S se calcula como: $S = (log(1 - p)log(1 - p^k))$.

4.6.2. Búsqueda iterativa de puntos cercanos

ICP es una de técnica utilizada para alinear geométricamente modelos tridimensionales. ICP fue propuesto por Chen y Medioni [56] y retomado por Besl y McKay en [51].

El método inicia con dos modelos y una relación estimada entre ellos, dicha relación está dada por una transformación la cual es afinada de manera iterativa generando pares de correspondencias y minimizando el error métrico. Como se describe en [57] ICP puede ser descompuesto en seis estados.

Selección: Se seleccionan algunos conjuntos de puntos en ambos modelos, esta selección usualmente se realiza de manera aleatoria.

Empatado: Se buscan correspondencias entre los puntos de ambos modelos, empatando cada punto de un conjunto con el punto más cercano en el otro modelo.

Medición: Se establece un método de medición para los pares de correspondencia empatados. Por ejemplo basándose en la distancia de los puntos o verificando la compatibilidad de las normales.

Rechazar: Se descartan aquellas correspondencias que se determine no pertenezcan al modelo (outliers).

Asignación del error métrico: Este error usualmente es la distancia entre puntos.

Minimizar el error métrico: Se minimiza el error de manera iterativa utilizando algoritmos de búsqueda de mínimos cuadrados.

Existe una gran variedad de modificaciones a este algoritmo con las que se obtienen ventajas particulares como velocidad y eficiencia, algunas combinaciones son mencionadas en [57].

CAPÍTULO 5

DESCRIPCIÓN DEL SISTEMA

En el diagrama 5.1 se muestra la arquitectura propuesta para realizar el entrenamiento y reconocimiento de objetos. Como se puede observar el sistema consta de tres bloques: captura de datos, entrenamiento y empatado, además de una base de datos.

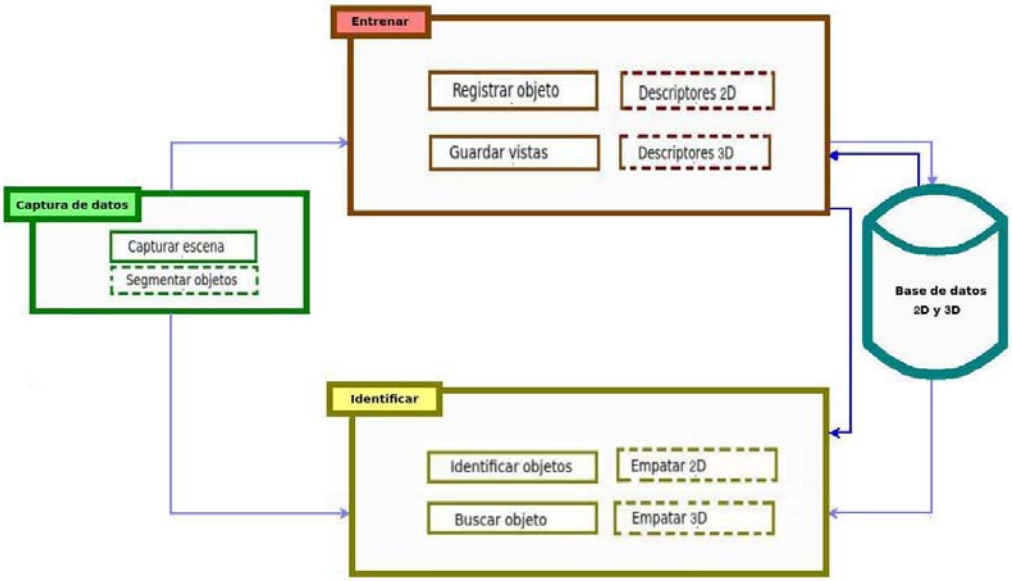


Figura 5.1: Descripción del sistema propuesto

La **base de datos** se encuentra almacenada en una carpeta llamada "objects" la cual está compuesta por un conjunto de objetos. Para cada objeto existe una sub carpeta en la cual se encuentra su respectiva información tanto 2D (imágenes RGB)

como 3D (nubes de puntos, modelo , descriptores).

Figura 5.2: Estructura de base de datos

El **bloque de captura de datos** cuenta con una serie de programas que tienen como finalidad obtener una escena del sensor Kinect y segmentarla para recuperar los objetos que se pretenden reconocer y/o registrar.

La segmentación se realiza con la información de la nube de puntos de la escena suponiendo que el robot está frente a una superficie plana donde se encuentran los objetos de interés. El proceso consiste en separar en cúmulos los puntos que se encuentran sobre el plano más grande detectado, finalmente cada cúmulo representa un objeto en la escena. Este proceso se describe detalladamente más adelante en la sección 5.2 de implementación.

El **bloque de entrenamiento** genera, a través del bloque de captura de datos, un conjunto de vistas de un objeto las cuales son almacenadas en la base de datos. Cada vista consta de una imagen RGB y su respectivas nubes de puntos en posiciones diferentes.

Las imágenes sirven para generar los descriptores SIFT y SURF, mientras que con las nubes de puntos se produce un modelo único del objeto (registro), con el cual se generan los descriptores SHOT, FPHF y RSD.

Cabe mencionar que los descriptores 2D son generados en tiempo de ejecución al iniciar el sistema, lo cual permite actualizar la base de datos con imágenes que mejoren los resultados sin tener que volver a entrenar el objeto.

En cuanto a los descriptores 3D estos se generan con una única nube de puntos, la cual puede ser: el modelo resultante del procedimiento de registro 3D del objeto ó una vista cuya nube de puntos el usuario considere tenga suficiente información para realizar la descripción del objeto.

No fue posible automatizar el entrenamiento de un objeto, la razón principal es que los parámetros y vistas para lograr una mejor detección de objetos, sobre todo con la información 3D, son muy variantes. Sin embargo, el procedimiento diseñado para la segmentación y almacenamiento de objetos en el bloque de captura de datos permiten actualizar la base de datos de una manera muy práctica.

Finalmente el **bloque de identificar** es el encargado de realizar las búsquedas de correspondencias 3D y 2D entre los objetos de la escena y los pertenecientes a la base. Este bloque tiene dos funciones principales: buscar un objeto en la escena e identificar todos los objetos en la escenas. La aplicación permite al usuario decidir usar o no descriptores 3D ó 2D y seleccionar la combinaciones que se desee(p.j. SIFT-RSD).

5.1. Herramientas

La implementación de este sistema está basada en bibliotecas de código abierto que contienen algoritmos e interfaces, bajo el lenguaje de programación C++, para la obtención y procesamiento de imágenes RGB-D capturadas por el dispositivo Kinect de Xbox 360 [58].

El Kinect es un sensor desarrollado por PrimeSense [59], el cual ofrece la posibilidad de observar una escena sincronizando la información de profundidad y color en cada observación tal como lo haría un humano. Este dispositivo está conformado por una cámara RGB la cual captura información de color por medio de un sensor CMOS y un sensor de profundidad basado en infrarrojo, que combinados hacen posible obtener

frames donde cada pixel almacena información sobre distancia respecto al sensor y color.

El rango visual del sensor de Kinect está entre 0.35 y 3.5 metros de distancia, con un ángulo de vista de 57.5° horizontalmente y 45° verticalmente. Las imágenes resultantes son de 640 x 480 pixeles RGB-D y captura 30 frames por segundos. Debido a su fácil uso, conectividad a través de USB y precio se ha vuelto uno de los dispositivos más usados.

OpenNI [60] es una biblioteca multiplataforma de código abierto utilizada para realizar la comunicación con sensores de audio y video por medio de la cual es posible extraer la información RGB-D de dispositivos como el Kinect.

OpenCV (Open Source Computer Vision) [61] es una biblioteca multiplataforma, desarrollada desde 1999 por Intel, integrada por cientos de algoritmos de visión computacional que van desde leer y guardar imágenes, aplicación de filtros, segmentación, extracción de características, hasta generación de descriptores y empatado entre imágenes.

Finalmente PCL (Point Cloud Library) [35] es una biblioteca que contiene una serie de algoritmos del estado del arte implementados con el propósito segmentar, filtrar, obtener descriptores y empatar nube de puntos.

5.2. Implementación

En esta sección se describen de forma detallada los componentes y procedimientos implantados en el sistema de entrenamiento y reconocimiento propuesto a través de las clases:

- CamaraKinect
- ImageCamaraKinect
- CapturaEscena
- EntrenarObjeto
- EncontrarObjeto
- Descriptor3D
- Descriptor2D

La figura 5.3 muestra el diagrama de clases del sistema propuesto:

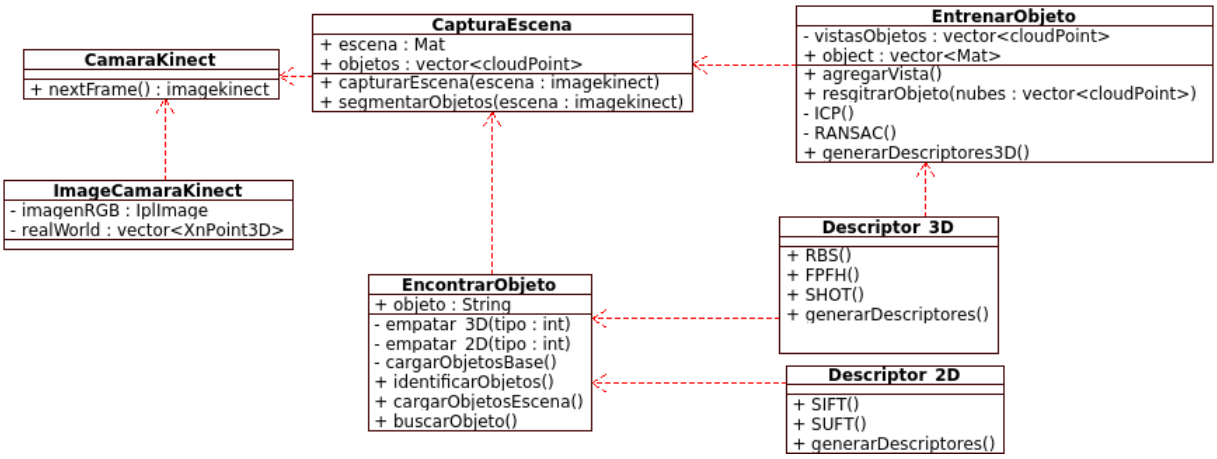


Figura 5.3: Diagrama de clases

Las clases **CamaraKinect e ImageCamaraKinect** fueron desarrolladas previamente por los Maestros en Ingeniería Abel Pacheco Ortega y Luis Angel Contreras Toledo integrantes del laboratorio de Bioróbotica bajo la supervisión del Dr. Jesús Savage Carmona. Estas clases son una interfaz para obtener la información tanto 3D como 2D del dispositivo Kinect.

