Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Reconocimiento de Objetos en Imágenes RGB-D

Tesis para Licenciatura en Ciencias de la Computación de la Universidad de Buenos Aires

Alumna: Nadia Mariel Heredia Favieri

Director: Francisco Gómez Fernández

Buenos Aires, 23 de Marzo de 2015

Resumen

En esta tesis, consideramos el problema del reconocimiento de objetos, siendo éste fundamental en robótica y visión por computadora, y desafiante por su dificultad. Recientemente ha surgido una gran cantidad de sensores de profundidad y color de bajo costo que funcionan en tiempo real, como Kinect y Xtion, representando una buena oportunidad de incrementar la robustez del reconocimiento de objetos.

Implementamos un sistema de reconocimiento de objetos, que puede ser instanciado con distintos métodos en cada etapa, que funciona con datos adquiridos con sensores RGBD del estilo de Kinect. Analizamos los descriptores FPFH, SHOT y Spin-Images, y estudiamos también cuál es la mejor distancia para comparar cada descriptor y con qué método establecer correspondencias.

Experimentamos detalladamente cambiando cada etapa del sistema, variando los parámetros y métodos, viendo cómo influía esto en el desempeño del sistema y en los tiempos. Observamos que el uso de más puntos lleva a mayores precisiones a costa del tiempo, y que un tamaño más chico en los radios de los descriptores y las normales hace que se acelere la generación de descriptores pero que la búsqueda de correspondencias sea más lenta.

Por último, analizamos el rendimiento del sistema implementado en dos bases de datos con información de ground truth, una siendo un caso ideal y la otra un caso real. Encontramos que el sistema puede ser adaptado a las circunstancias dadas por las bases, otorgando buenos compromisos entre exactitud y tiempo en la gran mayoría de los casos. Vimos que para la primera base, todos los descriptores otorgan un buen desempeño, siendo Spin-Images la mejor opción, y para la segunda base ya no todos se comportaban de igual manera, siendo FPFH muy lento, y SHOT y Spin-Images buenas opciones en cuanto a los valores de exactitud y tiempos.

Abstract

In this thesis, we consider the object recognition problem, which is fundamental in robotics and computer vision, and challenging because of its difficulty. Recently, low cost depth and color sensors which function in real time have emerged, such as Kinect and Xtion, representing a great opportunity for incrementing the robustness of object recognition.

We implemented an object recognition system, which can be instantiated with different methods in each step, and works with data acquired with RGBD sensors like Kinect. We analyzed the descriptors FPFH, SHOT and Spin-Images, and also studied which distance is better for comparing each descriptors, and with which methods correspondences should be established.

We experimented in detail changing each step of the system, and varying the parameters and methods, seeing how this would make an impact on the system performance and times. We could observe that the use of more points leads to better precisions at the expense of time, and that a smaller size in the descriptor and normal radii makes the descriptor generation faster, but correspondences search slower.

Lastly, we analyzed the performance of the implemented system in the databases with ground truth information, the first one being an ideal case, and the latter a real case. We found that the system provides a good compromise between accuracy and time for the most part. We saw that for the first database, all descriptors provided a good performance, Spin-Images being the best option, and that for the second database not all af them behaved the same way, FPFH being too slow, and SHOT and Spin-Images being good options regarding accuracy and time values.

Índice general

Capítulo 1

Introducción

En este trabajo nos enfocamos en determinar si un objeto está presente en una imagen. Este problema, el reconocimiento de objetos, es esencial y desafiante, y es una gran parte de áreas de investigación en visión por computadora y robótica. En las últimas décadas, una gran variedad de enfoques y algoritmos han sido propuestos y aplicados a este problema ([Low99], [BMP02], [BTVG06]), resultando en un avance significativo en esta tarea. Gran parte de la dificultad del problema proviene de los diferentes cambios que puede experimentar el objeto original en la imagen donde se lo quiere encontrar. Un mismo objeto puede verse de muchas formas distintas según el punto de vista desde el cuál se haya capturado la imagen que lo representa. Además, puede estar deformado, parcialmente oculto, junto a muchos otros objetos similares, rotado, trasladado, entre otros.

Recientemente han surgido sensores de profundidad y color de bajo costo que funcionan en tiempo real, entre ellos Kinect de Microsoft ([Kin]) y Xtion de Asus ([Xti]). Los sensores o cámaras RGB-D capturan información de color del mundo , y las distancias entre el sensor y los objetos del mundo, haciendo posible la obtención de datos tridimensionales de forma simple y accesible. Este tipo de cámaras representa una oportunidad de incrementar enormemente la robustez del reconocimiento de objetos. Además, existen amplias bases de datos con imágenes RGB-D ([Was], [Mü]) disponibles, es decir con color y profundidad, facilitando el estudio y desarrollo de nuevos métodos para la resolución del reconocimiento de objetos, y para otras áreas de visión por computadora. Algunos enfoques recientes en el reconocimiento de objetos usando imágenes RGB-D, se basan en el uso de la combinación de información provista, teniendo en cuenta tanto el color como la profundidad. Esta combinación de información geométrica, proveniente

de la profundidad, con apariencia, proveniente del color, puede otorgar un mejor rendimiento que usando solo una parte de la información [NOC+12] [ZBVH09].

Un sistema de reconocimiento puede dividirse en tres etapas: extracción de puntos de interés, generación de descriptores y búsqueda de correspondencias. La generación de descriptores es la extracción de representaciones significativas de los objetos a partir de observaciones de alto nivel como imágenes, videos y nubes de puntos. Los descriptores proveen información distintiva, y es deseable que tengan robustez a ciertos cambios como la escala, la rotación, el punto de vista, la iluminación, entre otros. Esto implica que el descriptor sigue siendo útil aún si el objeto a buscar se encuentra alterado con dichos cambios.

El objetivo principal de este trabajo es implementar, estudiar y evaluar un sistema de reconocimiento de objetos en tres dimensiones, usando imágenes capturadas por cámaras RGB-D. En particular el sistema funcionará en sensores de profundidad de bajo costo como Kinect y Xtion. También reconocerá objetos tridimensionales cuya forma es conocida de antemano. El sistema de reconocimiento propuesto es en realidad un esquema de sistema, al variar los descriptores y métodos utilizados, se obtienen múltiples sistemas de reconocimiento de objetos diferentes. Una meta de este trabajo es comparar los distintos sistemas instanciados en bases de datos de objetos usando información de *ground truth* para evaluar la exactitud de los resultados.

Los descriptores que utilizamos son *SHOT* [TSDS10], *FPFH* [RBB09], y *Spin-Images* [Joh97], que capturan información de la superficie mediante la acumulación de parámetros usando histogramas. SHOT acumula cosenos de ángulos entre normales y FPFH acumula ángulos representando la diferencia entre normales. Spin-Images por otro lado usa un histograma de dos dimensiones para acumular dos medidas de distancia entre puntos.

Realizamos un análisis experimental de la capacidad informativa de los distintos descriptores y sus distancias asociadas. Vimos que usar el cociente entre los dos vecinos más cercanos de un punto para asociarlo con otro daba mejores resultados que usar solamente la distancia al punto más cercano.

Analizamos también el desempeño de distintas instanciaciones del sistema en dos bases de datos, siendo una un caso ideal de reconocimiento, con mucho detalle y poco ruido, y la otra un caso real con poco detalle, mucho ruido, información faltante y de sobra.

Consideramos tanto la exactitud del sistema, dada por la diferencia entre el resultado obtenido y el esperado, como el tiempo que demora. Obtuvimos que en la primera base el rendimiento fue bueno con cualquier elección de descriptor, siendo Spin-Images el que mejor exactitud otorgaba con mejores tiempos similares a los dados por los otros descriptores, y que en la segunda base el rendimiendo general ya no fue tan bueno. En ésta FPFH demoró demasiado, y el uso de Spin Images y SHOT fue comparable, siendo Spin-Images la mejor opción porque otorga para esta base mejor exactitud y menor tiempo.

Organización

El trabajo está organizado de la siguiente manera:

- En el Capítulo 2 explicaremos el concepto de nube de puntos, utilizado a lo largo del trabajo, la adquisición de datos a partir de sensores de profundidad, y algunos conceptos básicos del procesamiento de las nubes.
- En el Capítulo 3 trataremos el tema de descriptores, explicando aquellos que decidimos usar en nuestro sistema, y haciendo una evaluación de éstos junto con sus distancias.
- En el Capítulo 4 se presentará el concepto de un sistema de reconocimiento de objetos, explicando en particular las características de nuestra implementacion.
- En el Capítulo 5 discutiremos los resultados obtenidos al utilizar el sistema implementado en dos bases de datos con los distintos descriptores. Analizaremos la precisión según la información de ground truth provista por cada base, y el tiempo, considerando las distintas etapas del sistema.
- Finalmente, en el Capítulo 6 se presentarán las conclusiones del trabajo, y se verán ideas para trabajo futuro.

Capítulo 2

Representación de Datos

En este capítulo se explicará la representación de datos usada a lo largo de este trabajo, la nube de puntos, juntos con su relación las imágenes RGB-D; así como también su adquisición, y las propiedades fundamentales de su procesamiento.

2.1. Introducción

Los humanos obtienen una gran cantidad de información sobre el mundo a través del sentido de la vista. Al reflejarse la luz en los objetos del mundo puede obtenerse una imagen en la retina de cada ojo. De este par de imágenes se deduce gran parte de la estructura de un ambiente 3D.

Una imagen de dos dimensiones representa en general una *proyección* de una porción del mundo tridimensional. Una imagen 2D digital consiste simplemente en una matríz numérica cuyos valores representan intensidades. Esta intensidad puede tener uno o más canales, según el modelo de color usado. En general si tiene un solo canal, éste representa la intensidad del valor de gris, y en el caso de tener más canales, representan colores, transparencia, saturación, etc. El modelo de color RGB es uno de los más utilizados, posee tres canales, uno para el color rojo, otro para el verde, y otro para el azul.

2.2. Mallas Poligonales

Una malla poligonal es una colección de vértices, ejes y caras que definen la forma de un objeto en 3 dimensiones. Un vértice corresponde a un punto en el espacio 3D, un eje es una conexión entre dos vértices, y una cara es un conjunto cerrado de ejes, siendo en general un triángulo o cuadrilátero. En la Figura 2.1 pueden verse dos ejemplos de mallas poligonales de distintas densidades (distancia promedio entre ejes) y polígonos representando el mismo modelo de un conejo.

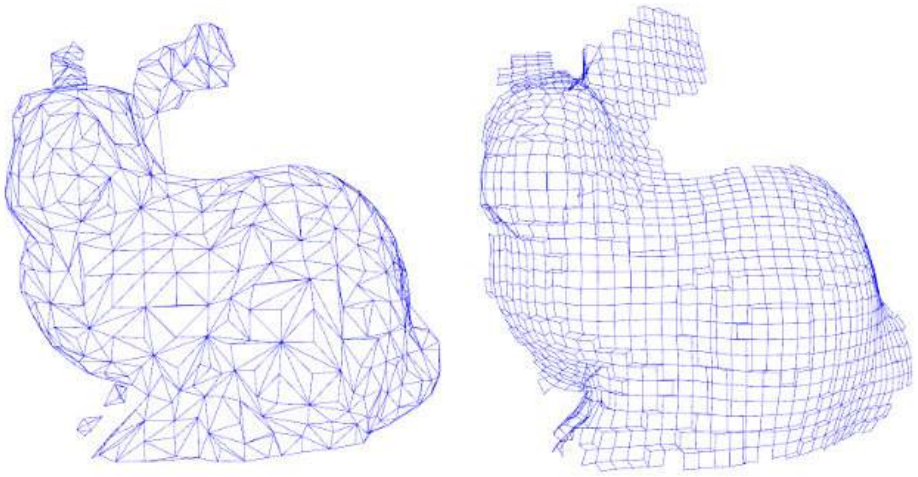
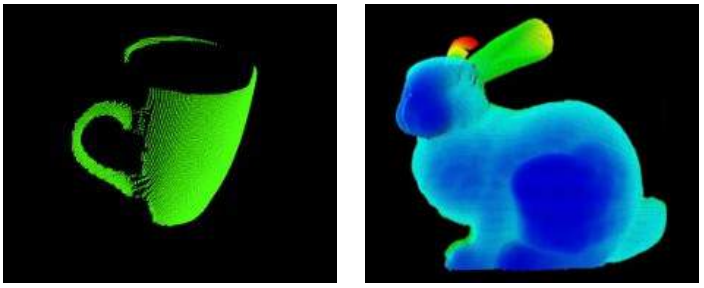


FIGURA 2.1: Ejemplos de mallas poligonales, el primer caso tiene poca definición, mientras que el segundo es mucho más detallado y provee una representación más suave.

2.3. Nubes de Puntos

Una nube de puntos es una colección de puntos tridimensionales. Ésta puede ser generada de forma artificial (siendo sintética), o provenir de capturas de elementos del mundo. Ejemplos de nubes de puntos pueden observarse en la Figura 2.2, representando una porción de una taza y un modelo detallado de un conejo. Las coordenadas $\{p_x, p_y, p_z\}$ de cualquier punto p de la nube, están dadas con respecto a un sistema de coordenadas fijo. Si la nube representa datos del mundo, entonces el origen del sistema de coordenadas suele ser el sistema de captura utilizado. En este caso, el valor de cada punto p representa la distancia desde el origen hasta la superficie donde el punto fue capturado. Las nubes de puntos pueden incluir mucha más información que solo las posiciones 3D de los puntos, como color, intensidad, punto de vista.



(A) (B)

FIGURA 2.2: Nubes de puntos, representando una taza (A), y un conejo (B).

Point Cloud Library (PCL)

Trabajaremos a lo largo de esta tesis con la biblioteca de gran escala **PCL** [1] [RC11] [AMT+12], usando el formato de nube de puntos provisto en ella.

La infraestructura de PCL contiene numerosos algoritmos incluyendo filtros, estimación de características, reconstrucción de superficies, ajuste de modelos y segmentación. Los algoritmos pueden ser usados por ejemplo, para filtrar valores atípicos de datos con ruido, unir nubes de puntos 3D, segmentar partes relevantes de una escena, extraer puntos de interés y computar descriptores para luego reconocer objetos en el mundo, basándose en la información geométrica y de color, y para crear superficies a partir de nubes de puntos para facilitar la visualización.

La biblioteca PCL se distribuye bajo una licencia BSD y es software de código abierto, además es multi-plataforma funcionando en Linux, Android, Windows, iOs y MacOS. Para simplificar el desarrollo, PCL está separado en una serie de bibliotecas de código más chicas, que pueden ser compiladas por separado. Esta modularidad es importante para la distribución de PCL en plataformas con restricciones de tamaño o cómputo.

Las figuras 2.1 y 2.2 forman parte de las imágenes de ejemplo de PCL.

2.4. Captura de datos

Hay muchas formas de estimar distancias a los objetos del mundo y convertirlas a puntos 3D, dadas por los distintos tipos de sensores de profundidad, entre ellos cámaras estéreo, sensores de tipo time-of-flight, y de luz estructurada. El tipo de sensor a elegir

[1] Point cloud library `http://pointclouds.org/about/`

depende de la situación, ya que cada sensor otorga distintos valores de precisión en las medidas de profundidad, resolución, y frecuencia de captura, entre otros. En este trabajo nos interesa utilizar sensores RGB-D, que adquieren información de color además de profundidad.

Sensores RGB-D

Los sensores RGB-D combinan información de color representada en RGB con información de profundidad, por cada píxel. Esto permite obtener la textura y geometría de los objetos del mundo. La tecnología de este tipo de cámaras está avanzando a gran velocidad, y en los últimos años han surgido sensores de este tipo de consumo con bajo costo, haciendo esta tecnología mucho más accesible ya que previamente los sensores de profundidad eran muy costosos.

Un sensor RGB-D contiene

- Una cámara RGB que captura una imagen 2D del mundo en color.
- Una cámara de profundidad formada por un emisor infrarrojo y un sensor infrarrojo, que sirve para estimar la profundidad de los objetos del mundo.

Esta cámara de profundidad es de luz estructurada, para la estimación de los valores de profundidad, el emisor proyecta un patrón de luz infrarroja, que luego el capturado por la cámara infrarroja. A partir de la diferencia entre el patrón proyectado y el capturado, el sensor estima la distancia entre éste y el mundo. La información de profundidad luego se sincroniza con la obtenida a partir de la cámara RGB calibrada.

Imágenes RGB-D

El resultado es un par de imágenes 2D, conformado por una imagen RGB y una imagen de profundidad, que en conjunto llamamos *imagen RGB-D*. Este par permite tener información más completa del mundo tridimensional que la provista con una única imagen 2D. La imagen RGB aporta la textura del mundo, mientras que la imagen de profundidad añade la geometría, que luego, gracias a los parámetros de calibración de la cámara (centro óptico más distancia focal), nos permite generar la nube de puntos 3D.

Sensor Kinect

Kinect es un sensor desarrollado por Microsoft, originalmente diseñado como parte de la consola Xbox 360, para proveer una forma de interacción natural al no existir la necesidad de tener contacto físico con un controlador. La interfaz de usuario provista reconoce gestos, comandos de voz, objetos e imágenes. Dado su bajo precio y accesibilidad, se ha vuelto un sensor muy usado en el campo de visión por computadora.

El controlador contiene [2] 4 micrófonos, una cámara RGB standard, una cámara de profundidad, formada por un emisor infrarrojo y un sensor infrarrojo, y un motor de inclinación, ubicados como muestra la Figura 2.3.

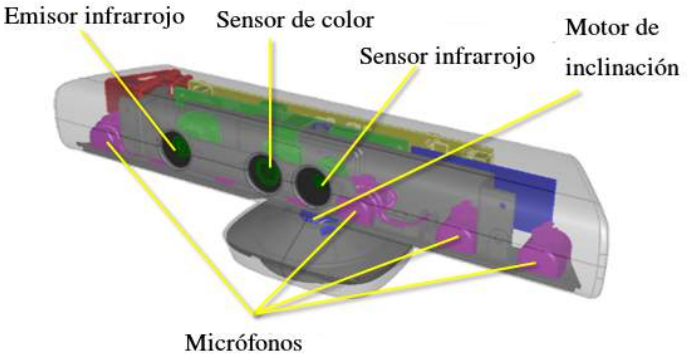
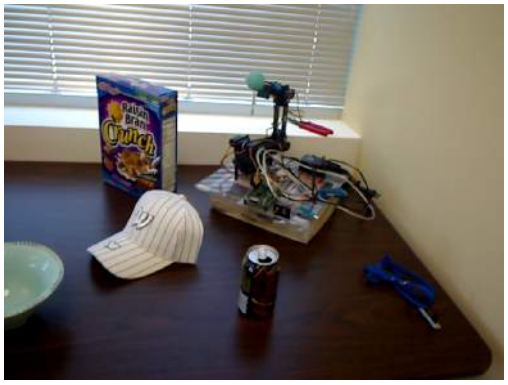


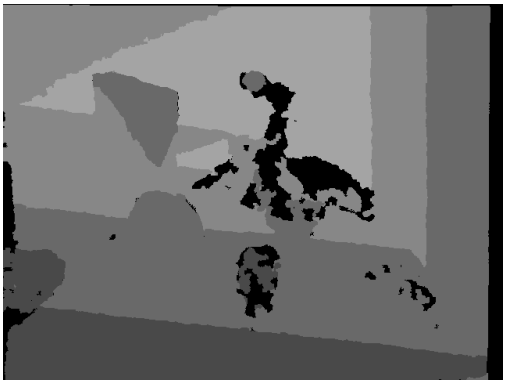
FIGURA 2.3: Componentes del sensor Kinect

El sensor kinect provee pares de imágenes sincronizadas de 640 x 480 píxeles a 30 cuadros por segundo. Un ejemplo de este tipo de imágenes es el presentado en la Figura 2.4, en la imagen de profundidad los colores más claros se corresponden a puntos más lejanos, los más oscuros a puntos más cercanos, y los puntos negros representan falta de información de profundidad.

[2]Especificaciones del sensor Kinect https://msdn.microsoft.com/en-us/library/jj131033.aspx

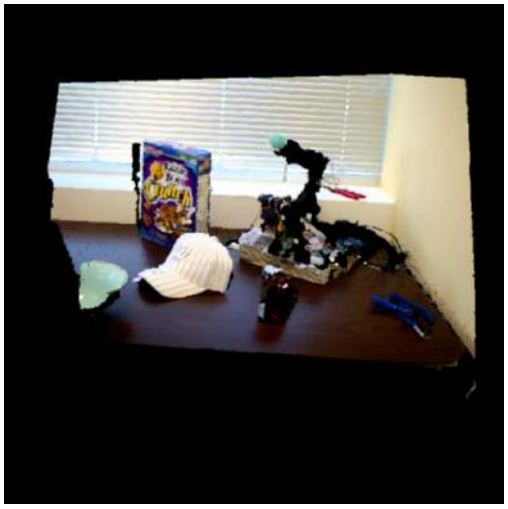


(A) Imagen color RGB con la textura del mundo

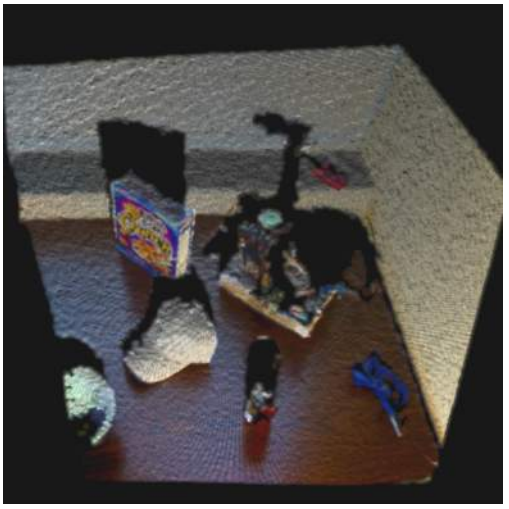


(B) Imagen de profundidad con la información geométrica del mundo

FIGURA 2.4: Ejemplo de imágenes capturadas con Kinect



(A) Vista frontal



(B) Vista superior

FIGURA 2.5: Dos vistas de la nubes de puntos generada a partir de las imágenes RGB-D en la Figura 2.4.

Obtención de una nube de puntos a partir de imágenes RGB-D

Como nuestro sistema trabaja con nubes de puntos, interesa obtener la nube correspondiente a un par de imágenes RGB-D, formado por la imagen de color *rgb* y la de

profundidad *depth*. A partir de la imagen de profundidad *depth* conseguimos una nube de puntos, donde cada punto $p = \{p_x, p_y, p_z\}$ lo obtenemos de la siguiente forma

$$\begin{aligned} p_x &= (x - c_x) * \textit{depth}\,(x, y)/f_x \\ p_y &= (y - c_y) * \textit{depth}\,(x, y)/f_y \\ p_z &= \textit{depth}\,(x, y) \end{aligned} \tag{2.1}$$

donde

- c_x y c_y son los centros de la cámara en x y en y
- f_x y f_y son las distancias focales de la cámara en x y en y

Para incluir la información de color, usamos la imagen *rgb* agregando entonces el campo rgb correspondiente al punto a la nube.

En la representación de puntos con color de la PCL [3], la información de color de un punto se empaqueta en un entero sin signo y luego se reinterpreta como un float. De este entero sin signo, se utilizan solo los 3 bytes más bajos, colocando los valores de r, g y b en orden, con b en el byte más bajo.

Combinando el par de imágenes RGB-D de la Figura 2.4 obtuvimos una nube de puntos con información de color para cada punto tridimensional, visible en la Figura 2.5 desde dos puntos de vista distintos.

2.5. Procesamiento

Explicamos a continuación algunos conceptos útiles del procesamiento de las nubes de datos, para mejorar la comprensión de capítulos futuros.

2.5.1. Vecindad de un punto

Para entender la geometría alrededor de un punto p, se usa el entorno del punto que representa una aproximación local a la superficie que lo rodea. El entorno está formado por vecinos más cercanos, pudiendo ser una cantidad k fija, o todos los vecinos dentro de un radio r del punto. Esta determinación de puntos más cercanos depende de la medida de distancia que se use, algunas opciones son norma 1 o $\mathcal{L}_1$ (Ecuación 2.2), norma 2,

[3] PCL PointXYZRGB http://docs.pointclouds.org/1.7.0/structpcl_1_1_point_x_y_z_r_g_b.html

norma euclídea o $\mathcal{L}_2$ (Ecuación 2.3), y norma χ cuadrado o χ^2 (Ecuación 2.4), definidas a continuación

$$\mathcal{L}_1(p, q) = \sum_{i=1}^{n} |p[i] - q[i]| \tag{2.2}$$

$$\mathcal{L}_2(p, q) = \sum_{i=1}^{n} (p[i] - q[i])^2 \tag{2.3}$$

$$\chi^2(p, q) = \sum_{i=1}^{n} \begin{cases} \frac{(p[i] - q[i])^2}{(p[i] + q[i])} & \text{si } (p[i] + q[i]) \neq 0 \\ 0 & \text{si no} \end{cases} \tag{2.4}$$

donde p y q son vectores numéricos, en este caso las coordenadas de puntos en el espacio; y n es la dimensión de dichos vectores, en este caso 3.

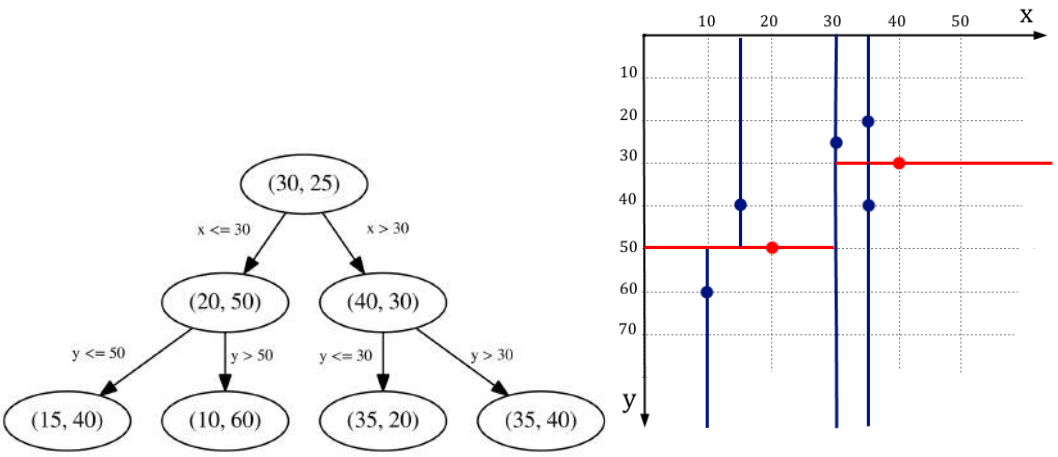
Para determinar los k puntos más cercanos a p, se deben considerar las distancias desde p hacia todos los otros puntos p_i ordenadas de menor a mayor, y tomar los correspondientes a las primeras k distancias. Este proceso de búsqueda puede acelerarse usando una estructura de datos especial para organizar puntos con muchas dimensiones, un **árbol *k*-d** , explicado a continuación. La estructura es particularmente adecuada para la búsqueda de vecino más cercano, y búsqueda por rango, para obtener todos los vecinos en un radio dado.

Árbol k-d

Un árbol k-d, abreviación de árbol k dimensional, es un árbol binario en el cual cada nodo es un punto de k dimensiones. Cada nodo $\boldsymbol{p}$ parte el espacio según una dimensión específica $i, 1 \leq i \leq k$, es decir que los nodos $\boldsymbol{n}$ cuyo valor de la dimensión i sea menor que el de $\boldsymbol{p}$, $\boldsymbol{n}_i < \boldsymbol{p}_i$, estarán ubicados en el subárbol izquierdo de $\boldsymbol{p}$, mientras que los nodos $\boldsymbol{n}$ con valores más grandes que $\boldsymbol{p}$ en esa dimensión i, $\boldsymbol{n}_i > \boldsymbol{p}_i$ estarán en el subárbol derecho.

En la Figura 2.6 se muestra un ejemplo de un árbol k-d con puntos de 2 dimensiones. La raíz particiona los nodos según la coordenada x, todos los nodos en su subárbol izquierdo tienen un valor de x menor o igual a 30, y todos los del subárbol derecho tienen un valor de x mayor a 30. Los nodos del segundo nivel particionan el espacio según la coordenada y. Si hubieran más nodos, en el tercer nivel se volvería a particionar usando la coordenada x.

Para el caso tridimensional, la partición se hace de la misma manera, teniendo en cuenta la coordenada z, un ejemplo 3D se puede ver en la Figura 2.7.



(A) Arbol k-d donde la raíz particiona según la coordenada x y sus hijos según la coordenada y.

(B) Divisiones del espacio 2D dadas por los puntos del árbol mostrado en A.

FIGURA 2.6: Ejemplo de árbol k-d de 2 dimensiones y la división espacial asociada. Las particiones se muestran con líneas, donde las azules corresponden a la coordenada x, y las rojas a la coordenada y.

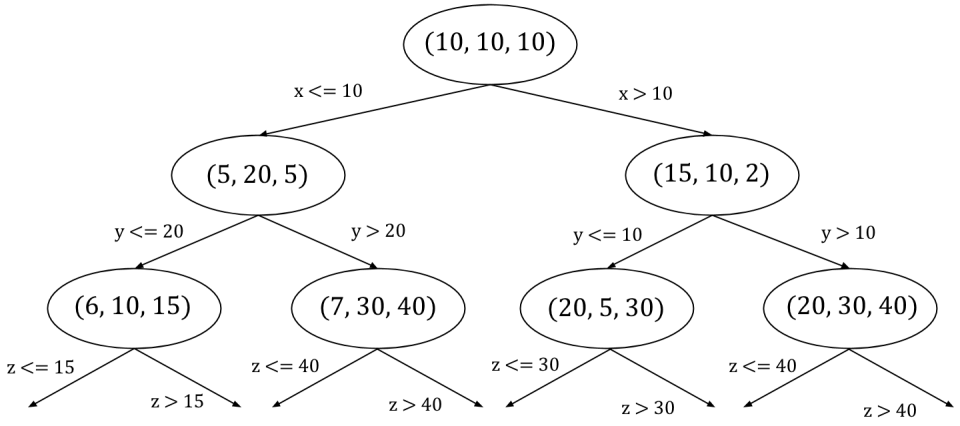


FIGURA 2.7: Ejemplo de árbol k-d de 3 dimensiones. Se muestran solo los primeros tres niveles que corresponden a divisiones en las coordenadas x, y, y z respectivamente.

Densidad de una nube

La densidad de una nube puede verse como la cantidad de puntos en las vecindades para un radio dado, si esta cantidad es la misma para todos los puntos de la nube, entonces decimos que la densidad es uniforme. La densidad (el valor promedio o de la mediana

de las cantidades de vecinos de todos los puntos) determina la cantidad de detalle de la superficie que la nube contiene, una densidad baja significa que los puntos están muy separados, dando poco detalle; y una densidad alta significa que hay muchos puntos muestreados de la superficie, haciendo que la representación sea más precisa.

2.5.2. Normal de un punto

La normal a un plano se define como el vector unitario perpendicular a él, y la normal a la superficie en un punto como el vector perpendicular al plano tangente a la superficie en ese punto.

Para nubes de puntos, podemos obtener la normal de un punto particular $\boldsymbol{p}$, ajustando un plano a los puntos de su entorno, estimando entonces el plano tangente que permite calcular su normal $\vec{\boldsymbol{n}}_p$, también denotada como $n\,(p)$. Esta estimación puede verse como un problema de ajuste de plano usando cuadrados mínimos, donde se minimiza la distancia de los vecinos de p al plano. Puede verse un ejemplo de la estimación de la normal para un punto en la Figura 2.8.