La clase **CapturarEscena** se encarga de obtener una escena, segmentar el área de interés y obtener un colección de objeto(s) que serán utilizados ya sea para identificarlos o almacenarlos.

La segmentación de la escena se realiza a partir de suponer que los objetos se encuentran en una superficie plana la cual es localizada con la clase SACSegmentation de PCL, esta clase está basada en RANSAC e identifica los puntos que pertenecen a un modelo, en este caso planos en la imagen. Una vez identificado el plano más grande se toma éste como referencia para construir un filtro de caja que permita segmentar un área de interés sobre la superficie.

Los puntos que quedan en el área de interés son separados en agrupaciones utilizando la clase EuclideanClusterExtraction de PCL. Cada agrupación representa la nube de puntos de cada objeto en la escena.

Finalmente para cada agrupación todos sus puntos son mapeados, utilizando la función ConvertRealWorldToProjective de OpenNI, en la imagen RGB de la escena para obtener la información de color del objeto. Las imágenes 5.4 y 5.5 ilustran el procedimiento de captura de escena.

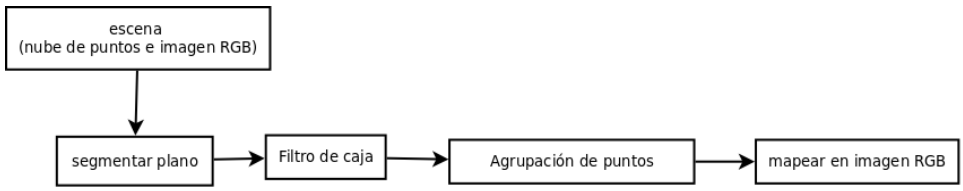


Figura 5.4: Captura de escena y segmentación de objetos

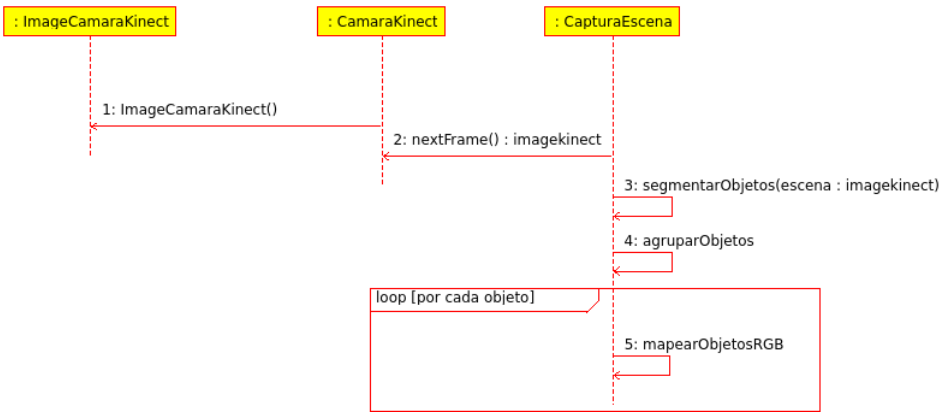


Figura 5.5: Diagrama de secuencia: capturar escena

La clase **EntrenarObjeto** contiene métodos para registrar vistas de un objeto y realizar su registro.

El entrenamiento se realiza capturado varias vistas del objeto, sus imágenes y nubes de puntos son almacenados por medio del método agregarVista de la clase entrenar objeto como se observa en la imagen 5.6.

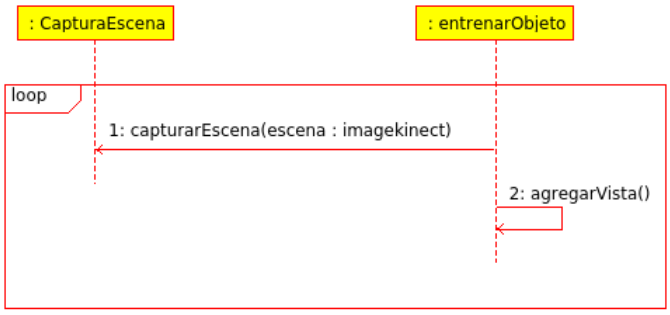


Figura 5.6: Diagrama de secuencia: entrenamiento de objetos

Una vez capturado el conjunto de vistas el usuario selecciona las nubes de puntos con las que se realizará *el registro 3D del objeto*, el cual es implementado en el método RegistrarObjeto de la clase EntrenarObjeto. Por último con el modelo resultante se generan y almacenan sus descriptores 3D, este procedimiento se observa en la figura 5.7.

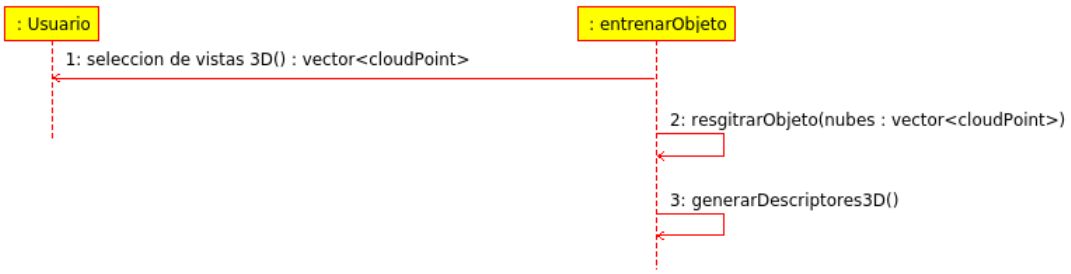


Figura 5.7: Diagrama de secuencia: registro de objetos

El proceso de registro se ilustra en la imagen 5.8. Tomando como entrada el conjunto de nubes de puntos seleccionado por el usuario se considera la primera nube como base. La base se va incrementando alineándola con la nube de puntos siguiente hasta generar el modelo final.

Para cada par de nubes de puntos, nube_base y nube_src, la alineación se hace en dos etapas. La primera consiste en obtener los descriptores FPFH de ambas nubes de puntos, con los cuales se realiza una búsqueda de correspondencias que son evaluadas rechazando aquellas que no cumplan con un umbral de distancia utilizando la

clase CorrespondenceRejectorDistance de PCL. Una vez que se tiene el conjunto de buenas correspondencias se calcula una transformación, que aproxima la nube_src a la nube_base, con la clase TransformationEstimationSVD de PCL. Finalmente se afina la alineación resultante aplicando iterativamente ICP usando la implementación de PCL en su clase IterativeClosestPointWithNormals.

Las variables *d_icp_max*, *d_icp_min*, *intervalos_icp*, controlan el número de iteraciones del algoritmos ICP, buscando que la distancia entre los puntos de ambas nubes de puntos disminuya en cada iteración hasta el umbral *d_icp_min*. Por otro lado *d_icp_max* controla la distancia de búsqueda en cada iteración, y es disminuida por *intervalos_icp*, esperando que cada vez la distancia entre puntos sea menor.

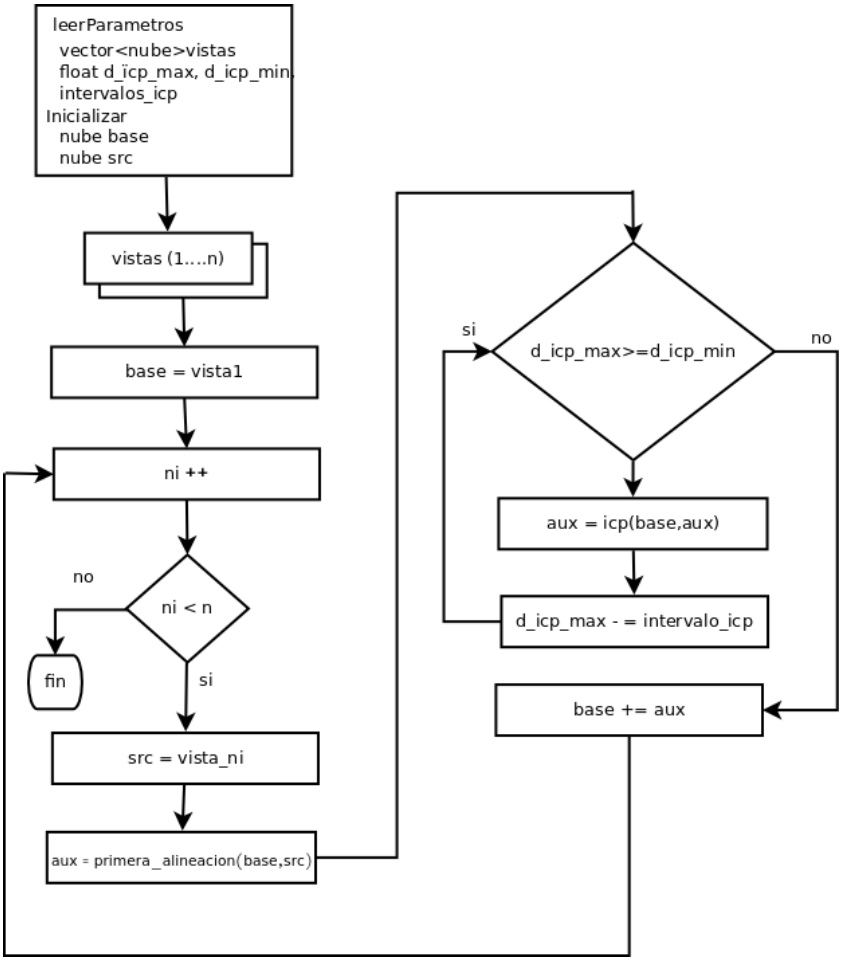


Figura 5.8: Proceso de registro 3D de objeto

Por medio de la clase **EncontrarObjeto** se elige buscar un objeto en la escena o identificar todos los objetos de la escena, para realizar las tareas mencionada esta clase carga los objetos almacenados, sus descriptores 3D y genera sus descriptores 2D. Por otro lado obtiene los objeto de la escena y genera su descriptores tanto 3D como 2D.

Una vez que se tienen todos los descriptores se inicia la búsqueda de correspondencias para realizar la identificador. La implementación de este proceso se ilustra en la figura 5.9.

La clase EncontrarObjeto utiliza las bibliotecas de OpenCV y PCL para realizar el empate de descriptores. Para empatar descriptores 2D se utilizaron las clases FlannBasedMatcher de OpenCV y para la información 3D la clase CorrespondenceEstimation de PCL.

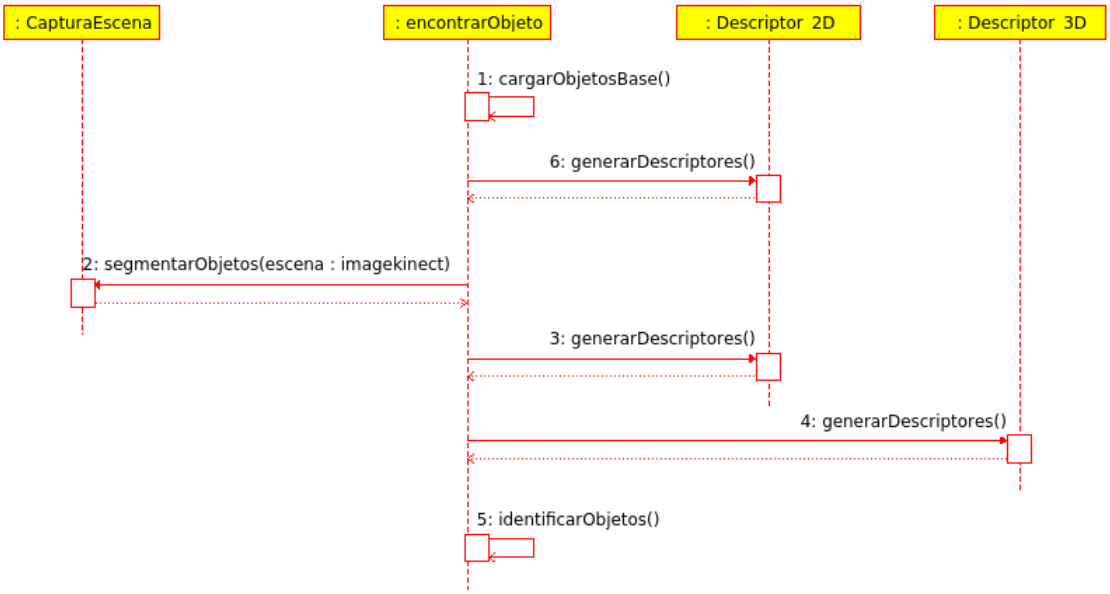


Figura 5.9: Diagrama de secuencia: Identificación de objetos en una escena

El diagrama 5.10 expone el *proceso de reconocimiento de objetos en una escena*, cada objeto de la escena es empatado con todos los de la base de datos, los descriptores que se utilizarán los define el usuario.

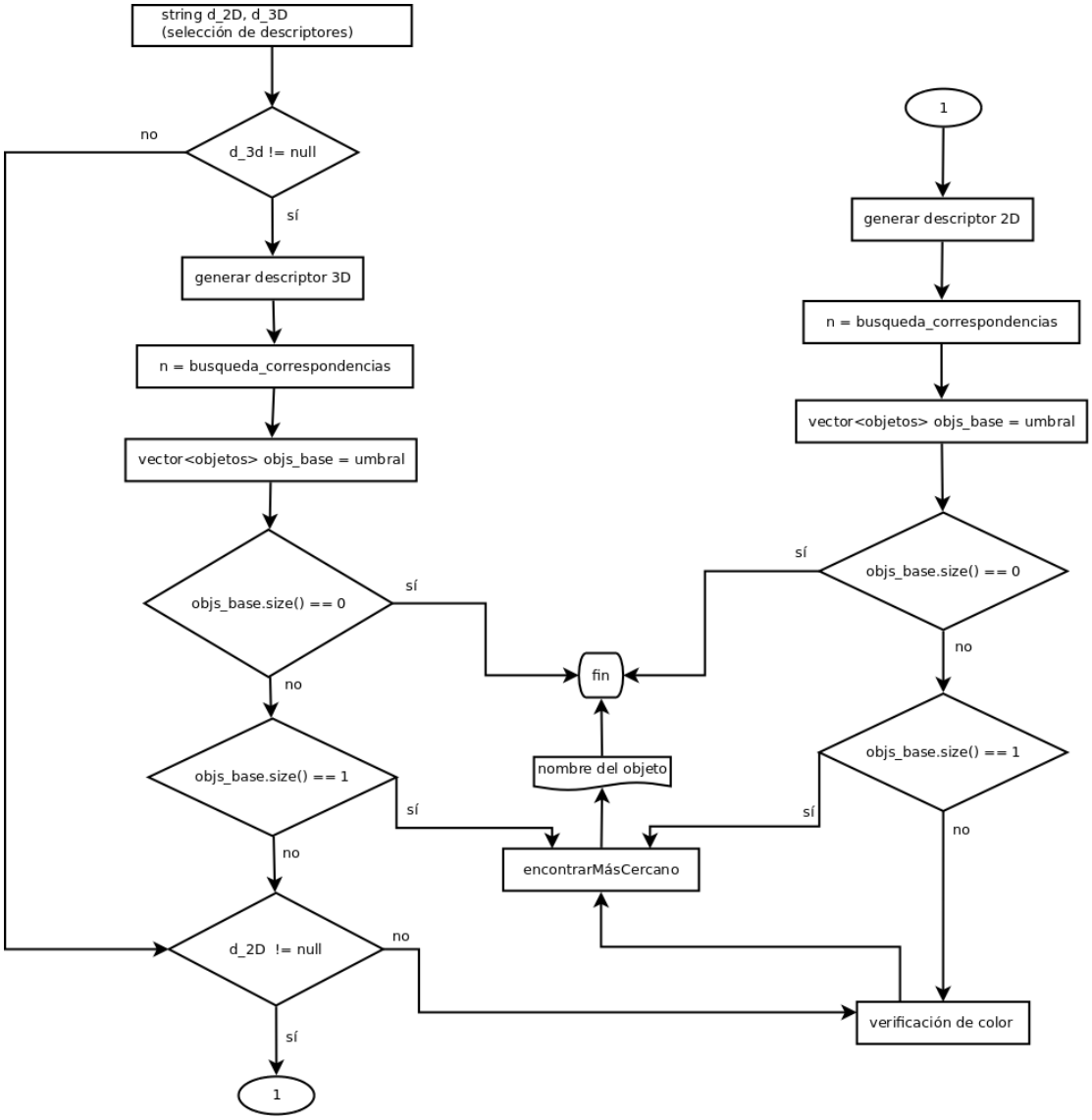


Figura 5.10: Proceso de reconocimiento de objetos

Primero se buscarán correspondencias con las nubes de puntos de la base y el objeto de la escena, cada búsqueda genera un número de correspondencias por lo que únicamente pasarán a la segunda etapa aquellos que tengan objetos de la base que generan mas del promedio de correspondencias.

En el segundo bloque se calculan las correspondencias generadas con los descriptores

2D, únicamente con los objeto que pasaron el filtro anterior. Si uno de los objetos de la base genera más correspondencias que todos los demás entonces se elige ese objeto, de lo contrario, similar al bloque anterior, sólo aquellos objetos de la base que hayan generado más del promedio de correspondencias 2D pasarán al siguiente bloque.

Con este último bloque se decide con que objeto de la base relacionar el de la escena que se está analizando por medio de similitud en el color. Para este propósito se obtienen los promedios de las imágenes de los objetos en el canal H del espacio de color HSV, aquel objeto de la base que presente una distancia menor con el objeto de la escena será el elegido.

La única diferencia que existe cuando se busca un objeto específico en la escena es que sólo se carga dicho objeto y éste es empatado con todos los objetos encontrados en la escena, al finalizar se indica que objeto de la escena fue identificado como el buscado. Si no se encuentran suficientes correspondencias en ningún objeto de la escena se obtiene como respuesta "objeto no encontrado".

Finalmente las clases **Descriptor3D y Descripoter2D** contienen las implementaciones de los descriptores SURF y SIFT que ofrece OpenCV y SHOT, FPFH, y RSD de PCL.