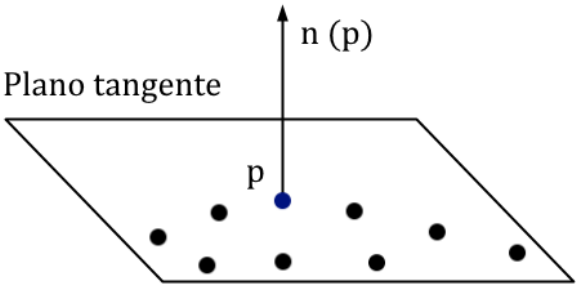


FIGURA 2.8: Ejemplo de estimación de la normal $n\,(p)$ para un punto p, mostrado en azul, considerando sus vecinos, mostrados en negro, para estimar el plano tangente.

Este plano se representa con un punto $\boldsymbol{x}$ y un vector normal $\vec{\boldsymbol{n}}$, y la distancia de un punto $\boldsymbol{p}_i$ al plano se define como $d_i = (\boldsymbol{p}_i - \boldsymbol{x}) \cdot \vec{\boldsymbol{n}}$.

Tomando $\boldsymbol{x} = \overline{\boldsymbol{p}} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{p}_i$ como el centroide de los k vecinos de $\boldsymbol{p}$, la solución para $\vec{\boldsymbol{n}}$ se obtiene analizando los autovectores y autovalores de una matríz de covarianza C creada a partir de los vecinos del punto, definida como

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \xi_i \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ \mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j},\ j \in \{0, 1, 2\}$$

donde $p_1, \ldots, p_k$ son los vecinos de p, λ_j es el j-ésimo autovalores de la matríz, $\vec{v_j}$ es el j-ésimo autovector, y ξ_i es un peso posible para p_i, en general igual a 1.

Si los autovalores son positivos, el primer autovector $\vec{v_0}$ es la aproximación de $\vec{n}$ o de $-\vec{n}$, es decir que no se sabe el signo de la normal. Éste no puede obtenerse de forma no ambigüa a partir del análisis de la matríz de covariaza [Rus10]. Para solucionar este problema, las normales se orientan hacia un mismo punto de vista, el $(0, 0, 0)$, o uno particular seleccionado. Para orientar las normales consistentemente hacia el punto de vista $\boldsymbol{v}_p$, es necesario satisfacer la restricción $\vec{n}_i \cdot (v_p - p_i) > 0$ para todos los puntos $\vec{p}_i$ y sus respectivas normales $\vec{n}_i$.

Para mallas poligonales, la estimación de normales es más simple porque los caras dan una aproximación de la superficie, haciendo que la obtención del plano tangente sea más sencilla. Además, la orientación de las normales ya viene dada por la configuración de la malla.

2.5.3. Transformación Rígida

Dado un objeto, si queremos describir su movimiento en el espacio, en principio debemos especificar la trayectoria de todos los puntos en él, dando las coordenadas de los puntos en función del tiempo. Si el objeto es rígido, en realidad no es necesario describir el movimiento de todos los puntos, sino que basta con especificar cómo se modifica solo uno de ellos y el sistema de coordenadas asociado a él.

La razón es que para los objetos rígidos, la distancia entre dos puntos sobre el objeto no cambia con el tiempo. Si $p(t)$ y $q(t)$ son las coordenadas en función del tiempo de dos puntos p y q, la distancia entre ellos se mantiene constante, es decir que $||p(t) - q(t)|| = d \quad \forall t \in \mathcal{R}$. Puede determinarse una transformación T que representa el movimiento del objeto rígido, siendo T entonces una transformación rígida, que preserva distancias, y los productos vectoriales e internos [Ma04].

El movimiento general de un objeto rígido en tres dimensiones se puede caracterizar con una rotación y una traslación. En la Figura 2.9 se muestra el movimiento de un objeto

rígido con un sistema de coordenadas C adjunto, respecto al sistema de coordenadas del mundo W.

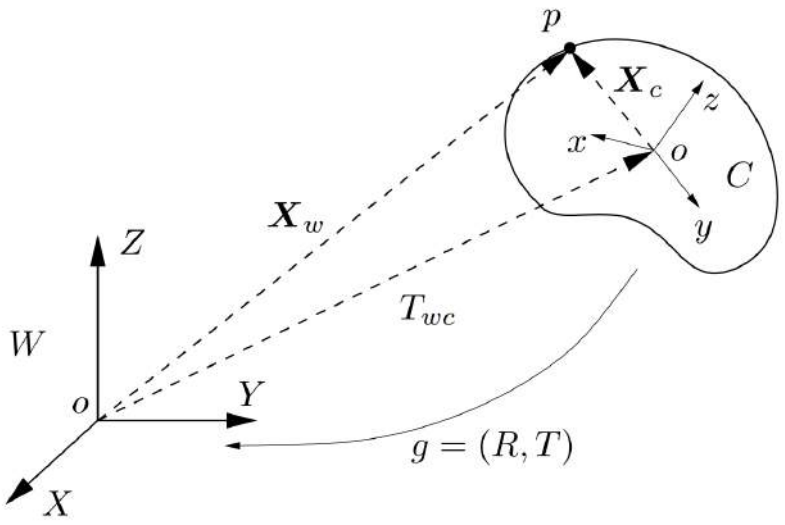


FIGURA 2.9: Ejemplo de movimiento rígido entre el marco de referencia del objeto, C, y del mundo, W. Imagen obtenida del libro de Yi-Ma [Ma04].

Las coordenadas de un punto p en el objeto respecto a W, se representan con el vector X_w, que es la suma de la traslación $T_{wc} \in \mathcal{R}^{3x1}$ del origen del sistema de coordenadas C relativo al centro de W, y el vector X_c, pero expresado relativo a W. Dado que este vector representa las coordenadas de p relativas al sistema de coordenadas C, X_c con respecto al sistema de coordenadas del mundo W se vuelve $R_{wc}X_c$, donde $R_{wc} \in \mathcal{R}^{3x3}$ es la rotación relativa entre los dos sistemas de coordenadas. Por lo tanto, las coordenadas de X_w están dadas por $X_w = R_{wc}X_c + T_{wc}$. El par (R, T) caracteriza a esta transformación, que no es solo una descripción de la configuración del objeto rígido, denominada la **pose**, sino también una transformación de coordenadas entre los dos sistemas de coordenadas.

Esta transformación se suele expresar en forma matricial de la siguiente manera:

$$T \in \mathcal{R}^{4x4} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

donde $R \in \mathcal{R}^{3x3}$ es la matríz de rotación, y $t \in \mathcal{R}^{3x1}$ es el vector de traslación.

Capítulo 3

Descriptores

En este capítulo se explican los descriptores utilizados, y se discuten los resultados de evaluaciones del poder discriminativo de cada uno.

3.1. Introducción

Los descriptores son representaciones de la información de una imagen, generalmente en forma vectorial, y son una parte fundamental de un sistema de reconocimiento de objetos. Para reconocer un objeto en una escena, podemos seguir los siguientes pasos: seleccionar puntos a describir, elaborar descriptores y luego compararlos para obtener asociaciones entre el modelo y la escena que permitan determinar si el objeto está presente y dónde. El concepto de sistema de reconocimiento será descripto más detalladamente en el Capítulo 4.

Para la elaboración de un descriptor, se utiliza información de su entorno, para tener datos de la superficie. Esta información proviene de los puntos en la vecindad (Subsección 2.5.1) del descriptor, y en general también de sus normales, formando el soporte del descriptor. La densidad (Sección 2.5.1) de la nube afecta la cantidad de información capturada en un soporte particular. Si la nube es densa entonces los descriptores serán muy informativos, y si es muy esparsa entonces ya no porque no existe la suficiente cantidad de puntos para otorgar una buena estimación de la superficie que rodea a cada punto.

Los descriptores pueden ser *locales* o *globales* (por ejemplo VFH [RBTH10] y GFPFH [RHBB09]) al objeto. Los descriptores locales sirven para distinguir entre puntos particulares, y suelen contener información de la superficie en un entorno del punto, y los

globales para distinguir entre segmentos. En un enfoque que utiliza descriptores globales es necesario contar con modelos de los objetos a identificar, por lo cual es necesario segmentar esos modelos con algún método.

El uso de descriptores locales es más apropiado para el reconocimiento de objetos en presencia de fondos con ruido, y datos extra y faltantes; ya que tienen resistencia a estos inconvenientes. Por otro lado, el uso de descriptores globales suele proveer un mejor desempeño, si no se tienen los inconvenientes mencionados.

A lo largo de este capítulo, para cada descriptor, explicaremos cómo se elabora, y cómo proponen obtener puntos y comparar descriptores según la publicación original de dicho descriptor.

3.2. SHOT

El descriptor SHOT [TSDS10] fue específicamente desarrollado para encontrar correspondencias de forma rápida y ser resistente al clutter y al ruido. Este descriptor codifica histogramas de las normales de los puntos en la vecindad del punto a describir, que son más representativas de la estructura local de la superficie que simplemente las coordenadas 3D. El uso de histogramas trae un efecto de filtrado requerido para lograr robustez frente al ruido.

En la elaboración del descriptor se introduce una forma de obtener un marco de referencia único, repetible y no ambiguo, usando descomposición en autovalores alrededor del punto, que hace que el descriptor sea invariante a rotación. El poder discriminativo del descriptor se aumenta introduciendo información geométrica sobre la posición de los puntos dentro del soporte.

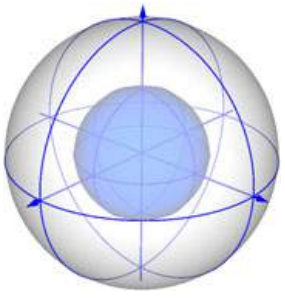


FIGURA 3.1: Soporte esférico utilizado en la elaboración del descriptor SHOT. Imagen adquirida del paper de Tombari, Salti y Stefano [TSDS10].

Para elaborar el descriptor, primero se computa un conjunto de histogramas locales sobre volúmenes 3D dentro de una esfera centrada en el punto, dicha esfera puede verse en la Figura 3.1. Luego se agrupan todos los histogramas locales para formar el descriptor. Para cada uno de los histogramas locales, se acumula la cantidad de puntos según el coseno del ángulo θ_i entre la normal $\boldsymbol{n}_i$ de cada punto q en la sección correspondiente de la grilla y la normal $\boldsymbol{n}_p$ del punto para el cual se realiza el descriptor. El descriptor finalmente se normaliza para que sume 1, para hacerlo más resistente a distintas densidades de puntos.

También existe una versión de este descriptor, llamada **CSHOT** que incluye información de color [TSDS11], haciendolo aún más discriminativo, y permitiendo mejorar la eficiencia de la búsqueda de correpondencias. Para la información de color, se acumulan las diferencias usando norma $\mathcal{L}_1$ (Ecuación 2.2) entre los vectores de intensidad de los puntos, en algún espacio de color, como por ejemplo RGB. Se analizan distintas posibilidades para este espacio de color, eligiendo CIELab por su propiedad de ser más perceptualmente uniforme que RGB [Fai13].

En el artículo original [TSDS10], los puntos se eligen submuestreando aleatoriamente (ver Subsección 4.2.1), y se comparan los descriptores usando norma 2 (ver Ecuación 2.3).

Implementación de PCL

SHOT se representa con dos vectores de valores de punto flotante, uno de 352 posiciones representando los histogramas agrupados, y uno de 9 posiciones representando un marco de referencia, donde se utilizan 3 posiciones por cada eje de coordenadas. En el caso de CSHOT el primer vector es de 1344 posiciones, por la incorporación de la información de color.

Como parámetros se puede elegir el radio del descriptor, y el radio usado para estimar los marcos de referencia. Si no se elige el radio de los marcos de referencia, su valor es el mismo que el radio del descriptor, que es lo que ocurre en el sistema implementado (Sección 4.3).

3.3. FPFH

El descriptor FPFH es una simplificación de PFH, que es preciso pero computacionalmente costoso. Por este motivo, explicaremos primero el descriptor PFH para mejorar la

comprensión de FPFH.

PFH (Persistent Feature Histogram) [RBMB08] tiene como objetivo capturar la geometría alrededor del punto a describir analizando la diferencia entre las direcciones de las normales en la vecindad del punto de interés. Una estimación de normales imprecisa puede producir descriptores de mala calidad. Primero se consideran todos los puntos en una vecindad de radio r, no solo el punto con sus vecinos sino los vecinos entre sí. En la Figura 3.2 se muestra la región de influencia en el cómputo de PFH para un punto p_q marcado en rojo y ubicado en el medio de una esfera de radio r en la cual se encuentran sus k vecinos. Para cada par se realiza un histograma con la relación entre ellos, haciendo que la complejidad de cómputo sea de $O(k^2)$.

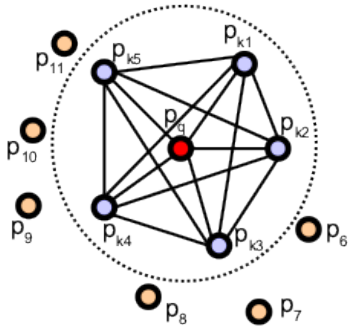


FIGURA 3.2: Diagrama de influencia para la elaboración del descriptor para el punto p_q, ubicado en el centro. Se consideran todos los pares de puntos en un radio r.

Para todos los pares de puntos p, q en la vecindad, se construye un marco de referencia fijo que consiste en tres vectores unitarios (u, v, w), donde

$$\boldsymbol{u} = \boldsymbol{n}_p \qquad \boldsymbol{v} = \boldsymbol{u} \times \frac{\boldsymbol{q} - \boldsymbol{p}}{\| \boldsymbol{q} - \boldsymbol{p} \|_2} \qquad \boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v}$$

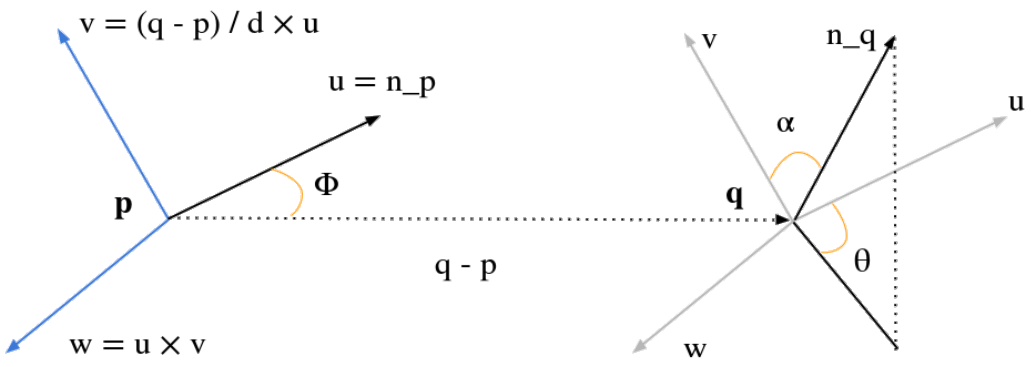


FIGURA 3.3: Marco de referencia para el par de puntos p, q, en el cual se definen los ángulos α, ϕ y θ para representar la diferencia entre las normales $\boldsymbol{n}_p$ y $\boldsymbol{n}_q$

Usando este marco de referencia, la diferencia entre las normales en $p(\boldsymbol{n}_p)$ y en $q(\boldsymbol{n}_q)$ puede ser representadas por tres ángulos (α, ϕ, θ) de la siguiente forma

$$\alpha = (\boldsymbol{v} \cdot \boldsymbol{n}_q) \qquad \phi = (\boldsymbol{u} \cdot \frac{q - p}{d}) \qquad \theta = \arctan(\boldsymbol{w} \cdot \boldsymbol{n}_q, \boldsymbol{u} \cdot \boldsymbol{n}_q)$$
$$\text{donde } d = \| q - p \|_2$$

El marco de referencia y los ángulos definidos se pueden ver en un ejemplo en la imagen 3.3. Estos tres ángulos (α, ϕ, θ) junto con d se acumulan para todos los pares de puntos considerados, usando un histograma de 125 posiciones. El descriptor final es la concatenación de los histogramas de cada variable.

FPFH (Fast PFH) [RBB09] [HRBB09] reduce la complejidad computacional de la obtención del descriptor de $O(k^2)$ a $O(k)$ para un punto, donde k es la cantidad de vecinos considerados. Se consideran solo las relaciones directas entre un punto y sus k vecinos, sin las relaciones adicionales entre los vecinos. Los histogramas que acumulan los ángulos (α, ϕ, θ) se generan de la misma manera que para PFH, sin tener en cuenta la distancia d. Este paso produce un histograma intermedio llamado SPFH (Simplified PFH).

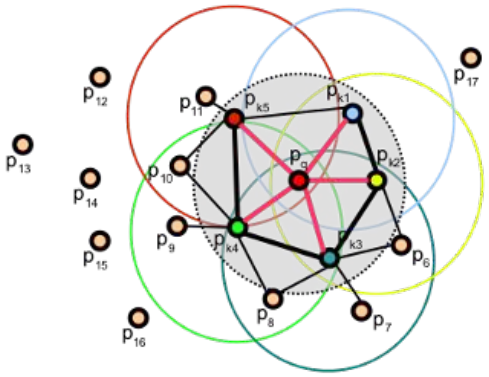


FIGURA 3.4: Diagrama de influencia para la elaboración del descriptor para el punto p_q, ubicado en el centro. Se consideran solo los vecinos dentro de un radio r, pero al tener en cuenta los histogramas de estos, se obtiene información de la superficie de hasta $2r$ de distancia.

Para compensar esta pérdida de información proveniente de las conexiones entre los vecinos, se toma un paso extra después de calcular los histogramas SPFH. Se suma el SPFH del punto con un promedio pesado del de sus vecinos, usando como peso generalmente la distancia. Esto hace que se incorpore información de puntos que están a una distancia hasta dos veces la seleccionada, $2r$, como puede verse en la imagen 3.4.

De forma más específica la forma de calcular el descriptor FPFH para un punto $\boldsymbol{p}$ es la siguiente

$$\mathsf{FPFH}(\boldsymbol{p}) = \mathsf{SPFH}(\boldsymbol{p}) + \frac{1}{k}\sum_{i=1}^{k}\frac{\mathsf{SPFH}(\boldsymbol{p}_i)}{||\boldsymbol{p}_i - \boldsymbol{p}||_2}$$

Los tres ángulos (α, ϕ, θ) se acumulan en tres histogramas de 11 posiciones cada uno. Por último se concatenan para formar el descriptor final del punto en cuestión.

En el artículo original [RBB09], las asociaciones entre descriptores se encuentran usando SAC-IA, un método muy similar a RANSAC (ver Sección 4.2.3), y los descriptores se elaboran para todos los puntos de la nube.

Implementación de PCL

FPFH se representa con dos vectores de valores de punto flotante representando los histogramas agrupados. Como parámetros se puede elegir el radio del descriptor, y la cantidad de intervalos utilizados para acumular los ángulos. En el sistema implementado en esta tesis (ver Sección 4.3) no se modifican estas cantidades y se usa el valor predeterminado de 11 intervalos por ángulo, resultando en un vector final de 33 posiciones (la longitud del vector cambiará si la cantidad de intervalos se modifica).

Nota: Las imágenes de los diagramas de influencia presentados en esta sección fueron adquiridas del paper de Rusu, Blodow y Beetz [RBB09].

3.4. Spin Images

El descriptor *Spin-Image* [Joh97] [JH99] es local, invariante a transformaciones rígidas y capáz de manejar el clutter y la oclusión. Surge como una posible solución al problema de comparar superficies, se consideran los descriptores de spin image, o las spin images, de puntos particulares, y si existen muchas correspondencias entre puntos entonces se considera que las superficies se corresponden. Este enfoque parte el problema de correspondencia de superficies en muchos subproblemas más chicos.

En el sistema propuesto, la forma de la superficie se describe mediante una colección de puntos 3D y sus normales. Además a cada punto se le asocia una imagen descriptiva que codifica propiedades globales de la superficie. Esta imagen, es el descriptor *Spin-Image* que consiste en un histograma de dos dimensiones. Se utiliza una malla poligonal para almacenar la información de los puntos y de sus normales, sin embargo, el sistema también funciona para nubes de puntos, ya que pueden guardar estos mismos datos. Se presupone que los vectores de las normales están normalizados, es decir que su norma es igual a 1. En la Figura 3.5 se muestran las componentes del sistema de correspondencias de superficies, la superficie en sí, los puntos tridimensionales con sus respectivas normales, y algunos ejemplos de spin images de puntos de la superficie.

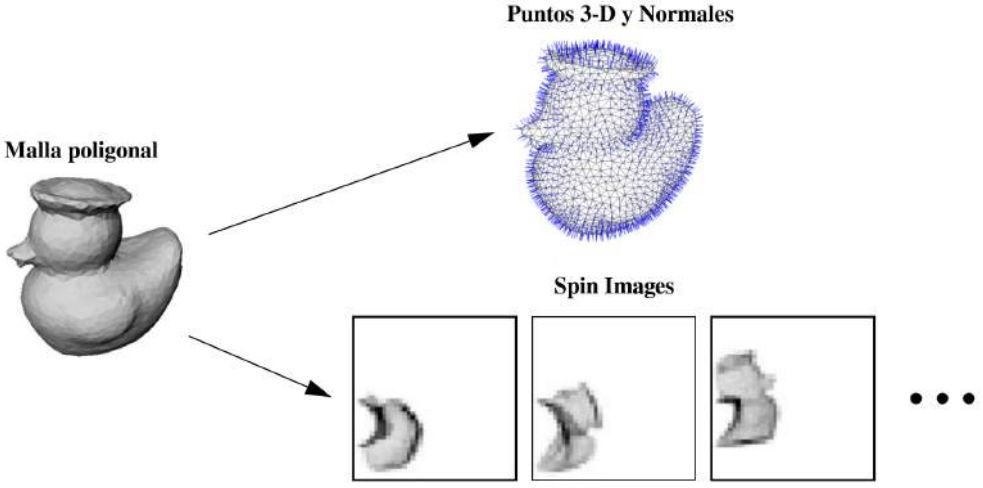


FIGURA 3.5: Componentes del sistema de correspondencias de superficies.

Un concepto importante al trabajar con mallas poligonales es el de *resolución de la malla*, análogo a la densidad de una nube (ver Sección 2.5.1), definida como la mediana entre todas las distancias entre vértices. Es deseable que las mallas a utilizar tengan una resolución uniforme, por lo cual, en el trabajo original presentan un algoritmo [Joh97] para controlar la resolución de la malla, basado en simplificación de mallas poligonales, que agrega y remueve puntos hasta lograr un muestreo uniforme.

Elaboración del descriptor

Un componente principal en la representación de correspondencia de superficies es un punto orientado, que es un punto tridimensional con una dirección asociada. Las coordenadas tridimensionales de un punto $\boldsymbol{p}$ junto con la normal a la superficie $\boldsymbol{n}$ en $\boldsymbol{p}$, forman un *punto orientado* $\boldsymbol{O} = (\boldsymbol{p}, \boldsymbol{n})$.

Para un punto orientado (p, n), pueden definirse dos parámetros que describen al resto de los puntos respecto a él, esto se logra usando el plano $\mathcal{P}$ tangente a la normal que pasa por el punto, y la dirección de la normal $\mathcal{L}$, como puede verse en la Figura 3.6. Un punto x en la superficie se describe en esta base con $\boldsymbol{\alpha}$ que será la distancia a p sobre el plano $\mathcal{P}$, y $\boldsymbol{\beta}$ que será la distancia al plano $\mathcal{P}$ sobre $\mathcal{L}$.

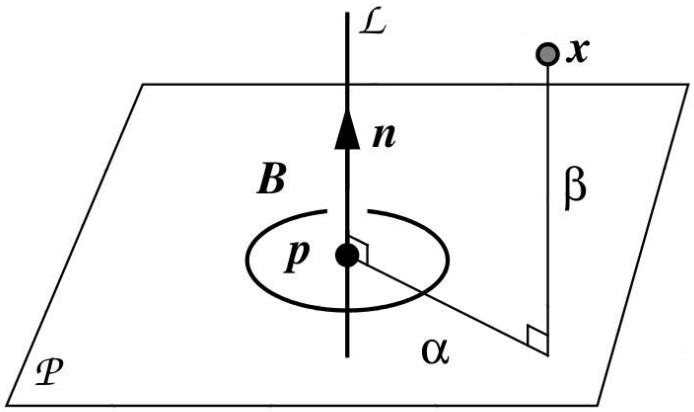


FIGURA 3.6: Base local formada por el punto p, y su normal asociada n. Pueden calcularse dos coordenadas: α, la distancia radial a a la línea normal a la superficie $\mathcal{L}$, y β, la distancia axial sobre el plano tangente $\mathcal{P}$.

Los puntos presentes en un círculo paralelo a $\mathcal{P}$ y centrado en $\mathcal{L}$, tendrán todos las mismas coordenadas (α, β) en esta base, lo que le da al descriptor invarianza a rotación.

En una base completa en 3D cada punto puede ser determinado de forma unívoca. La base formada por el punto orientado determina solamente dos de las tres coordenadas

de las puntos, haciendo que sea incompleta. Si se quisiera extender esta base para que fuera completa, se deberían poder determinar los tres ejes del sistema de coordenadas. Un eje ya está determinado y es la normal, entonces los otros dos ejes restantes deberían estar sobre el plano tangente. Sin embargo, estos posibles ejes son mucho más susceptibles al ruido que la normal.

La base elegida, que es la graficada en la Figura 3.6 resulta estable y sirve para describir la posición de un punto respecto a los otros. Además es un sistema de coordenadas centrado en el objeto que sirve para describir la forma del objeto sin importar la pose.

Para determinar α y β para un punto x, se considera primero el vector y de la diferencia entre los puntos, $y = x - p$. β es la proyección de esta diferencia en la dirección de la normal, que puede obtenerse con el producto interno entre n e y. Para obtener α, se considera el triángulo formado por y, α, y β, donde se cumple que $y^2 = \alpha^2 + \beta^2$, enconces $\alpha = \sqrt{y^2 - \beta^2}$.

Se define entonces, un *mapa spin* s_O para el punto orientado $O = (p, n)$, como la función que proyecta puntos 3D en coordenadas 2D, y permite obtener para un punto x los valores de α y β correspondientes. Cada punto orientado O en la superficie tiene un *mapa spin* s_O único asociado a él.

$$s_O(x) = (\sqrt{||y||^2 - (n \cdot y)^2},\ n \cdot y)) = (\alpha, \beta) \quad \text{donde } y = x - p$$

En la Figura 3.7 se muestran tres casos de aplicación de un mapa spin de un punto a todos los demás de la malla de un modelo de un pato de goma. Aplicar el *mapa spin* s_O a todos los vértices muestreados de la superficie, resulta en una imagen descriptiva pero con mucha sensibilidad al ruido y a variaciones locales en la posición, la representación que se usa es entonces un vector 2D, funcionando como un histograma, que codifica la densidad de los puntos.

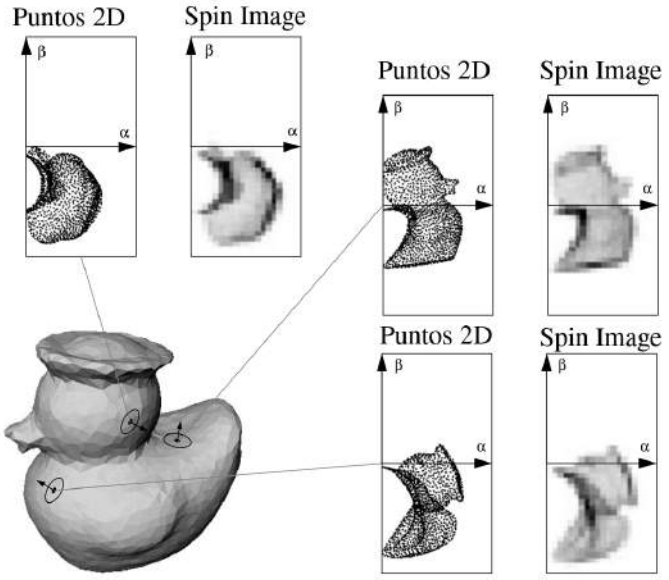


FIGURA 3.7: Ejemplo de spin images para distintos puntos en la superficie.

Acumulación de parámetros

Para un punto orientado O, se consideran todos los otros puntos p_i en la superficie, y para cada uno de ellos se aplica el mapa spin s_O, obteniendo las coordenadas α y β. Para saber si este punto se tiene en cuenta en la generación de la spin imagen del punto O, se considera la distancia a O y el ángulo entre O y la normal de p_i. Si se tiene en cuenta, se busca la posición del punto en el vector 2D y se incrementan las 4 posiciones que lo rodean usando interpolación bilineal, como se muestra en la Figura 3.8. Esta interpolación hace que la contribución de un punto propague su posición en el vector haciendo que sea menos sensible a la posición particular del punto, y por lo tanto más robusto al ruido.

En particular los incrementos para la imagen de spin image I correspondiente a la posicion i, j se realizan como

$$\begin{aligned} I(i,j) &+= (1-a)*(1-b) \\ I(i+1,j) &+= a*(1-b) \\ I(i,j+1) &+= (1-a)*b \\ I(i+1,j+1) &+= a*b \end{aligned} \tag{3.1}$$

donde a y b son los pesos bilineales.

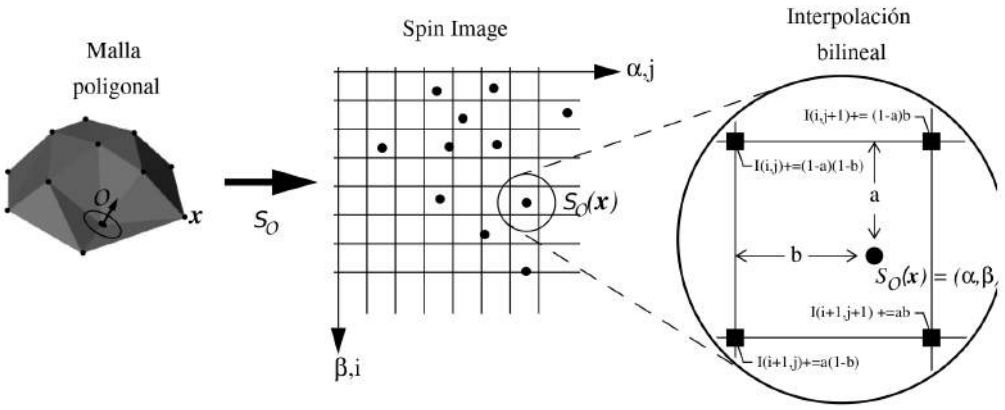


FIGURA 3.8: Incorporación de un punto x a la representación 2D de una spin image para el punto orientado O.

Parámetros de las spin images

Hay tres parámetros que controlan la generación de spin images: tamaño de los intervalos del histograma, ancho de la imagen, y ángulo de soporte.

El tamaño de los intervalos del histograma determina el tamaño de la spin image y el promediado que reduce el efecto de las posiciones individuales. También tiene un efecto es la descriptividad. Se sugiere usar uno o dos veces el valor de la resolución de la malla para suavizar la posición de puntos individuales de manera suficiente describiendo a la vez de manera adecuada la forma.

Las spin images suelen ser cuadradas por lo cual su tamaño puede ser descripto con un único parámetro que es el ancho. La distancia de soporte, el ancho de la imagen por el tamaño de los intervalos, determina cuánta información de la superficie captura una spin image. Para una cantidad fija de intervalos, reducir el ancho de la imagen hace que la descriptividad de una spin image se reduzca dado que la cantidad de información global incluida se reduce, y por otro lado, también disminuye las posibilidades de que el clutter afecte a la spin image.