CAPÍTULO 6

APLICACIÓN EN EL ROBOT DE SERVICIO JUSTINA

En esta sección se describen las pruebas del sistema de detección y reconocimiento de objetos realizadas. La primera parte trata sobre detección de objetos, la segunda muestra el procedimiento de registro y almacenamiento de objetos y finalmente en la tercera se exponen las pruebas de reconocimiento.

6.1. Detección de objetos en escena

Suponiendo que los objetos se localizan sobre una superficie plana, la detección de estos se realiza capturando la escena por medio de un dispositivo Kinect. A continuación de describe el procedimiento a detalle.

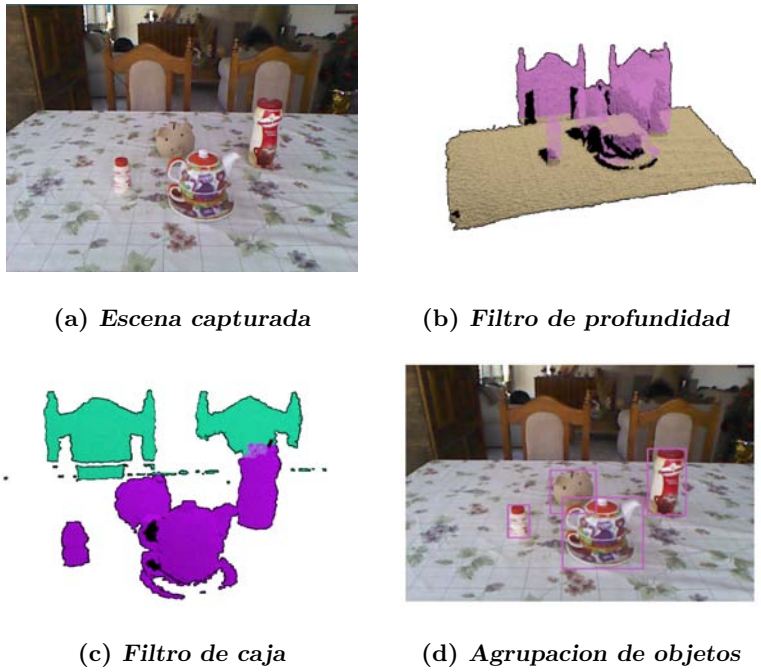
El primer paso consiste en filtrar la escena con los datos de profundidad quedándose únicamente aquellos puntos que están a una distancia determinada por el usuario. Lo anterior con el fin de reducir el área de búsqueda en la siguiente etapa.

El segundo paso consiste en encontrar el plano más grande en la escena filtrada, esperando que este corresponda al plano en el que están localizados los objetos, la búsqueda del plano se realiza utilizando RANSAC. En este paso se remueven todos los puntos que pertenecen al plano detectado y se guardan las coordenadas de su centroide.

Con las coordenadas del plano se aplica un filtro de caja a los puntos restantes en la escena, para restringir el área en la que se encuentran los objetos debido a que puede ser que el primer filtro no restringa únicamente el área deseada.

Finalmente los puntos restantes pertenecen a los objetos de interés en la escena, estos puntos son separados mediante un proceso de agrupación basado en distancia euclidana, cada cúmulo de puntos agrupado es asociado a un objeto en la escena. El objeto o vector de objetos que se obtienen como salida de este proceso se utilizan para realizar el entrenamiento o identificación de objetos según sea la tarea que vaya a ejecutar el robot.

La imagen 6.1 ilustra el proceso de extracción de objetos anteriormente explicado, aplicado a una escena en particular.



(a) ***Escena capturada*** **(b)** ***Filtro de profundidad***

(c) ***Filtro de caja*** **(d)** ***Agrupacion de objetos***

Figura 6.1: Proceso de detección de objetos a) Escena capturada; b) Puntos después de aplicar el filtro de profundidad, en tono café se muestran los puntos que pertenecen al plano; c) En tono morado se muestran los puntos que mantiene el filtro de caja; d) Objetos detectados

Para evaluar el desempeño referente a la tasa de detección se analizaron las escenas capturadas para las pruebas de reconocimiento.

En total fueron capturadas 101 escenas en las que se localizan diversos objetos (conocidos y desconocidos). El número total de detecciones esperadas en estas escenas fue de 492, de las cuales, 467 fueron exitosas, y las restantes 25 presentaron dos tipos de problemas, provocados principalmente por el ruido del Kinect:

1. Objetos no detectados.
2. Agrupaciones falsas, es decir, un objeto es agrupado en varios cúmulos ó se agrupan los puntos pertenecientes a más de un objeto en un sólo cúmulo.

La gráfica 6.2 ilustra los resultados obtenidos:

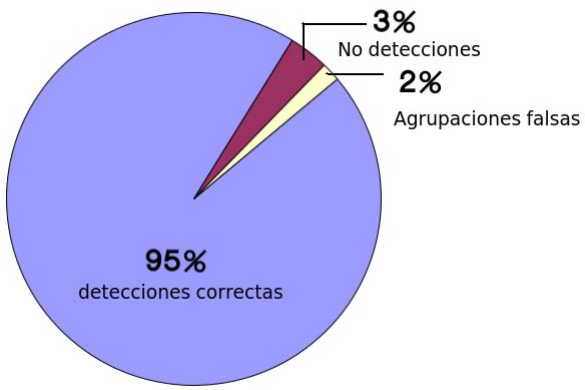
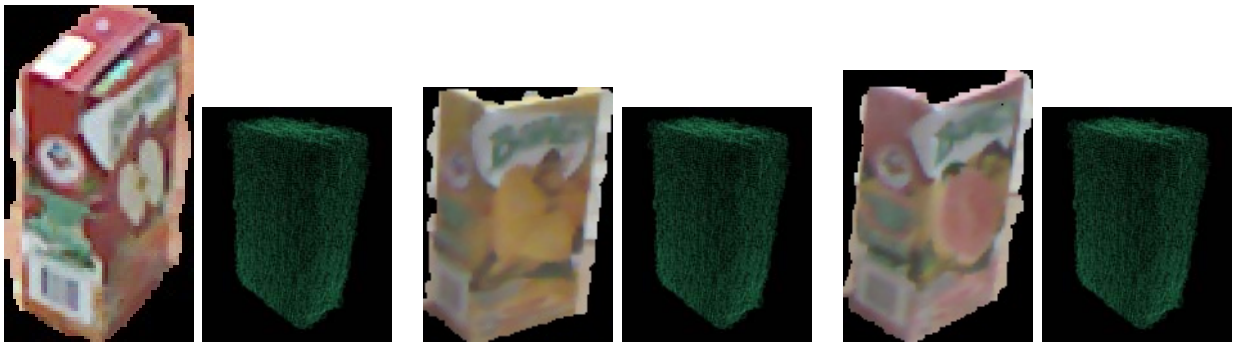


Figura 6.2: Desempeño del procedimiento de detección de objetos

6.2. Entrenamiento de objetos

El entrenamiento es el proceso de generar y almacenar modelos de los objetos que se desea reconozca el robot posteriormente.

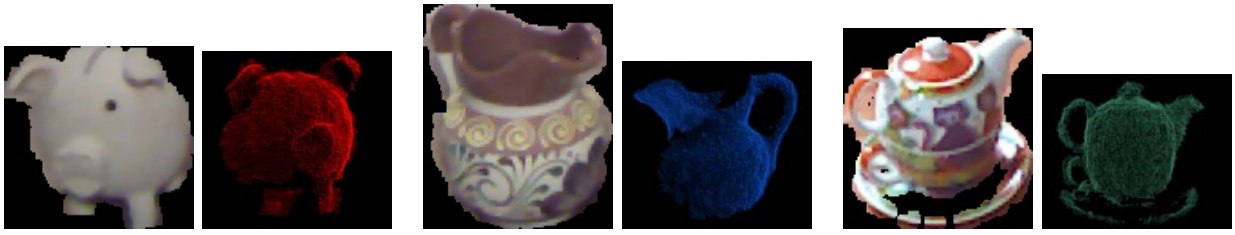
La tabla 6.1 muestra los objetos entrenados para realizar las pruebas de reconocimiento del presente trabajo. Estos fueron elegidos con el propósito de evaluar la robustez de los algoritmos para generar descriptores suficientemente distintivos ante similitudes de formas (tazas, jugos y latas), y de información visual (jugos).



1. Jugo manzana | 2. Jugo de mango | 3. Jugo de guayaba

4. Jugo grande | 5. Taza 1 | 6. Taza 2

7. Refresco(lata) | 8. Tequila(lata) | 9. Leche



10. Puerco | 11. Jarra | 12. Tetera

Tabla 6.1: Objetos registrados

El entrenamiento de cada nuevo objeto se realiza en dos etapas. La primera etapa consiste en tomar suficientes vistas del objeto y almacenarlas según la estructura de la base de datos descrita anteriormente. En la segunda etapa se realiza el registro 3D del objeto.

Para realizar el registro 3D de los objetos, es necesario que el usuario seleccione para cada objeto las nubes de puntos que pertenecen a las vistas que más le beneficien para obtener un buen modelo para generar sus descriptores 3D.