El ángulo de soporte θ_s determina qué puntos influyen en la generación de las spin images según la dirección de sus normales. Para dos normales n_i, n_j se puede obtener el coseno del ángulo entre ellas muy facilmente mediante el producto escalar, porque se sabe que están normalizadas, por lo tanto $\text{acos}(n_i \cdot n_j)$ da el ángulo entre ellas. Para la spin image de un punto orientado $O = (p, n)$, un punto (x, n_x) contribuye sii $\text{acos}(n, n_x) < \theta_s$. Un ángulo de soporte más chico ayuda a limitar los efectos de auto oclusión y clutter

durante la correspondencia de spin images, pero también disminuye la descriptividad.

Una vez fijados estos parámetros, se pueden determinar los valores de i, j, a y b usados para incorporar la información de un punto x en la elaboración de una spin image en un punto orientado O (Figura 3.8). i, j son las coordenadas del intervalo de la spin image, y a, b son los pesos bilineales usados para incrementar los intervalos que rodean a i, j.

$$\begin{aligned} i &= \left\lfloor \frac{\text{image\_width}/2 - \beta}{\text{bin\_size}} \right\rfloor & j &= \left\lfloor \frac{\alpha}{\text{bin\_size}} \right\rfloor \\ a &= \alpha - j * \text{image\_width} & b &= \beta - i * \text{image\_width} \end{aligned} \tag{3.2}$$

donde α y β son las coordenadas del punto x en la base de O, image_width es el ancho de la imagen, y bin_size es el tamaño de los intervalos.

En la tesis original [Joh97], se describen todos los puntos de las superficies, y luego se comparan las *Spin-Images* usando un coeficiente de correlación, como se explica a continuación.

Comparación de spin images

La forma de comparar spin images que se propone es el coeficiente de correlación lineal normalizado. Dadas dos spin images P y Q con n intervalos cada una, el coeficiente de correlación es

$$\mathcal{R}(P,Q) = \frac{n \sum p_i q_i - \sum p_i \sum q_i}{\sqrt{(n \sum {p_i}^2 - (\sum p_i)^2)(n \sum {q_i}^2 - (\sum q_i)^2)}} \tag{3.3}$$

$\mathcal{R}$ está entre -1 (anti correlacionado) y 1 (completamente correlacionado) y mide el error normalizado usando la distancia entre los datos y el mejor ajuste lineal a los datos usando cuadrados mínimos. Si $\mathcal{R}$ es alto, las imágenes de dos *Spin-Images* serán similares, y si es bajo, no.

Implementación de PCL

La representación de spin images es un vector de valores de punto flotante de 153 posiciones. Como parámetros se puede elegir el radio del descriptor como en todos los

otros descriptores, el coseno del ángulo entre las normales, que es el ángulo de soporte θ_s que se nombró previamente, y el ancho de las imágenes usadas. El tamaño de los intervalos no se puede elegir, su valor depende del radio del descriptor y el ancho de la imagen. Además se puede seleccionar la cantidad mínima de puntos que contribuyen en la elaboración de una spin image.

En el sistema implementado el coseno se fija en 0.5, dando un ángulo de 60 grados, para el ancho de las imágenes se utiliza el valor predeterminado de 8, y la cantidad mínima de puntos es 1.

Las imágenes que representan a las spin images son rectangulares, y no cuadradas como sugiere la tesis de Johnson [Joh97], su tamaño predeterminado es de 8 x 16, esto es una imagen el doble de alto que de ancho, más una fila y una columna extra para tener en cuenta la interpolación usada en la acumulación de parámetros, resultando en un tamaño de 9 x 17.

El tamaño de los intervalos se calcula como search_radius/image_width/$\sqrt{2}$, donde search_radius es el radio del descriptor, e image_width es el ancho de la imagen, que es igual a 8 en el sistema implementado.

Los parámetros (α, β) se calculan de la siguiente manera:

$$\alpha = ||y|| * \sqrt{1{,}0 - \frac{(n \cdot y)^2}{||y||^2}} \qquad \beta = |n \cdot y| \qquad \text{donde } y = x - p$$

siendo β igual que en la tesis de Johnson [Joh97] pero considerado siempre positivo, y α equivalente porque

$$\begin{aligned}
\alpha &= ||y|| * \sqrt{\left(\frac{1{,}0}{||y||^2}\right)\left(||y||^2 - (n \cdot y)^2\right)} \\
&= ||y|| * \sqrt{\frac{1{,}0}{||y||^2}} * \sqrt{||y||^2 - (n \cdot y)^2} \\
&= ||y|| * \frac{1{,}0}{||y||} * \sqrt{||y||^2 - (n \cdot y)^2} \quad \text{porque } ||y|| \geq 0 \\
&= \sqrt{||y||^2 - (n \cdot y)^2}
\end{aligned}$$

Los valores de i, j, a y b, se obtienen de la siguiente manera

$$i = \left\lfloor \frac{\beta}{\text{bin\_size}} \right\rfloor + \text{image\_width} \qquad j = \left\lfloor \frac{\alpha}{\text{bin\_size}} \right\rfloor$$

$$a = \frac{\alpha}{\text{bin\_size}} - j \qquad b = \frac{\beta}{\text{bin\_size}} + \text{image\_width} - i$$

i se calcula de esta manera y no como en la fórmula 3.2 porque el ángulo β siempre es positivo, j queda igual y a y b se calculan de forma equivalente.

El incremento con interpolación bilineal se hace de la misma manera que la mostrada en las fórmulas 3.1.

Nota: Las imágenes usadas en este capítulo fueron adquiridas de la tesis de Johnson [Joh97].

3.5. Evaluación de Descriptores

En un sistema de reconocimiento de objetos, las *correspondencias* (ver Subsección 4.2.3) son las asociaciones que se dan entre descriptores, que se usan para lograr determinar la presencia del objeto. Para establecer correspondencias, es fundamental contar con un método de evaluación de la calidad de las asociaciones entre pares de puntos. Este método suele basarse fuertemente en la medida de distancia elegida para comparar descriptores. En una situación ideal, ocurre que las distancias entre descriptores de puntos que se deberían corresponder son más chicas que las distancias entre descriptores de puntos que no se deberían corresponder. Esta diferencia puede no ser tan clara al utilizar datos reales.

Llamaremos *correspondencias verdaderas* a aquellas asociaciones entre descriptores de puntos que deberían corresponderse, y *correspondencias falsas* a las que se dan entre descriptores de puntos que no deberían corresponderse. Idealmente existe cierto valor d de distancia que sirve para separar el grupo de correspondencias verdaderas del de correspondencias falsas. Los conjuntos de pares de puntos cuyos descriptores estén a una distancia menor a d serán considerados correspondencias. Puede pasar que algunas correspondencias falsas queden en este conjunto, generando *falsos positivos*, también puede ocurrir que algunas correspondencias reales queden fuera de este conjunto, ocasionando *falsos negativos*.

Queremos evaluar la calidad de los descriptores con distintas medidas de distancia asociadas, para esto realizamos dos experimentos con distintas medidas entre descriptores, uno con distancias entre puntos más cercanos, explicado en la Subsección 3.5.2, y otro con cocientes entre los primeros y segundos puntos más cercanos, explicado en la Subsección 3.5.3. Para cada descriptor y medida de distancia, consideramos dos cuadros

entre los cuales sabemos la transformación que los relaciona, y obtenemos conjuntos de puntos que van a conformar las correspondencias verdaderas y las falsas. Luego medimos las distancias, o los cocientes según el experimento, entre los descriptores de los puntos de los conjuntos y finalmente graficamos su frecuencia.

Las distancias consideradas entre los vectores numéricos p y q que representan los descriptores de dos puntos, son $\mathcal{L}_1$, $\mathcal{L}_2$ y χ^2 definidas en la Subsección 2.5.1. Para spin images además consideramos la distancia de correlación definida como $-R(p,q)+1$ donde R es el coeficiente de correlación (Ecuación 3.3).

Base utilizada

La base de datos que utilizamos para los experimentos es `freiburg1_xyz`, parte del repositorio RGB-D SLAM [Mü] [SEE+12]. Esta base contiene cuadros de un escritorio de oficina con una gran variedad de objetos: monitores, teclados, libros, sillas, tazas, como puede observarse en la Figura 3.9. Cuadro a cuadro el movimiento de la cámara para esta base es muy chico, y la rotación es casi inexistente.

FIGURA 3.9: Imágenes de la base de datos `freiburg1_xyz`.

La base contiene imágenes de color y de profundidad. Como ground truth se provee la pose (ver Subsección 2.5.3) de la cámara para cada timestamp.

Por cada timestamp, la información provista es de la forma $t_x\ t_y\ t_z\ q_x\ q_y\ q_z\ q_w$, donde

- $t = (t_x, t_y, t_z)$ representa en forma vectorial la posición del centro óptico de la cámara respecto al origen del mundo.
- $q = (q_x, q_y, q_z, q_w)$ representa en forma de cuaternión la rotación del centro óptico de la cámara respecto al origen del mundo.

Usando los timestamps podemos asociar las imágenes de color con las de profundidad con el ground truth correspondiente para cada par.

3.5.1. Descripción de los experimentos

Consideramos dos nubes de puntos, que se corresponden a dos cuadros F_1 y F_2, y calculamos las distancias para puntos que se corresponden y que no se corresponden. Sabemos cuáles son los puntos que se corresponden usando la información de ground truth.

Transformación que relaciona dos cuadros

Consideramos la información de ground truth l_1 y l_2 de los cuadros, y obtenemos T_1 y T_2, las transformaciones que relacionan el origen del mundo con las posiciones de la cámara en timestamp$_1$ y timestamp$_2$ respectivamente.

A partir de la información de ground truth $l_1 = (t_{1x}\ t_{1y}\ t_{1z}\ q_{1x}\ q_{1y}\ q_{1z}\ q_{1w})$ para el primer timestamp, obtenemos la transformación asociada $T_1 = \left(\begin{smallmatrix} R_1 & t_1 \\ 0 & 1 \end{smallmatrix}\right)$, donde R_1 es la matríz de rotación obtenida a partir del cuaternión $q_1 = (q_{1x}, q_{1y}, q_{1z}, q_{1w})$, y t_1 es el vector de traslación $(t_{1x}, t_{1y}, t_{1z})^T$.

T_2 lo calculamos de manera análoga a partir de l_2, y finalmente, obtenemos la transformación T que relaciona los dos cuadros [1] $T = {T_1}^{-1}\ T_2$.

Búsqueda de Puntos Correspondientes

Dados dos cuadros F_1, F_2, y la transformación T que los relaciona, podemos obtener los puntos correspondientes. Para un punto p perteneciente al primer cuadro F_1, obtenemos el punto que le corresponden, primero transformándolo según T, consiguiendo $\hat{p} = T(p)$, y luego buscando en el segundo cuadro el punto más cercano a $\hat{p}$ (ver Subsección 2.5.1).

El conjunto de correspondencias verdaderas va a estar formado por pares de descriptores de puntos correspondientes. Para las correspondencias falsas consideramos pares aleatorios de descriptores de puntos que no son correspondientes.

Podemos ver un ejemplo de puntos que deberían corresponderse en la Figura 3.10.

[1] https://vision.in.tum.de/data/datasets/rgbd-dataset/tools

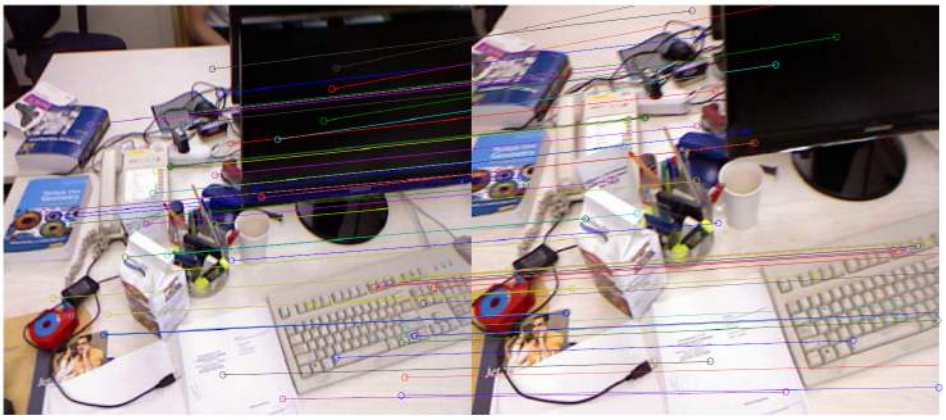


FIGURA 3.10: Ejemplo de pares de cuadros con algunos puntos asociados.

Obtención de datos

1. Tomamos puntos del primer cuadro, submuestreando uniformemente cada 1 cm (ver Subsección 4.2.1).
2. Buscamos los puntos correspondientes en el segundo cuadro, usando la transformación que relaciona dos cuadros.
3. Elaboramos los descriptores (Spin Images, CSHOT o FPFH) de los puntos del primer y segundo cuadro.
4. Obtenemos las distancias (o los cocientes) entre los descriptores de puntos correspondientes.
5. Obtenemos las distancias (o los cocientes) entre los descriptores de puntos no correspondientes.

Generación de Gráficos

Calculamos dos histogramas, uno para las distancias, o cocientes, de las correspondencias verdaderas, y otro para las distancias, o cocientes, de las correspondencias falsas, ambos normalizados para que la suma de los porcentajes de aparición sea 100 %. Luego graficamos los valores de los histogramas, pero como puntos en vez de barras, para facilitar la visualización.

En el eje x se muestra la distancia, o el cociente, según la medida que se esté considerando, y en el eje y el porcentaje de veces (entre 0 y 100) que ese valor de distancia o cociente ocurre.

3.5.2. Evaluación usando la distancia

En este experimento, queremos evaluar para un descriptor particular, cuál es la métrica más efectiva para separar lo más posible las correspondencias buenas de las malas. Queremos encontrar un umbral tal que los pares de puntos con distancia menor a ese umbral sean considerados buenas correspondencias, y aquellos que tengan una distancia mayor sean rechazados. Este umbral será util como un primer paso en la implementación de la búsqueda de correspondencias en el sistema (ver Subsección 4.2.3).

En un caso ideal, como el presentado en la Figura 3.11 la superposición entre las distribuciones de las distancias para las correspondencias verdaderas y las falsas es muy chica, haciendo que la elección de un umbral sea simple. Eligiendo el valor 0.35 como umbral, conseguimos que la mayor parte de las correspondencias buenas (aproximadamente mayor a 90 %) sean consideradas, teniendo un porcentaje muy bajo (aproximadamente menor a 10 %) de correspondencias falsas con distancia menor al umbral. Esto significa que utilizar este umbral para decidir si dos puntos se deberían corresponder es una muy buena opción.

Otras opciones para el umbral son: 0.25 para dejar afuera todas las correspondencias malas a cambio de mantener menos correspondencias buenas, y 0.45 que acepta todas las correspondencias buenas pero incorpora más correspondencias malas.

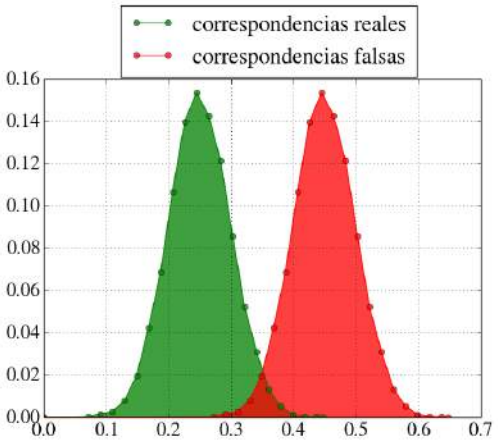


FIGURA 3.11: Distribución de distancias para correspondencias verdaderas y falsas en un caso ideal, la superposición entre las distribuciones es muy chica.

Análisis de las distribuciones de distancia obtenidas

Spin Images

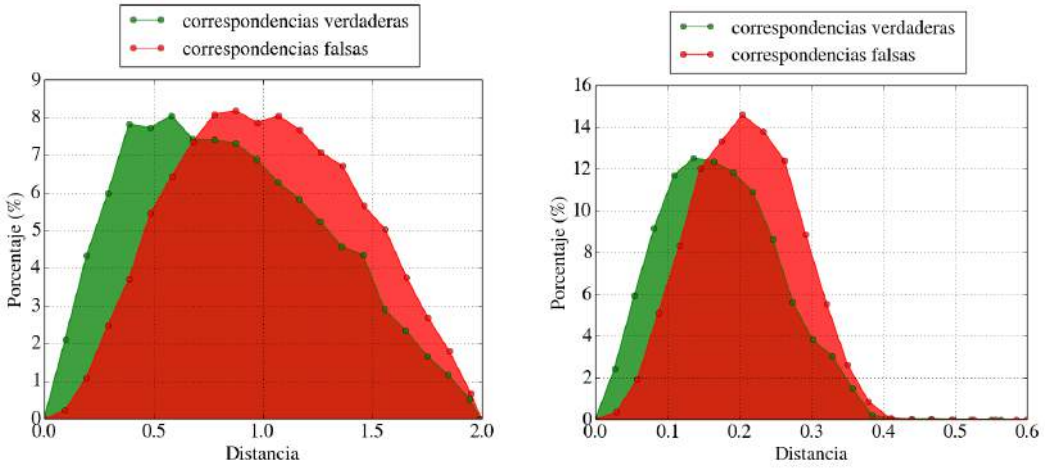
En el caso de Spin Images (Figuras 3.12 y 3.13), con las distancias $\mathcal{L}_1$, $\mathcal{L}_2$ y χ^2, podemos determinar umbrales que hacen que la cantidad de correspondencias falsas sea controlable, para que no sea tan grande, a cambio de dejar una gran cantidad de correspondencias buenas afuera.

Para $\mathcal{L}_1$, un valor de 0.7 o 0.8 cumple este propósito, dejando más de la mitad de las correspondencias verdaderas falsas y verdaderas fuera, pero conservando más correspondencias verdaderas que falsas, ya que se ve en la Figura 3.12A que la curva verde está por encima de la curva roja para las distancias menores al umbral.

Para $\mathcal{L}_2$, un valor de 0.15 cumple esta función por razones análogas como puede observarse en la Figura 3.12B.

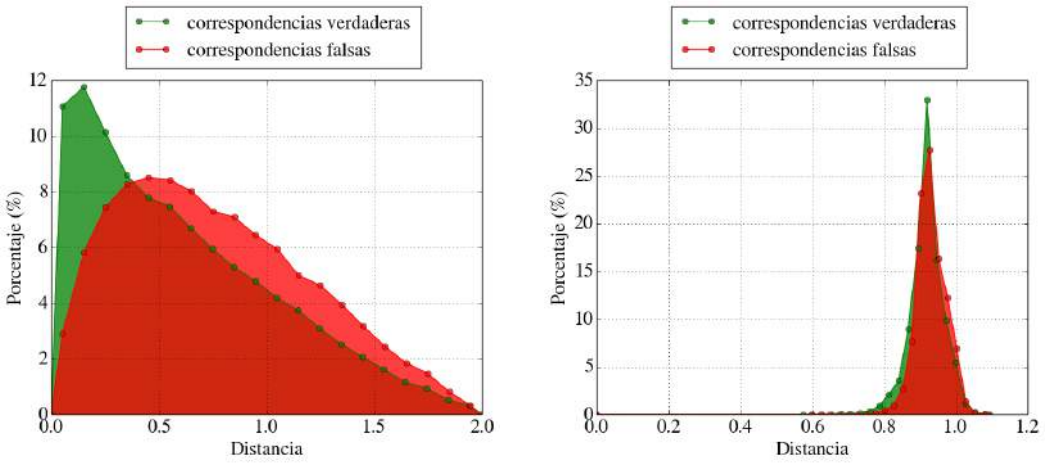
Para χ^2, un valor de 0.4 sería de utilidad, por las mismas razones que antes, y es interesante observar en la Figura 3.13A la distribución de las correspondencias verdaderas, la frecuencia de las distancias va disminuyendo a medida que aumenta la distancia, lo cual es deseable.

Para la distancia de correlación, este valor no puede determinarse porque la superposición es demasiado grande, como se ve en la Figura 3.13B, haciendo que no sea una buena opción como métrica a utilizar.



(A) (B)

FIGURA 3.12: Spin Images usando distancias $\mathcal{L}_1$ (A) y $\mathcal{L}_2$ (B).



(A) (B)

FIGURA 3.13: Spin Images usando distancia χ^2 (A) y de correlación (B).

CSHOT

En el caso de CSHOT (Figuras 3.14 y 3.15), también obtuvimos un alto porcentaje de superposición, pero podemos determinar un umbral que deje más cantidad de correspondencias verdaderas que falsas, sin dejar muchas correspondencias verdaderas afuera.

Para $\mathcal{L}_i$, podemos usar un valor de 10 para este propósito como puede observarse en la Figura 3.14A, que la curva verde supera a la curva roja para las distancias menores a dicho umbral.

Para $\mathcal{L}_2$, un valor de 1.3 cumple esta función por el mismo motivo que antes. En este caso la norma no es muy buena porque la diferencias entre las cantidades de correspondencias verdaderas y falsas para distancias menores al umbral es muy chica, como puede observarse en la Figura 3.14B.

Para χ^2, un valor de 9 sería de utilidad, por las mismas razones, como se observa en la Figura 3.15.

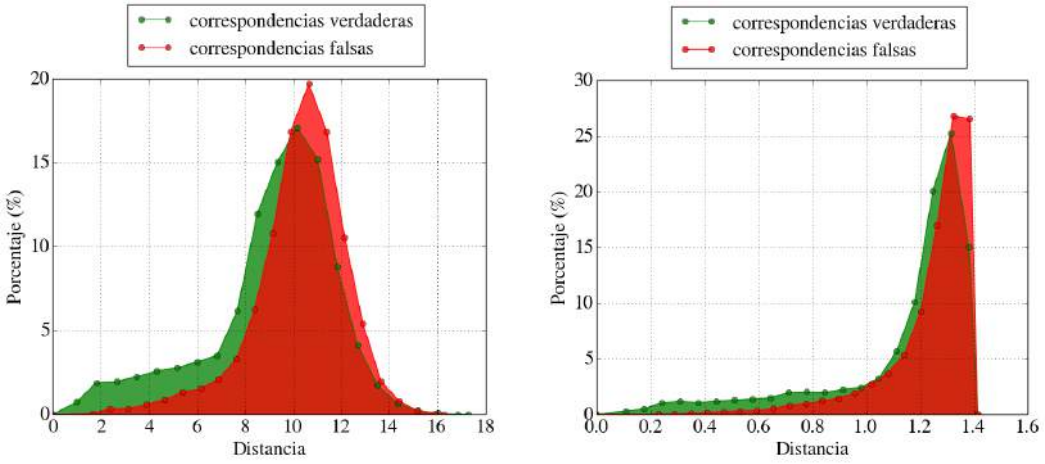


FIGURA 3.14: CSHOT usando distancia $\mathcal{L}_1$ (A) y $\mathcal{L}_2$ (B).

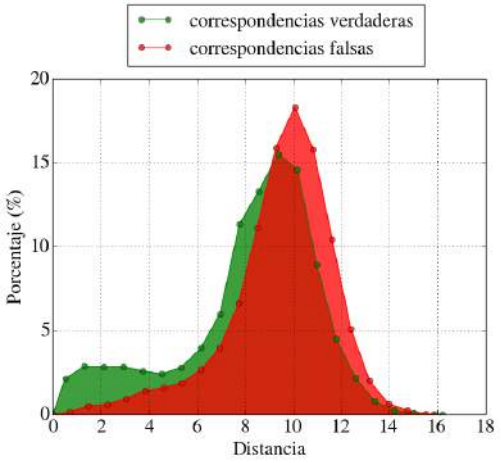


FIGURA 3.15: CSHOT usando distancia χ^2

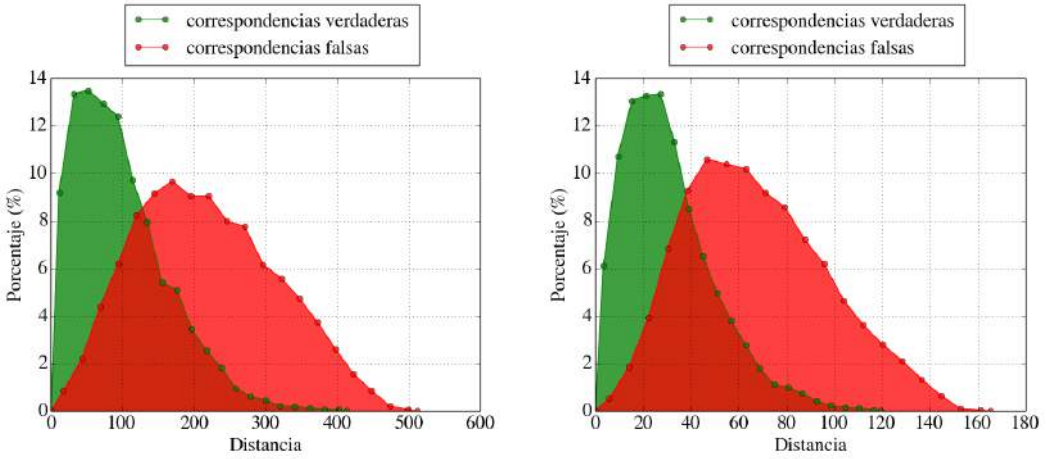
FPFH

En el caso de FPFH (Figuras 3.16 y 3.17), aunque haya superposición, las distribuciones se pueden separar de forma más clara, las distancias para las correspondencias verdadera son más chicas que las de las correspondencias falsas.

Para $\mathcal{L}_1$, podemos usar un valor de 120 como umbral, dejando muchas correspondencias falsas afuera, y rechazando pocas correspondencias buenas, como puede observarse en la Figura 3.16A, que la mayor parte de la curva roja queda fuera de la sección considerada, y además la curva verde supera por mucho a la curva roja para las distancias menor a dicho umbral. También notamos que si la distancia es mayor a 410 ya no hay posibilidad de que sea una buena correspondencia.

Para $\mathcal{L}_2$, un valor de 35 cumple la función de umbral por las mismas razones que antes, como puede observarse en la Figura 3.16B. Al igual que en el caso anterior, si la distancia es mayor a 120 la correspondencia será mala con una probabilidad muy alta.

Para χ^2, un valor de 40 sería de utilidad, ya que captura más de la mitad de las correspondencias verdaderas, y deja afuera más del 80 % de las correspondencias falsas, como se observa en la Figura 3.17.



(A) (B)

FIGURA 3.16: FPFH usando distancia $\mathcal{L}_i$ (A) y $\mathcal{L}_2$ (B).

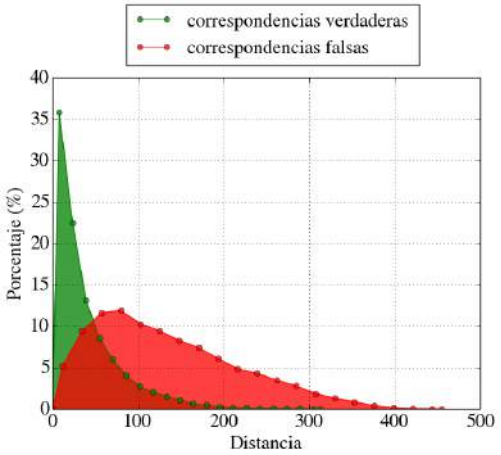


FIGURA 3.17: FPFH usando distancia χ^2

3.5.3. Evaluación usando el cociente

Otra forma de determinar si un punto p debería corresponderse con el punto q, es considerar el cociente entre la distancia de p a q y la distancia de p a $\hat{q}$, donde q es el punto más cercano a p y $\hat{q}$ es el segundo punto más cercano a p, para alguna medida de distancia entre los descriptores de los puntos [Low04]. $\hat{q}$ es el punto más cercano a p después de q. La idea es que este cociente sea más chico para las correspondencias buenas que para las malas.

Para las correspondencias verdaderas el cociente debería ser menor porque tienen que tener el vecino más cercano, el correcto, mucho más cerca que el segundo más cercano, que sería incorrecto. En cambio para las correspondencias falsas, muchos otros puntos que no debería corresponderse están a distancia similar. Si k_0 es la distancia entre p y q, y k_1 es la distancia $\hat{q}$, entonces $k_0 < k_1$ y vale que $0 \leq \frac{k_0}{k_1} < 1$, siendo 0 en el caso en que k_0 sea 0, es decir cuando el vecino más cercano a p sea indistinguible de él (ideal); y 1 cuando $k_0 = k_1$, es decir cuando los dos vecinos más cercanos a p sean indistinguibles entre sí (peor caso).

El experimento realizado es muy parecido al caso de evaluación usando la distancia, solo que en vez de calcular distancias calculamos porcentajes, la base de datos utilizada es la misma, y los puntos que deberían corresponderse se obtienen de la misma forma.

Cálculo de cocientes

Queremos obtener el cociente dado por $\frac{k_0}{k_1}$ para un punto p, donde k_0 es la distancia al punto más cercano a p, y k_1 es la distancia al segundo punto más cercano a p.

Consideramos pares de puntos (p, q) como en el experimento anterior, siendo k_0 la distancia entre ellos. Para k_1 calculamos la distancia de p a todos los keypoints del segundo cuadro, quedándonos con la menor distancia que sea mayor a k_0. Esta sería la distancia al segundo punto más cercano, si consideráramos q como el punto más cercano a p.

Los gráficos también se realizaron de la misma manera, y se muestran a continuación, junto a un análisis de ellos.

Análisis de los distribuciones de cociente obtenidas

Spin Images

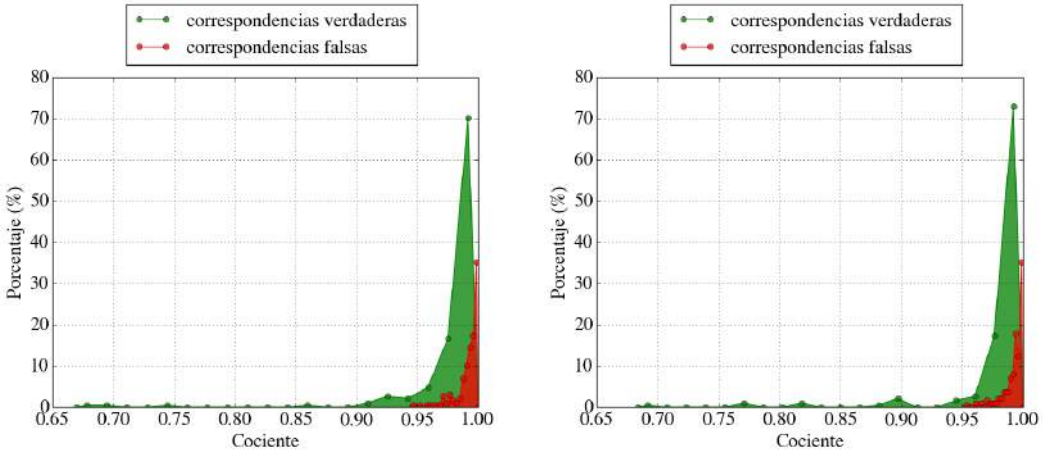
En el caso de Spin Images (Figuras 3.18 y 3.19), con las distancias $\mathcal{L}_1$, $\mathcal{L}_2$ y χ^2, podemos determinar umbrales que hacen que la cantidad de correspondencias falsas sea muy chica, dejando muchas correspondencias buenas afuera.

Para $\mathcal{L}_1$ podemos elegir el umbral de 0.95 o 0.96, que deja pasar pocas cantidades de correspondencias siendo el porcentaje de correspondencias falsas mucho menor que el de las verdaderas, como puede observarse en la Figura 3.18A. También puede elegirse un umbral más alto, como 0.98 para dejar pasar más correspondencias buenas, pero permitiendo más cantidad de correspondencias falsas.

Para $\mathcal{L}_2$ el resultado es muy similar, como se ve en la Figura 3.18B, y podemos usar los mismos umbrales por las razones descriptas anteriormente.