Cabe mencionar que los parámetros de los diferentes algoritmos involucrados en el proceso de registro expuesto en la sección 5.2 varían según el objeto que se quiere registrar. La tabla 6.2 muestra los parámetros utilizados para cada objeto.

objeto/parámetro	1	2	3	4	5	6	7	8	9
leche	0.002	0.02	0.03	0.05	0.03	0.005	0.02	0.001	5e-07
tetera	0.002	0.005	0.02	0.005	0.01	0.01	0.05	0.005	0
jarra	0.002	0.005	0.02	0.005	0.01	0.01	0.05	0.005	0
jugo chico	0.002	0.03	0.05	0.05	0.02	0.01	0.05	0.005	1e-08
jugo grande	0.002	0.02	0.03	0.05	0.03	0.005	0.02	0.001	5e-07
puerco	0.002	0.01	0.02	0.01	0.02	0.01	0.05	0.005	1e-08
taza1	0.002	0.02	0.05	0.005	0.03	0.01	0.1	0.001	0.0001
taza2	0.0015	0.03	0.05	0.1	0.05	0.005	0.1	0.005	1e-06

Tabla 6.2: Parámetros utilizados para el registro de objetos: 1. Radio de búsqueda para la extracción de puntos de interés. 2. Radio de búsqueda para el cálculo de normales 3. Radio de búsqueda para la generación de descriptores PFFH 4. Distancia para rechazar falsas correspondencias (RANSAC) 5. Radio de búsqueda para encontrar puntos cercanos 6. Intervalo para disminuir la distancia máxima de correspondencias de ICP 7. Distancia máxima de correspondencias de ICP 8. Distancia mínima de correspondencias de ICP 9. Epsilon utilizado por ICP. Para más información ver la documentación de PCL [35]

Los parámetros 6, 7 y 8 de la tabla 6.2 definen el número de veces que se ejecu-

tará ICP. En cada iteración la distancia máxima de correspondencias (parámetro 7) toma un valor menor, reducido por el intervalo (parámetro 6), hasta llegar al mínimo (parámetro 8), esperando que en la ejecución sucesiva la nube de entrada se acerque más al modelo.

Se consideró que para la generación de descriptores era suficiente, para el caso de objetos simétricos, tener en el registro la información de la mitad del objeto.

Para los objetos jugo de manzana, jugo de mango y jugo de guayaba, los cuales se muestran en la tabla 6.2 como jugo chico, se utiliza el mismo modelo; mientras que para los objetos de lata se utiliza como modelo una única vista debido a que su simetría no permite aumentar información al modelo.

A continuación se ilustra, como ejemplo del proceso de registro 3D, los objetos: tetera (figura 6.3) y leche (figura 6.4). La primera fila de las imágenes muestra la información de color; la segunda las nubes de puntos respectivas a cada toma del objeto, las nubes que se encuentran dentro de rectángulos son las seleccionadas para realizar el registro. Finalmente en la última fila se puede observar el proceso de registro al aplicar RANSAC e ICP.

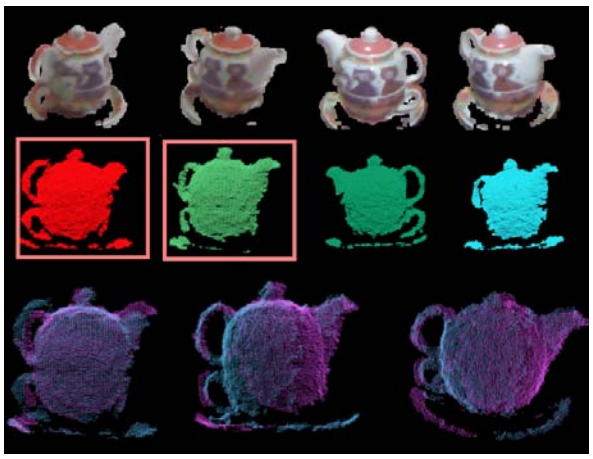


Figura 6.3: Registro de objetos: Tetera

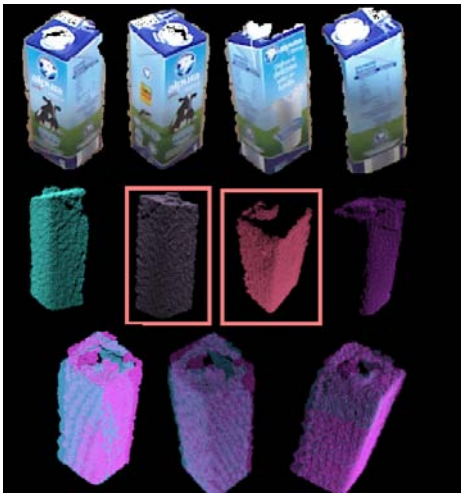


Figura 6.4: Registro de objetos: Leche

Respecto a la información de color, se almacenan las imágenes recuperadas de cada vista del objeto para generar sus descriptores 2D en la etapa de reconocimiento.

6.3. Reconocimiento de objetos

Dados los objetos almacenados en la base de datos del robot se desea que éste pueda identificarlos en diferentes escenas con el propósito de complementar tareas básicas, por ejemplo, llevar un objeto específico de un lugar a otro.

Una vez que se ha capturado la escena de interés el robot puede:

1. **Buscar un objeto específico**: cada objeto en la escena es empatado con los descriptores de un objeto previamente almacenado que el usuario elija. Si el objeto solicitado no se encuentra en la base de datos o no es identificado en la escena se obtiene como repuesta *"object no found"*. En la figura 6.5 se ilustra la búsqueda del objeto *taza* el cual es localizado y enmarcado en azul y el ejemplo de una búsqueda donde no se ha localizado el objeto solicitado.

(a) ***Busqueda del objeto taza*** (b) ***Busqueda de un objeto no encontrado***

Figura 6.5: Buscar un objecto en la escena

2. **Identificar los objetos detectados en la escena**: cada objeto en la escena se empata con todos los objetos en la base de datos. En la figura 6.6 se muestra un ejemplo donde se observa que cada objeto detectado fue etiquetado según el resultado de la búsqueda.

Figura 6.6: Identificar los objetos en la escena

Para realizar la búsqueda el usuario puede elegir que descriptores usar y sus combinaciones. La detección se toma en base al número de buenas correspondencias encontradas. Si existe un empate entre el número de correspondencias se discrimina utilizando un promedio de color.

6.3.1. Prueba 1. Comparación de desempeño entre descriptores

En esta prueba se evalúan los descriptores SIFT, SURF, FPFH, RSD, SHOT así como sus combinaciones, empatando los objetos detectados en la escenas con todos los que se encuentran en la base de datos.

Las escenas utilizadas para esta prueba se ilustran en la figura 6.7.

Figura 6.7: Prueba 1: Escenas analizadas

La figura 6.8 muestra los objetos enmarcados en las escenas según fueron detectados, como se puede observar en las escenas 3 y 4 se presento, en el objeto "taza1", el problema descrito en la sección anterior, donde un objeto es agrupado en dos cúmulos diferentes. Por otro lado se observa que se detectan de manera correcta objetos ocluidos que se encuentran en la parte trasera de las escenas.

Figura 6.8: Prueba 1: deteccion de objetos en escenas

En la tabla 6.3 se exponen los resultados obtenidos utilizando los diferentes descriptores y sus combinaciones resultantes entre descriptores 3D y 2D. Las primeras cuatro columnas muestran el número de detecciones correctas en cada escena, la siguiente columna contiene la tasa de reconocimiento por descriptor y en la última se observa el tiempo promedio consumido. Cada escena contienen 8 objetos (6 conocidos y 2 desconocidos), lo que da un total de 32 objetos.

Descriptor	Escena **1**	Escena **2**	Escena **3**	Escena **4**	Tasa de reconocimiento	Tiempo promedio [segundos]
SIFT	3	4	1	2	31.25 %	2.91
SURF	2	2	4	2	31.25 %	1.59
FPHF	1	4	3	4	37.5 %	4.44
SHOT	2	3	0	1	18.75 %	42.72
RSD	1	2	1	2	18.75 %	0.66
SIFT /FPFH	2	6	3	4	46.87 %	5.31
SIFT/SHOT	3	3	6	3	46.87 %	45.34
SIFT/RSD	1	3	4	2	31.25 %	1.84
SURF/FPFH	5	4	4	4	53.12 %	4.69
SURF/SHOT	4	3	3	3	40.82 %	45.34
SURF/RSD	3	4	4	3	43.75 %	1.22

Tabla 6.3: Prueba 1: Comparación de desempeño entre descriptores

A continuación se exponen las gráficas que resumen los resultados obtenidos, la figura 6.9 muestra la referente al desempeño de detección reportado, donde se observa que los procedimientos realizados combinando descriptores 3D y 2D en su mayoría obtuvieron resultados con tasas superiores que utilizandolos por separado.

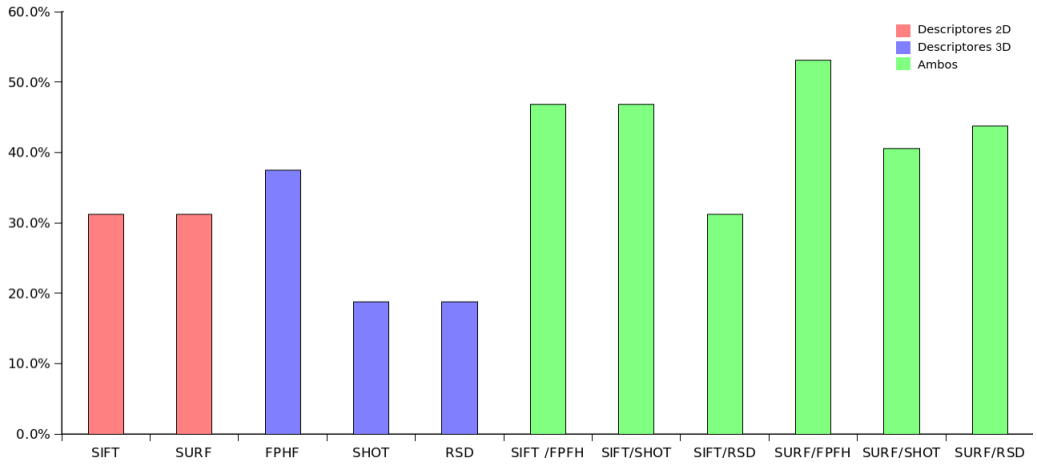


Figura 6.9: Prueba 1. Promedio de reconocimiento

La gráfica de consumo de tiempo, que se ilustra en la figura 6.10, muestra que el descriptor SHOT consume considerablemente mucho más tiempo que los demás descriptores. Por lo anterior este descriptor no es considerado en las siguientes pruebas.