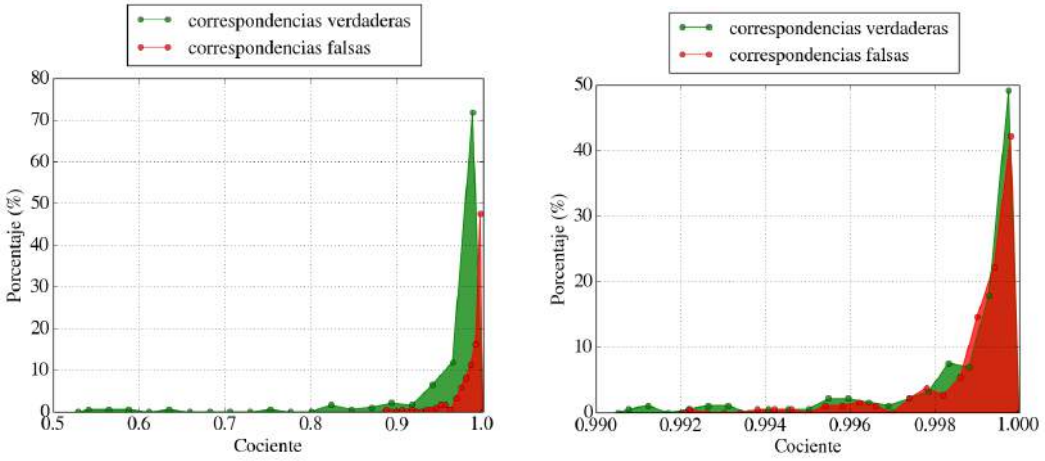
Para χ^2 vemos en la Figura 3.19A que ocurre lo mismo, y nuevamente podemos usar 0.95 como un buen umbral.

Para la distancia de correlación, existe más superposición como se ve reflejado en la Figura 3.19B, aún así podría utilizarse un valor de umbral más alto, como 0.998 para controlar de alguna manera la cantidad de correspondencias falsas. Aún así, esta medida no es buena porque la superposición es demasiado grande.



(A) (B)

FIGURA 3.18: Spin Images usando distancia $\mathcal{L}_1$ (A) y $\mathcal{L}_2$ (B).



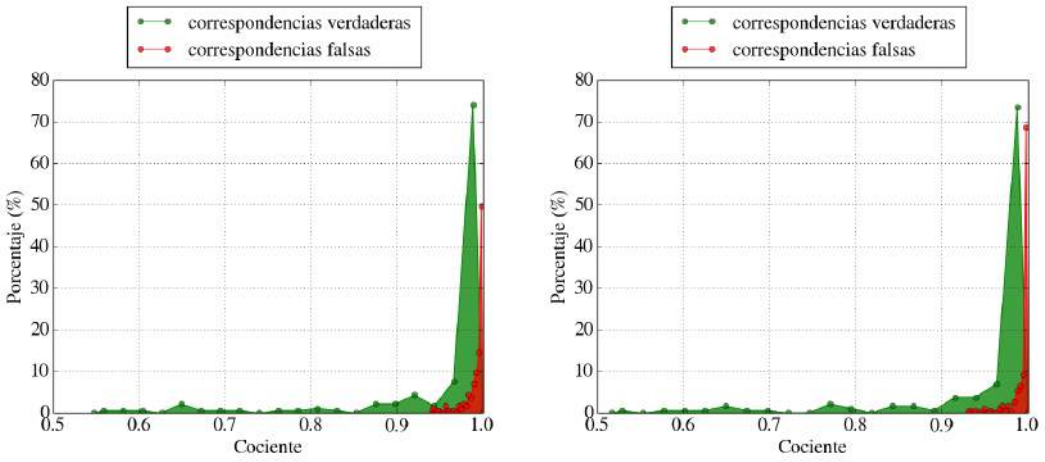
(A) (B)

FIGURA 3.19: Spin Images usando distancia χ^2 ($\blacktriangle$) y correlación (B).

CSHOT

En el caso de CSHOT (Figuras 3.20 y 3.21), los cocientes para las correspondencias falsas toman valores muy cercanos a 1, y se superponen poco con las correspondencias verdaderas. La superposición solamente ocurre para valores muy grandes del cociente.

Para $\mathcal{L}_1$ y $\mathcal{L}_2$ podemos usar un valor de 0.95 como umbral, dejando casi ninguna correspondencia falsa, como puede observarse en las Figuras 3.20A y 3.20B. Para $\mathcal{L}_1$ podrían considerarse valores de umbral más grandes para incorporar más correspondencias verdaderas.



(A) (B)

FIGURA 3.20: CSHOT usando distancia $\mathcal{L}_1$ ($\blacktriangle$) y $\mathcal{L}_2$ (B).

Para χ^2, graficado en la Figura 3.21, un umbral de 0.95 también es util aunque se puede usar un valor más chico como 0.9, para eliminar lo más posible las correspondencias falsas.

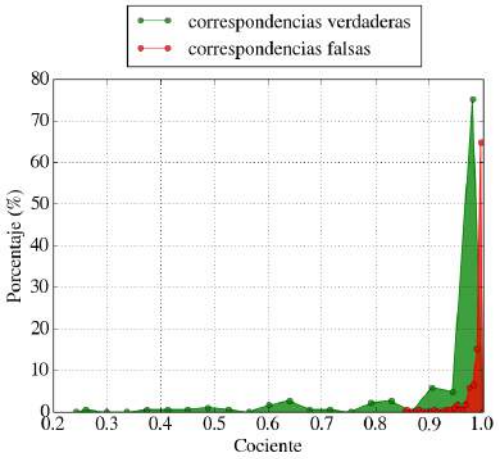
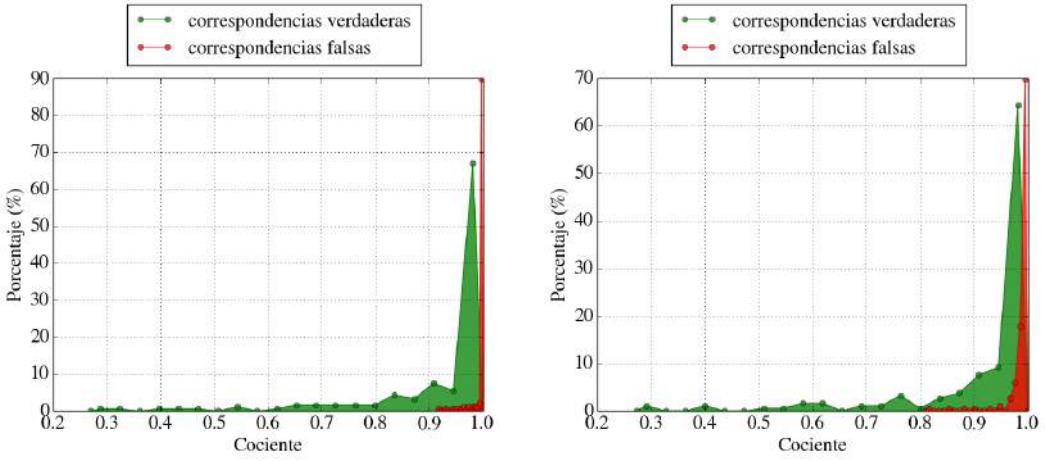


FIGURA 3.21: CSHOT usando distancia χ^2

FPFH

Para FPFH (Figuras 3.22 y 3.23) es aún más marcada la diferencia entre las distribuciones de los cocientes para las correspondencias falsas y verdaderas. Muchas correspondencias falsas toman valores muy cercanos a uno.



(A) (B)

FIGURA 3.22: FPFH usando distancia $\mathcal{L}_1$ (A) y $\mathcal{L}_2$ (B).

Los resultados para todas las distancias son muy similares, como se observa en las Figuras 3.22A, 3.22B y 3.23. Como para los otros descriptores, también podemos usar

un umbral de 0.95 para las distancias $\mathcal{L}_1$, $\mathcal{L}_2$ y χ^2, dejando pocas correspondencias falsas (menos del 10 %) y una buena cantidad de correspondencias buenas (alrededor del 30 %).

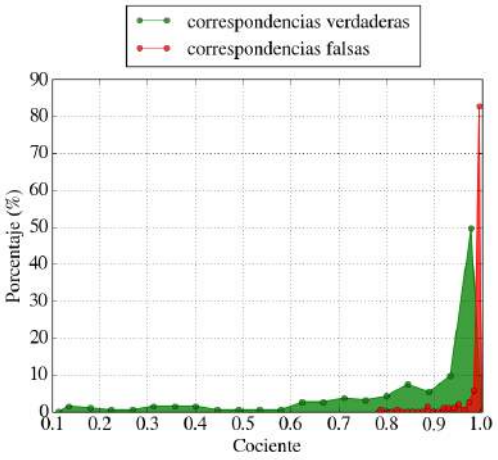


FIGURA 3.23: FPFH usando distancia χ^2

3.5.4. Conclusión

En ambos experimentos encontramos que la superposición entre las distribuciones de correspondencias verdaderas y falsas era inevitable, lo que nos indica que ninguno de los métodos es suficiente para determinar un buen conjunto de correspondencias. El conjunto de pares de puntos asociados obtenido contiene correspondencias falsas, y utilizar directamente ese conjunto como correspondencias finales no es suficiente para lograr un buen reconocimiento, es necesario filtrar de alguna otra manera las correspondencias.

En el sistema implementado, logramos obtener un mejor conjunto de correspondencias incorporando medidas adicionales de rechazo y agrupación de correspondencias, que se explicarán en el capítulo siguiente, en particular en la Sección 4.2.3.

Con el método de la distancia encontramos que en la mayoría de los casos, las distancias $\mathcal{L}_1$ y χ^2 otorgaban mejores resultados que $\mathcal{L}_2$, es decir que las distribuciones de correspondencias verdaderas y falsas resultaban más separadas al usar $\mathcal{L}_1$ y χ^2 que al usar $\mathcal{L}_2$. Para Spin Images usando la distancia de correlación se obtuvieron resultados malos, siempre con demasiada superposición.

Para el método del cociente, vimos que las distribuciones resultaban muy similares para los distintos descriptores y distancias (excepto con correlación para Spin Images), pudiendo usar además el mismo valor umbral (valores entre 0.90 y 0.96), permitiendo

aislarse de la elección del descriptor al momento de realizar la búsqueda de correspondencias. Además en todos los casos se pueden elegir umbrales que dejen a todas las correspondencias falsas afuera, manteniendo pocas correspondencias verdaderas, algo que no ocurre con el método de la distancia.

Al usar esta base, vimos que puede usarse un mismo valor de umbral en todos los casos, pudiendo elegir este umbral entre el rango 0.7 y 0.95. Valores menores a 0.7 ya no sirven porque dejan muy pocas correspondencias, y valores mayores a 0.95 tampoco porque dejan demasiadas correspondencias malas. Al usar otras bases, el umbral debe adaptarse, usando este rango como guía.

Capítulo 4

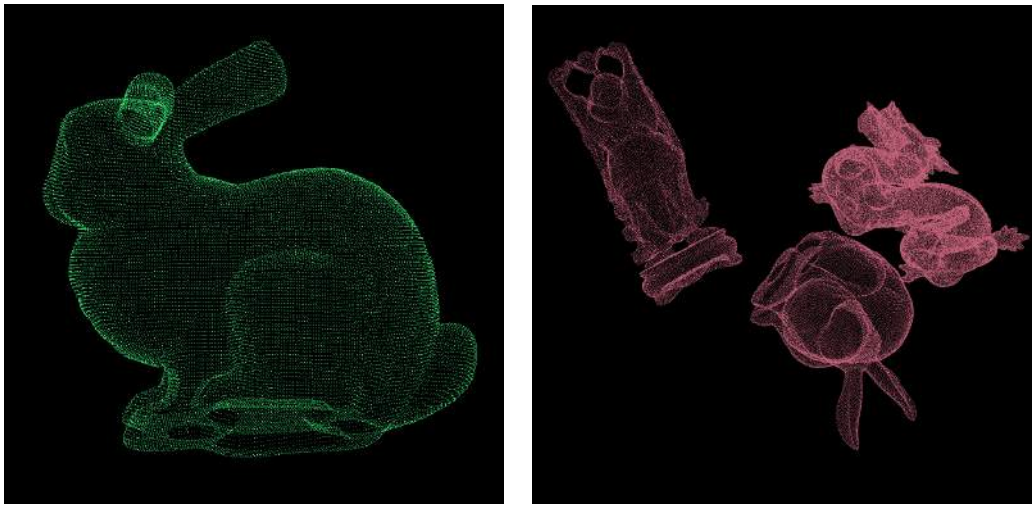
Sistema de Reconocimiento de Objetos

4.1. Introducción

Un sistema de reconocimiento de objetos tiene el propósito de encontrar objetos en imágenes o nubes de puntos. Llamaremos **modelo** a la nube de puntos que contiene el objeto a buscar, y **escena** a la nube en la cual se quiere buscar el objeto.

En la nube modelo, o query, el objeto generalmente es completamente visible, y en la escena pueden haber varios objetos, entre los cuales puede estar el objeto a buscar. Si el objeto está en la escena, es probable que no esté en las mismas condiciones que en la nube modelo, por ejemplo puede estar tapado por otros objetos, rotado, o iluminado de diferente forma.

Por ejemplo, si el sistema de reconocimiento recibe las nubes presentes en la Figura 4.1, debería determinar que el objeto de la nube modelo está presente en la nube escena. Una forma de hacer esto es visualizando la localización del objeto en la escena, como se puede ver en la Figura 4.2, donde en un caso solamente se muestra donde está la instancia del objeto, en amarillo, y en el otro además se ven las asociaciones entre puntos particulares del objeto en la nube modelo y el objeto transformado y llevado a la escena, como líneas rojas. Las nubes no tienen color, por lo cual se les asigna uno para poder visualizarlas.



(A) Modelo (B) Escena

FIGURA 4.1: Ejemplo de nubes provistas al sistema de reconocimiento. La escena contiene tres objetos, entre ellos el buscado, que es el conejo.

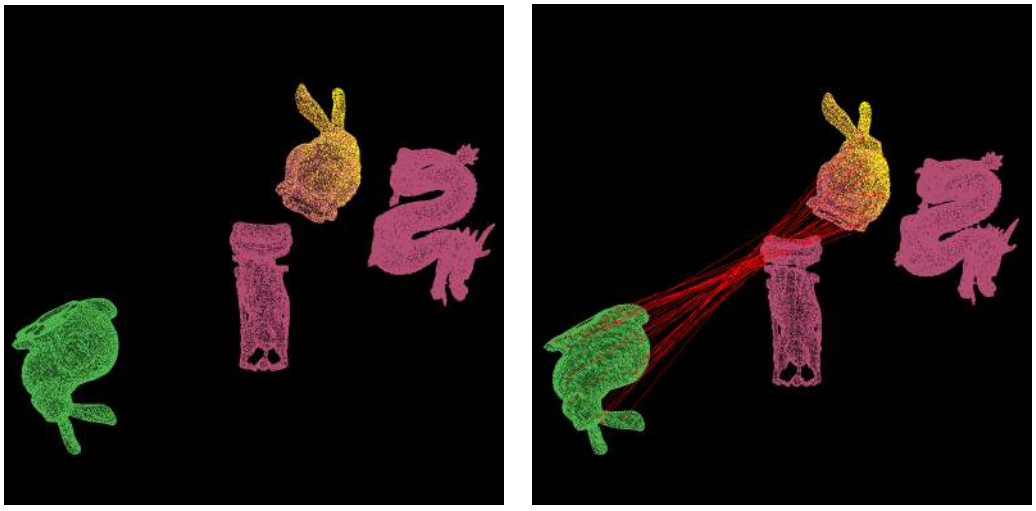


FIGURA 4.2: Ejemplo exitoso de reconocimiento usando las nubes presentadas en la Figura 4.1 en el cual se muestra la instancia del modelo en la escena en amarillo, por sí sola (A), y con correspondencias (B).

Un sistema de reconocimiento con descriptores locales puede dividirse en tres etapas

1. Detección de **keypoints**, un subconjunto de los puntos originales ya que el costo de computar descriptores es alto, y es necesario elegir qué puntos describir.