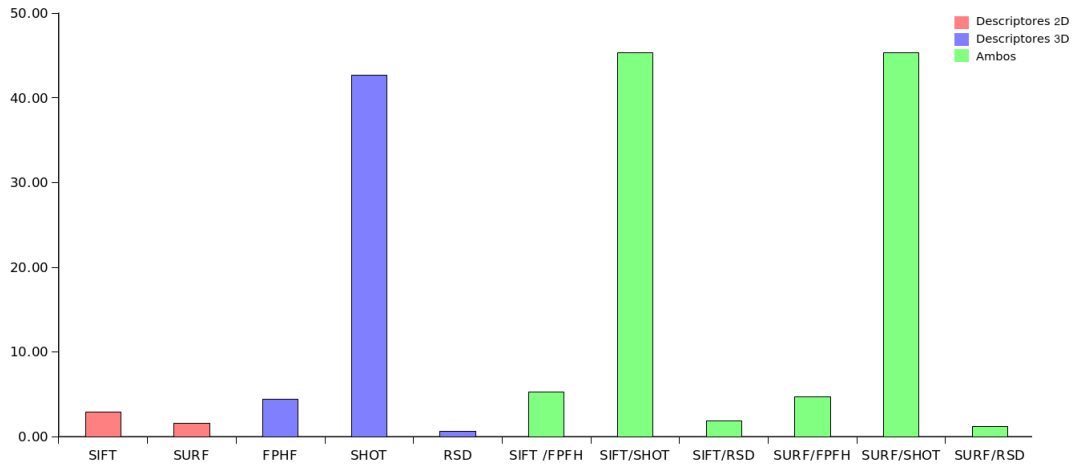


Figura 6.10: Prueba 1. Consumo de tiempo por objeto (segundos)

6.3.2. Prueba 2: Comparación de desempeño entre los descriptores que presentaron menor consumo de tiempo

Para esta prueba se capturaron 70 escenas, las cuales fueron tomadas con el dispositivo a un metro de distancia. En resumen se detectaron correctamente 335 objetos, de los cuales 228 corresponden a objetos conocidos, es decir, que pertenecen a la base de datos y 107 a desconocidos.

A continuación se muestra en la figura 6.11 la gráfica de desempeño en cuanto a reconocimiento utilizando los descriptores SIFT, SURF, FPHF, RSD y sus combinaciones. Quien logro mayor porcentaje de reconocimiento fue la combinación SIFT-RSD con el 51.9 %.

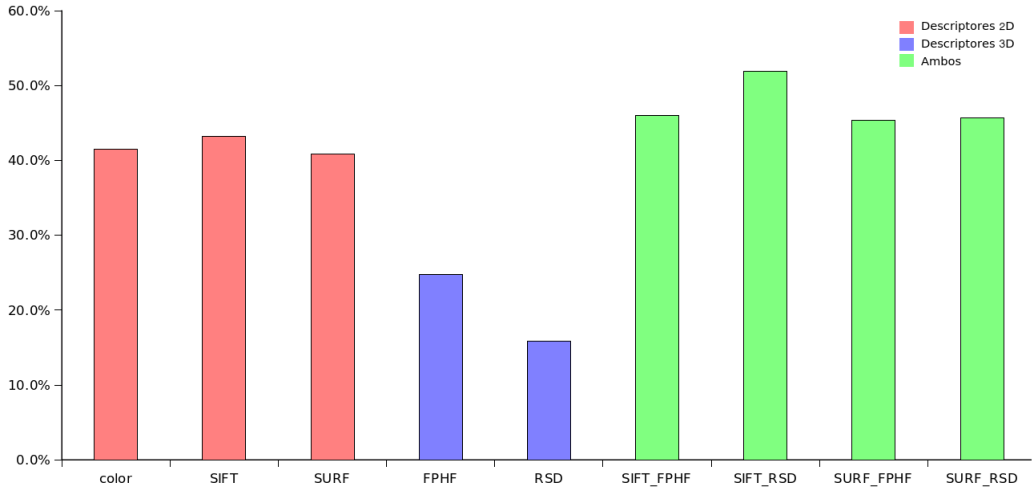


Figura 6.11: Prueba 2. Promedio de reconocimiento

La figura 6.12 muestra el tiempo promedio consumido por objeto utilizando los diferentes descriptores. En la gráfica se observa que las combinaciones con el descriptor RSD mejoran el tiempo de consumo de los descriptores 2D usándolos por separado.

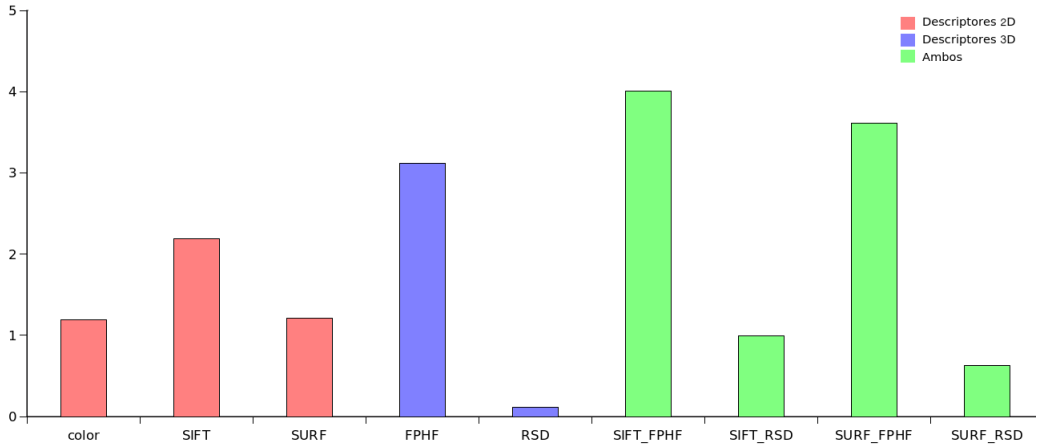


Figura 6.12: Prueba 2. Consumo de tiempo por objeto (segundos)

6.3.3. Prueba 3: Comparación de desempeño entre descriptores en escenas cercanas

En esta prueba fueron tomadas 19 escenas capturadas con el dispositivo más cerca, a medio metro de distancia. Se detectaron correctamente 96 objetos, 68 conocidos y 28 desconocidos.

La figura 6.13 muestra los resultados obtenidos utilizando los diferentes descriptores. Se puede observar que la tasa de reconocimiento aumento utilizando los descriptores 2D, sin embargo, los procedimientos que involucran ambos tipos de descriptores siguen manteniendo mejores resultados.

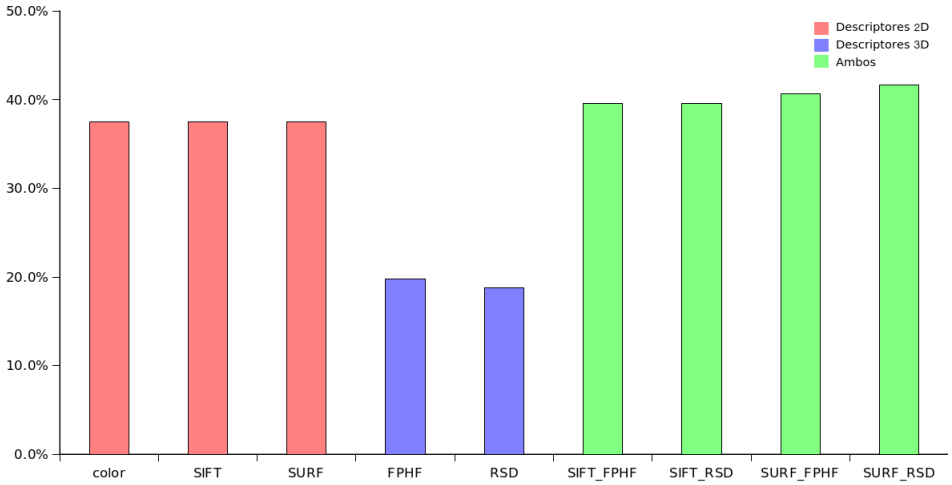


Figura 6.13: Prueba 3. Promedio de reconocimiento

La imagen 6.14 expone los promedios de consumo de tiempo por objeto de los descriptores.

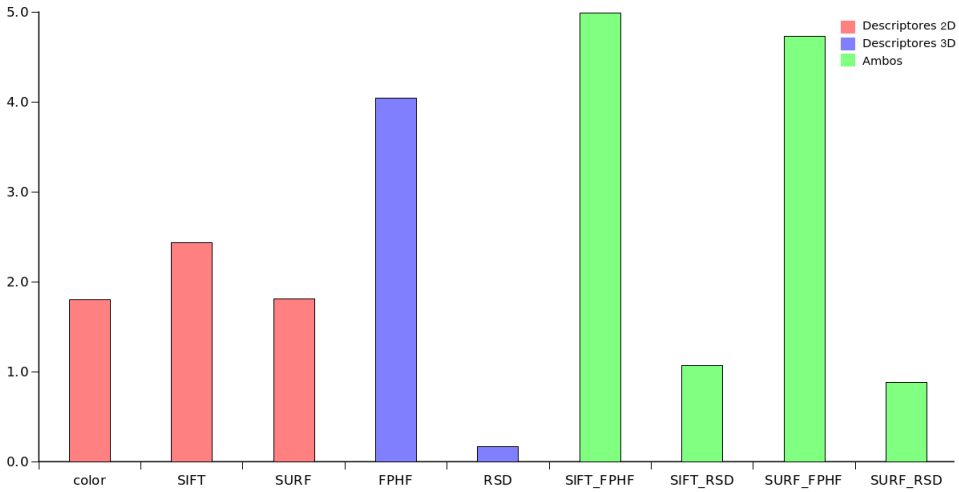


Figura 6.14: Prueba 3. Consumo de tiempo por objeto (segundos)

En esta prueba se observa que la combinación de SURF-RSD mantiene obteniendo los mejores resultados en cuanto a reconocimiento y tiempo consumido.

6.3.4. Prueba 4: Comparación de desempeño entre descriptores en escenas cercanas sin objetos desconocidos

En esta prueba fueron tomadas 12 escenas capturadas con el dispositivo cercano, a medio metro. En esta pruebas sólo se utilizaron objetos pertenecientes a la base de datos, en total se obtuvieron 36 detecciones.

La figura 6.15 muestra la rasa de reconocimiento reportada en la prueba utilizando los diferentes descriptores.

En esta prueba se encuentra la tasa de reconocimiento más alta en todas las pruebas con SIFT-FPFH, sin embargo el porcentaje no es significativamente mayor de cuando se involucran objetos desconocidos.

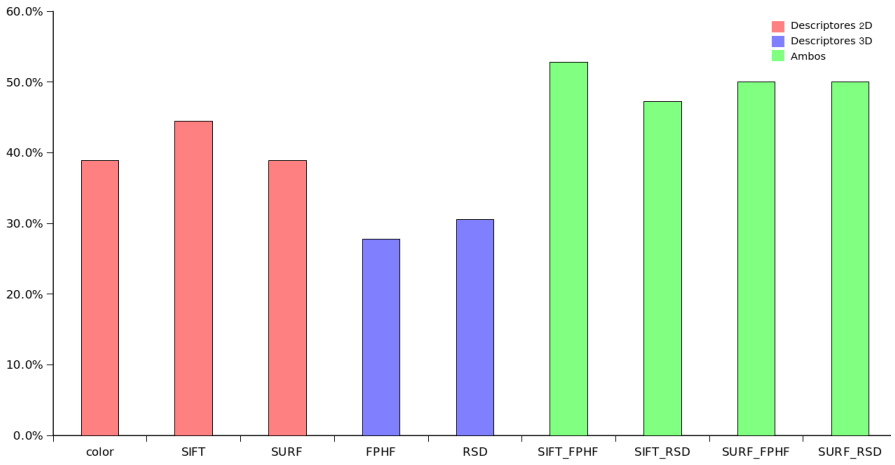


Figura 6.15: Prueba 4. Promedio de reconocimiento

La gráfica de consumo de tiempo que se muestra en la figura 6.16 expone resultados muy similares a los obtenidos en las pruebas anteriores, donde las combinaciones que utilizan RSD muestran ser más rápidas.

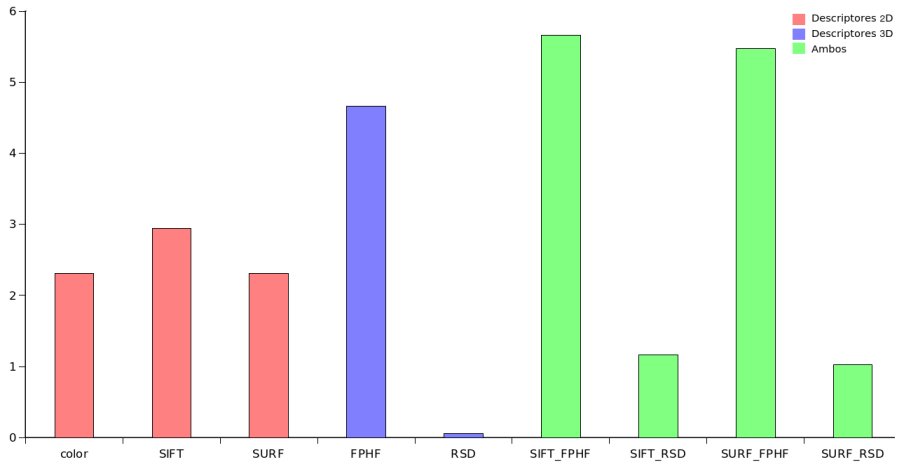


Figura 6.16: Prueba 4. Consumo de tiempo por objeto (segundos)

En las pruebas se puede observar que la combinación de técnicas 2D y 3D aumentan la tasa de reconocimiento. RSD en combinación con los descriptores 2D muestra altos resultados en el reconocimiento sin comprometer tanto el tiempo como FPFH, sin embargo FPFH muestra en la mayoría de los casos promedios mayores de reconocimiento.

CAPÍTULO 7

CONCLUSIONES

El reconocimiento y descripción de objetos es una capacidad indispensable que permite a un robot de servicio hacer tareas complejas como llevar objetos a los usuarios.

Durante el trabajo se expusieron diversos algoritmos dedicados a la adquisición de descriptores y empatado de características para el reconocimiento de objetos tanto 2D como 3D, los cuales tienen como finalidad encontrar y describir características en imágenes y en nube de puntos respectivamente. A pesar de existir una gran variedad de implementaciones de los algoritmos descritos es difícil encontrar una que logra resultados satisfactorios sin comprometer el tiempo de ejecución.

La propuesta de este trabajo fue hacer un sistema híbrido, es decir, que combinara el uso tanto de información de color como de su forma para el reconocimiento de objetos, esperando evitar problemas como el de reconocer un objeto donde únicamente se encuentre su imagen estampada en otro o confundirse entre formas similares así como reducir el campo de búsqueda.

Los resultados obtenidos muestran en general mejoras respecto a usar un único descriptor, aún probando con objetos no conocidos. Sería deseable que estas mejoras fueran mayores. Por el tiempo de ejecución y tasa de reconocimiento una buena combinación es SURF-RSD.

Otro objetivo que se buscó en el trabajo presente fue buscar soluciones que permitan al robot ser lo más autónomo posible para la adquisición de nuevos modelos en una etapa de alimentación de la base de datos. Para lo cual se encontraron diversas dificultades, por lo cual no fue posible, como: los algoritmos utilizados (RANSAC y ICP) requerían

parámetros muy variables dependientes del objeto a registrar. Por ejemplo, las nubes de puntos de objetos simétricos solían empalmarse con el modelo anterior en vez de complementarse y los objetos con más de una superficie plana tendían a empalmar estas equivocadamente.

Un aspecto no mencionado anteriormente es que todos los parámetros variables son modificables a través de un archivo de configuración de texto plano, el cual beneficia la usabilidad del sistema permitiendo hacer muchas pruebas usando diferentes algoritmos y parámetros sin necesidad de recopilar el sistema. Algo deseable es implementar una interfaz que haga más amigable su uso.

Además del reconocimiento de objetos, la visión ayuda a reconocer otras situaciones de la vida diaria de humano, como una situación de emergencia, detectar por medio de señales un llamado o identificar instrucciones que estén plasmadas en imágenes. Cada uno de estos problemas se resuelven de maneras muy diversas, utilizando diferentes técnicas y algoritmos por lo que para desarrollar este tipo de aplicaciones es muy necesaria la intervención del humano y su intuición.

7.0.5. Trabajo a futuro

Uno de los puntos que se desea, es el diseño e implementación de un proceso para realizar el registro de un objeto de manera autónoma por medio del robot, en este trabajo se diseño una posible solución, sin embargo, no se logro la autonomía.

Agregar más métodos de segmentación de objetos tomando en cuenta diversos escenarios donde un objeto es sujetado por una pinza róbotica o una mano de humano, entre otros.

Trabajar en un sistema de visión que dé retroalimentación al robot en todo el tiempo sobre qué objetos está identificando. Para esto sería necesario utilizar implementaciones muy rápidas y que no comprometan las demás funciones del robot a pesar de la gran cantidad de recursos que implica trabajar con imágenes, una de las soluciones es aprovechar la tecnología como GPU y CUDA para agilizar estas tareas.

CAPÍTULO 8

APÉNDICE

8.1. Aplicaciones de visión comunicacional en RoboCup

A continuación se describen otras aplicaciones relacionadas con visión computacional que fueron realizadas para resolver pruebas de la competencia RoboCup

RoboZoo (interacción hombre - robot)

Esta prueba trata sobre la interacción humano -robot. Lo que se realizó para interactuar con el público fue mostrar a Justina bailando, cantando, hipnotizando, dichas actividades eran instrucciones indicadas al robot a través de unas paletas que tenían dos características, la primera era un marco de color que indicaba el idioma con el cual el robot debía comunicare con la persona y la segunda era la actividad ilustrada por una imagen representativa.

La detección de las actividades se realiza en dos pasos, el primero es identificar y segmentar la paleta por medio del marco, y el segundo es identificar las actividades utilizando SIFT en la imagen del marco.

	cantar	bailar	prender luz	hipnotizar	no detecto
cantar	6	2	0	0	2
bailar	0	8	0	0	2
prender luz	0	0	5	0	5
hiponitzar	0	0	0	7	3

Tabla 8.1: Detección de acciones Robot Zoo

Situación de emergencia: Accidente en casa

En una casa donde se encuentra una sola una persona de la tercera edad quien experimenta un accidente y requiere ser auxiliado inmediatamente. El robot debe de detectar el accidente ya sea escuchando o viendo a la persona caer. Al detectar el accidente el robot debe de moverse hacia la persona y notificar a una ambulancia.

La detección de una persona cayendo se realizó suponiendo que información de la distancia entre los frames que contiene la caída seria variada de una manera brusca en el momento que una persona cayera en un escena.

Por otro lado es posible que el robot este "mirando" a otra parte cuando ocurre la caída, es por eso que la prueba permite que la persona después del accidente realice señales moviendo sus brazos pidiendo auxilio. La detección de estas señales se realizó calculando la diferencia en color entre escena y escena, posteriormente se realiza un filtro de color tratando de obtener aquellos movimientos en objetos que asemejen el color de la piel con lo que se obtiene una máscara de la escena que indica donde hubo movimiento. Finalmente se descartan las áreas muy grandes mediante un umbral y se indica que hay una señal de auxilio cuando en esta máscara queda alguna región positiva.

Situación de emergencia: Fuego en el departamento

Se espera que un robot de servicio pueda reaccionar en una situación de emergencia detectándola y auxiliando a los humanos.

Para esta prueba el robot debe de identificar cuando hay fuego en una cocina, realizar un plan de rescate ante la emergencia el cual incluía buscar personas y guiarlas hacia la salida de emergencias.

Para la detección del incendio se utilizaron técnicas de visión computacional para buscar humo en las imágenes capturadas por el robot.

El humo fue detectado analizando en flujo óptico entre escena y escena, una vez detectado era segmentado en la imagen para indicar donde se encontraba con la ayuda de la información de profundidad.

BIBLIOGRAFÍA

[1] Diane E Papalia and Sally Wendkos Olds. *Psicología*. McGraw-Hill Interamericana de México, 2009. 1

[2] D. Marr. *Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information*. WH San Francisco: Freeman and Company, 1982. 1

[3] Jin-Fang Shi, Cai-Jian Hua, and Guo-Hui Li. A simplifying method of vision attention simulating human vision in machine vision system. In *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, volume 6, pages 3097–3100. IEEE, 2010. 1

[4] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001. 1, 5, 6

[5] Parisa Kamani, Elaheh Noursadeghi, Ahmad Afshar, and Farzad Towhidkhah. Automatic paint defect detection and classification of car body. In *Machine Vision and Image Processing (MVIP), 2011 7th Iranian*, pages 1–6. IEEE, 2011. 1

[6] Alberto Tellaeche and Beatriz Robles. 3d machine vision and artificial neural networks for quality inspection in mass production pieces. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–4. IEEE, 2010. 1

[7] Chuho Yi, Young Ceol Oh, Il Hong Suh, and Byung-Uk Choi. Indoor place classification using robot behavior and vision data. *International Journal of Advanced Robotic Systems*, 8(5):49–60, 2011. 1

[8] S. Ruiz-Correa, R.W. Sze, H. J. Lin, L.G. Shapiro, M. L. Speltz, and M.L. Cunningham. Classifying craniosynostosis deformations by skull shape imaging. In

Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on, pages 335–340, 2005. 1

[9] N. Ayache. Medical image analysis a challenge for computer vision research. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1255–1256 vol.2, 1998. 1

[10] S. Bharathi, P.D. Shenoy, V.J. Shreyas, R.P. Anirudh, S.M. Sanketh, K.R. Venugopal, and L.M. Patnaik. Performance analysis of segmentation techniques for land cover types using remote sensing images. In *India Conference (INDICON), 2012 Annual IEEE*, pages 775–780, 2012. 1

[11] Y. Sakamoto, K. Tajiri, T. Sawai, and Y. Aoki. Three-dimensional imaging of objects in accumulated snow using multifrequency holography. In *Geoscience and Remote Sensing, IEEE Transactions on*, volume 26, pages 430–436, 1988. 1

[12] V.S. Tseng, Ja-Hwung Su, Hao-Hua Ku, and Bo-Wen Wang. Intelligent concept-oriented and content-based image retrieval by using data mining and query decomposition techniques. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 1273–1276, 2008. 1

[13] Carlos Merino-Gracia, Majid Mirmehdi, José Sigut, and José L González-Mora. Fast perspective recovery of text in natural scenes. In *Image and Vision Computing*, volume 31, pages 714–724. Elsevier, 2013. 2

[14] Junsik Lim, Soohyung Kim, Jonghyun Park, Gueesang Lee, HyungJeong Yang, and ChilWoo Lee. Recognition of text in wine label images. In *Pattern Recognition, 2009. CCPR 2009. Chinese Conference on*, pages 1–5, 2009. 2

[15] Justin Testa. The vision-guided robot grows up-a look at the new machine vision technology for robots. In *Industrial Robot: An International Journal*, volume 22, pages 13–15. MCB UP Ltd, 1995. 2

[16] Geoff Collins. Sophisticates image processing controls assembly robot. In *Industrial Robot: An International Journal*, volume Vol. 27 Iss: 6, pages 445–448, 2000. 2

[17] Pedro U Lima, Pedro Santos, Ricardo Oliveira, Aamir Ahmad, and João Santos. Cooperative localization based on visually shared objects. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 350–361. Springer, 2011. 2

[18] Mohsen Malmir and Saeed Shiry. Object recognition with statistically independent features: A model inspired by the primate visual cortex. In Jacky Baltes, MichailG. Lagoudakis, Tadashi Naruse, and SaeedShiry Ghidary, editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 204–214. Springer Berlin Heidelberg, 2010. 2

[19] E. Machida, Meifen Cao, T. Murao, and H. Hashimoto. Human motion tracking of mobile robot with kinect 3d sensor. In *SICE Annual Conference (SICE), 2012 Proceedings of*, pages 2207–2211, 2012. 2

[20] Mauricio Correa, Javier Ruiz-del Solar, S Isao Parra-Tsunekawa, and Rodrigo Verschae. A realistic simulation tool for testing face recognition systems under real-world conditions. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 13–24. Springer, 2011. 2

[21] Robocup Federation. Accedido en 2013-9-20 a http://www.robocup.org/. 4

[22] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 5, 7, 8

[23] L Enrique Sucar and Giovani Gómez. *Visión Computacional*. Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México. 6

[24] Robert M Haralock and Linda G Shapiro. *Computer and robot vision*. Addison-Wesley Longman Publishing Co., Inc., 1991. 7

[25] A. Pacheco. *Adeacuación de técnicas de descripción visual utilizando información 3D y su aplicación en robótica*. PhD thesis, UNAM, 2011. 9

[26] Marius Muja, Radu Bogdan Rusu, Gary Bradski, and David G Lowe. Rein-a fast, robust, scalable recognition infrastructure. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2939–2946. IEEE, 2011. 9, 10

[27] D.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 vol.2, 1999. 11

[28] DavidG. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 11, 13, 16

[29] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006. 17, 19

[30] Georg Arbeiter, Steffen Fuchs, Richard Bormann, Jan Fischer, and Alexander Verl. Evaluation of 3d feature descriptors for classification of surface geometries in point clouds. pages 1644–1650, 2012. 23

[31] A. Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *Robotics Automation Magazine, IEEE*, 19(3):80–91, Sept 2012. 23, 24, 31, 32

[32] Robotics Group of the University of León. 3d object recognition (descriptors). Accedido en 2015-01-06 http://sindominio.net/ash. 23, 24

[33] R.B. Rusu, N. Blodow, Z.C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391, Sept 2008. 24, 25, 27

[34] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 643–650. IEEE, 2008. 24, 25, 26, 27

[35] PCL. Point cloud library. Accedido en 2015-01-06 http://pointclouds.org. 24, 44, 57

[36] 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. 24, 27, 28

[37] 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. 24, 30, 31

[38] Matthias Grundmann, Franziska Meier, and Irfan Essa. 3d shape context and distance transform for action recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008. 24

[39] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62. ACM, 2010. 24

[40] 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. 24, 29, 30

[41] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. 24

[42] 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. 25

[43] Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. *OUR-CVFH–Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. Springer, 2012. 25, 34

[44] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992, Dec 2011. 25

[45] Zoltan-Csaba Marton, Dejan Pangercic, Radu Bogdan Rusu, Andreas Holzbach, and Michael Beetz. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 365–370. IEEE, 2010. 29

[46] FLANN. Fast library for approximate nearest neighbors. Accedido en 2015-01-06 http://www.cs.ubc.ca/research/flann. 32

[47] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007. 33

[48] Luz Martínez, Patricio Loncomilla, and Javier Ruiz-del Solar. Object recognition for manipulation tasks in real domestic settings: A comparative study. *18th RoboCup International Symposium*, 2014. 34, 35, 36

[49] Javier Ruiz-del Solar, Patricio Loncomilla, and Christ Devia. Fingerprint verification using local interest points and descriptors. In José Ruiz-Shulcloper and WalterG. Kropatsch, editors, *Progress in Pattern Recognition, Image Analysis and Applications*, volume 5197 of *Lecture Notes in Computer Science*, pages 519–526. Springer Berlin Heidelberg, 2008. 34

[50] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R.B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 585–592, Nov 2011. 34

[51] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 14, pages 239–256, Feb 1992. 36, 38

[52] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. 30(11):1311–1327, 2011. 36

[53] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5031–5037. IEEE, 2011. 36

[54] PCL. Registration. Accedido en 2015-01-06 http://pointclouds.org/documentation/tutorials/registrationapi.php. 37

[55] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, volume 24, pages 381–395. ACM, 1981. 38

[56] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE, 1991. 38

[57] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152, 2001. 39

[58] Kinect de XBOX 360. Accedido en 06-01-2015 http://support.xbox.com/en-US/xbox-360/system/manual-specs. 43

[59] SensorKinect. Accedido en 2015-01-06 https://github.com/PrimeSense/Sensor. 43

[60] OpenNI. Accedido en 2015-01-06 http://www.openni.ru/openni-sdk/index.html. 44

[61] OpenCV. Accedido en 2015-01-06 http://opencv.org/. 44