2. Codificación o elaboración de **descriptores**, tanto para el modelo como para la escena, representando la información de los objetos de forma significativa y distintiva.

3. Matching o búsqueda de **correspondencias**, asociaciones entre los descriptores del modelo y de la escena.

A partir de estas asociaciones se puede determinar si el objeto está presente o no, y su ubicación, en el caso de estar presente. Puede observarse en la Figura 4.3 como funcionaría un sistema de reconocimiento con estas etapas.

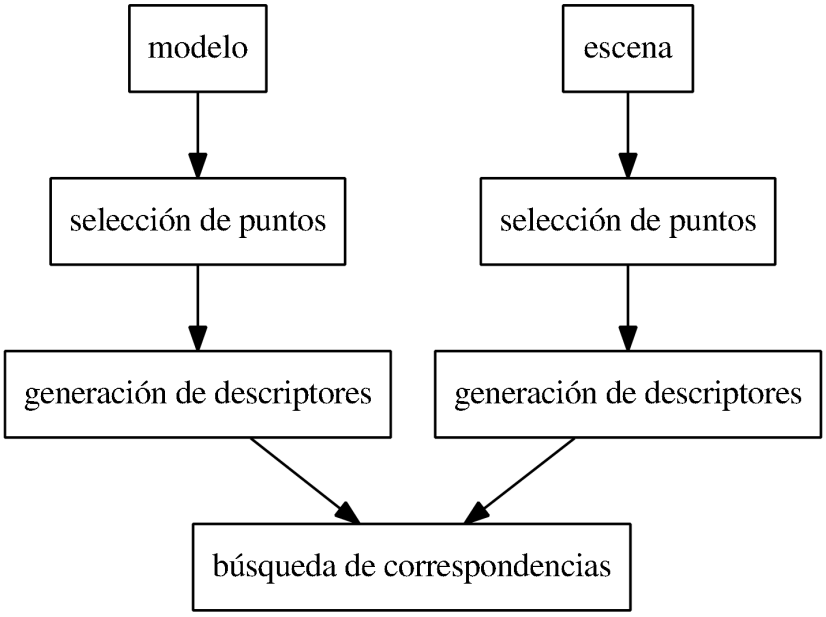


FIGURA 4.3: Diagrama de las etapas existentes en un sistema de reconocimiento de objetos.

4.2. Estructura del sistema de reconocimiento

A continuación veremos más detalladamente cada una de las etapas que conforman al sistema de reconocimiento.

4.2.1. Determinación de keypoints

Las nubes de puntos pueden tener una cantidad muy grande de puntos, por ejemplo si provienen de capturas de kinect son más de 300000. En general los algoritmos que usan la totalidad de los puntos toman una cantidad excesiva de tiempo. Una opción para incrementar la velocidad es reducir la complejidad de la nube, esto es, remover todos los puntos que no son interesantes. Los puntos interesantes o keypoints tienen las siguientes características [MBO10]:

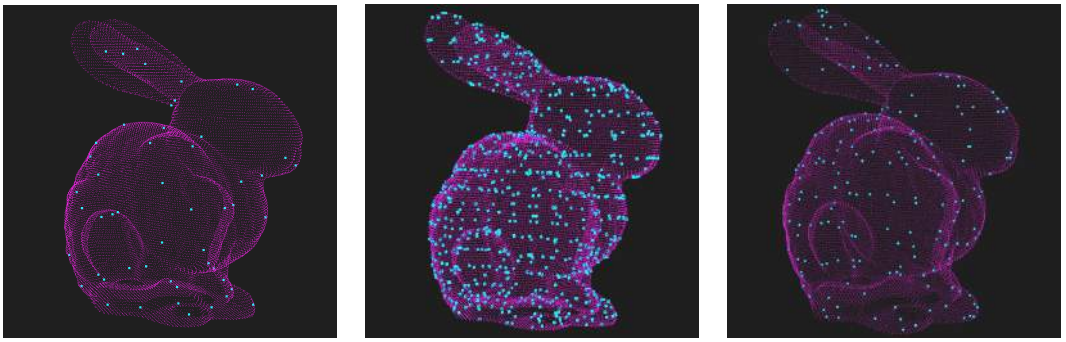
- son **repetibles**, es decir que existe una alta probabilidad de que el mismo punto sea elegido en distintas situaciones de captura de los datos, como variación del punto de vista e iluminación. Si un punto es detectado en un cuadro de una escena, es deseable que dicho punto sea también detectado en cualquier otra imagen de la misma escena.
- son **distintivos**, es decir son adecuados para lograr un reconocimiento efectivo, el entorno del punto debe proveer información significativa para caracterizar el punto. Cuando luego se elabore un descriptor en un keypoint, éste contendrá información significativa del entorno facilitanto la búsqueda de correspondencias, y, consecuentemente, la precisión de todo el sistema.

Opcionalmente puede tener invarianza a escala / resolución, que significa que un mismo punto debe poder ser elegido con alta probabilidad aunque la escala de la imagen cambie. Existen distintos detectores de keypoints, que seleccionan puntos con las características descriptas (y opcionalmente algunas más), y la elección de cuál utilizar depende de los requerimientos del sistema y de las características de las nubes de entrada [TSDS13], [STDS11].

Un detector de keypoints muy utilizado es el de Harris [HS88] que detecta esquinas, puntos en los cuales la superficie local dada por la nube varía en 2 direcciones distintas.

Otra forma de seleccionar puntos a utilizar es submuestrear los puntos de la nube, y considerar todos los puntos resultantes como los keypoints. Hay distintas formas de submuestrear, una posibilidad es elegir puntos de la nube de forma aleatoria, otra es considerar una grilla 3D sobre la nube, y para cada posición aproximar todos los puntos en ella con uno solo, ese punto puede ser el centro de esa posición en la grilla, o el centroide, resultando en un submuestreo uniforme. Se puede seleccionar el tamaño de las posiciones de la grilla, controlando la densidad de los puntos resultantes. Esta manera de elegir keypoints es rápida, pero no garantiza que los puntos seleccionados tengan las características mencionadas anteriormente.

En la Figura 4.4 se muestra la selección de puntos usando Harris y submuestreando cada 1 cm y 2 cm, para el modelo de un conejo. Puede verse que al submuestrear cada 1 cm obtenemos muchos puntos, distribuídos por toda la superficie, sin importar si la posición de un punto es distintiva o no, lo mismo ocurre para el submuestreo cada 2 cm pero con puntos más espaciados. Por otro lado, usando el detector de keypoints de Harris se obtienen pocos puntos pero estos son distintivos, están ubicados solo en partes específicas de la superficie del conejo.



(A) (B) (C)

FIGURA 4.4: Ejemplo de puntos seleccionados, usando detector de keypoints de Harris (A), usando un submuestreo uniforme de 1 cm (B), y de 2 cm (C).

4.2.2. Elaboración de descriptores

Un descriptor es una representación lo suficientemente informativa como para distinguir puntos. Las coordenadas 3D de un punto o la información RGB no son suficientes como descriptores, no son únicos, y tienen mucha variabilidad. Dos puntos pueden compartir las mismas coordenadas estando en superficies distintas, y usar únicamente la información de color trae problemas ya que cambia mucho por la iluminación y otros factores.

Un descriptor debe poseer:

- robustez frente a transformaciones rígidas (Subsección 2.5.3), es decir las rotaciones y traslaciones 3D no deben afectar la estimación del descriptor.
- robustez al ruido, los errores de medición que causan ruido no deberían cambiar demasiado la estimación del descriptor.
- invarianza a resolución, es decir que los resultados deberían ser similares aunque la densidad de puntos sea distinta.

Además si se trabaja con escenas provenientes de capturas del mundo real, el descriptor debe ser resistente a la oclusión parcial, y la presencia de datos extra [Low99].

Los decriptores locales suelen contener información no solo del punto en sí sino también de la superficie que rodea al punto, usando la vecindad del punto (ver Subsección 2.5.1) describiendo la geometría local de la superficie. Las normales son un ejemplo simple de descriptor que utiliza esta información. Muchos descriptores utilizan información de las normales, por lo cual es necesario obtener una estimación de las normales de la superficie para ellos (ver Subsección 2.5.2).

4.2.3. Búsqueda de correspondencias

Luego de haber obtenido los descriptores del modelo y la escena, queremos encontrar relaciones entre ellos para poder asociar puntos. La idea es que el descriptor de un punto en el modelo va a ser muy parecido al punto que le correspondería en la instancia del modelo en la escena. Muchos descriptores se representan con vectores numéricos que luego pueden ser comparados utilizando alguna norma vectorial, como las presentadas en la Subsección 2.5.1. Para asociar un punto p de una nube con algún punto de la otra, puede buscarse cuál es el punto q más cercano a p respecto a la distancia entre sus descriptores, realizando la búsqueda como fue explicado en la Subsección 2.5.1. Para evitar asociaciones falsas, se puede establecer un umbral máximo de distancia, tal que si la distancia entre dos descriptores es mayor a ese umbral, entonces es muy probable que no sea una buena correspondencia y sea descartada.

Si se desean encontrar todas las instancias de un modelo en la escena, hay que buscar para cada keypoint en la escena el keypoint correspondiente en el modelo. Si sabemos que hay una sola instancia del modelo, o solo nos interesa encontrar una de ellas, la búsqueda se puede hacer desde el modelo hacia la escena, es decir para cada keypoint del modelo se busca el más cercano en la escena.

Un conjunto de correspondencias en general no garantiza la presencia del objeto en la escena, por lo tanto dentro de la búsqueda de correspondencias, pueden existir las sub-etapas de rechazo de correspondencias, y agrupación de correspondencias.

Rechazo de correspondencias

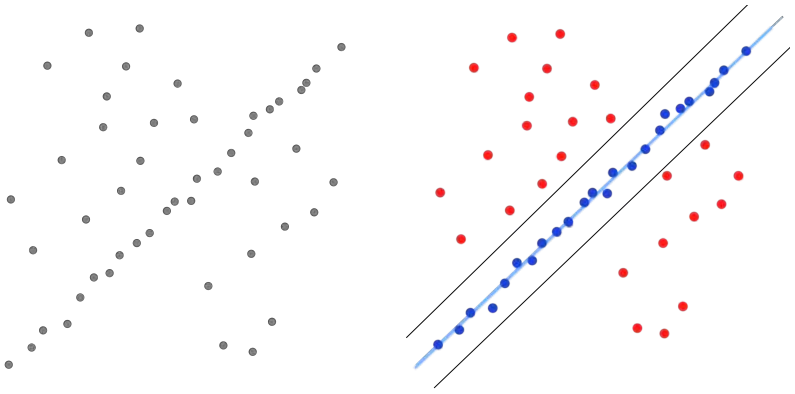
Puede pasar que existan correspondencias que deberían ser consideradas falsas, y tengan que ser eliminadas, dado que para asociar puntos usamos un umbral sobre la distancia y ese umbral no es el ideal para todos los pares de puntos (ver Sección 3.5). Se pueden rechazar correspondencias que no cumplan cierto criterio, como que cada keypoint del modelo se corresponda con solo uno de la escena o que el ángulo entre las normales de dos keypoints que se corresponden sea menor a un umbral, y que la distancia sea menor a la mediana de las distancias de todas las correspondencias. También se pueden rechazar correspondencias teniendo en cuenta la distancia al segundo vecino más cercano de cada punto, si la distancia entre el primer vecino más cercano y el segundo es muy chica, se rechaza esa correspondencia. Esta forma de rechazo se discute en la Subsección 3.5.3. Otra opción es rechazar usando RANSAC, explicado a continuación.

RANSAC

RANSAC (Random Sample Consensus) [FB81] es un método iterativo no determinístico muy usado en la práctica, otorgando excelentes resultados. Es robusto, ya que es tolerante a *outliers*, y general, ya que permite usar distintos modelos para interpretar los datos.

A partir de un subconjunto minimal con la información suficiente para determinar un modelo, se busca el conjunto maximal de puntos consistentes con éste, denominado conjunto de consenso. Es decir que en principio se usa un conjunto con la menor cantidad de información posible, y luego se agranda con información coherente. Los puntos obtenidos se denominan *inliers*.

Un ejemplo en dos dimensiones, que se puede observar en la Figura 4.5, puede ser un conjunto 2D como información, y una recta como modelo, donde el objetivo sería encontrar la recta que mejor aproxima el conjunto de puntos.



(A) Puntos 2D de entrada (B) Ajuste con RANSAC

FIGURA 4.5: Ejemplo de ajuste de puntos bidimensionales con una recta usando RANSAC. La línea celeste es el mejor ajuste obtenido, siendo los puntos azules los inliers. Las líneas negras determinan la porción del plano en la cual un punto puede ser considerado como perteneciente al modelo. Los puntos que se encuentran fuera de esta región son outliers, marcados en rojo.

En nuestro caso, el modelo es una transformación rígida, formada por una rotación y una traslación, y la información dada son las correspondencias. La transformación rígida se puede estimar con tres pares de correspondencias, por lo tanto el conjunto aleatorio inicial es de 3 elementos. El resultado es una transformación T, y un subconjunto C de las correspondencias iniciales, con la mayor cantidad de correspondencias (p, q), donde cada una cumple que q está a una distancia menor a un umbral definido del punto p transformadado, $T(p)$.

Agrupación de correspondencias

Una forma de verificar la validez de las correspondencias es mediante agrupación de correspondencias. Las correspondencias se separan en grupos llamados *clusters*. Dos métodos de agrupación son consistencia geométrica y votación de Hough [Bal81].

Consistencia Geométrica

Los pares de correspondencias deberían tener distancias similares en el modelo y en la escena si forman parte del mismo objeto. Dos pares de puntos asociados $(p_0, q_0), (p_i, q_i)$ son considerados parte del mismo grupo si la distancia entre p_0 y q_0 está dentro de cierto umbral de la distancia entre p_i y q_i. Sobre cada grupo consistente encontrado se realiza RANSAC (ver Sección 4.2.3), obteniendo una transformación y las correspondencias restantes.

Votación de Hough

Este método es más complejo y utiliza una técnica llamada votación de Hough, un método originalmente desarrollado para detectar líneas en imágenes 2D, que fue extendido para trabajar con formas arbitrarias. La idea es representar al objeto como una colección de partes, donde cada una vota en un espacio de votación discreto, que en nuestro caso se corresponde con la posición trimensional. Cada voto se corresponde con una hipótesis de la posición del objeto, y el objeto se identifica por la agrupación de votos. La posición con más votos será la elegida.

Notar que después de haber aplicado RANSAC Sección 4.2.3 a todas las correspondencias se obtiene un conjunto de correspondencias geométricamente consistentes con el modelo, por lo cual después de ese paso no es necesario aplicar un método de agrupación, las correspondencias ya están agrupadas. Por otra parte el uso de consistencia geométrica también aplica RANSAC, solo que primero separa las correspondencias en grupos consistentes, haciendo que la agrupación demore menos. De ambas formas, se obtiene una transformación rígida T (ver Subsección 2.5.3) que, aplicada al modelo lo alinea con su instancia en la escena.

4.2.4. Preprocesamiento

Antes de extraer keypoints, puede agregarse una etapa de preprocesamiento de las nubes para facilitar el trabajo y hacer más precisos los cálculos. Esta etapa en general consiste en eliminar puntos, ya sea submuestreando o con una condición.

El submuestreo es eficiente en casos en los cuales una nube tiene una densidad muy alta (ver Sección 2.5.1) para todos los puntos, siendo muy precisa, pero teniendo una cantidad excesiva de puntos con los cuales trabajar. Notar que este submuestreo previo no es reemplazado por la extracción de keypoints, ya que los keypoints solamente determinan dónde elaborar descriptores, tanto la estimación de las normales como el cómputo de los descriptores en sí hacen uso de todos los puntos en la superficie.

La eliminación condicional consiste simplemente en sacar puntos que no cumplan una restricción dada, como por ejemplo aquellos cuyo valor de profundidad sea mayor a cierto valor, si no interesa considerar puntos muy lejanos, o aquellos con una cantidad de vecinos en un radio menor a la indicada. Esta última condición sirve para eliminar valores atípicos con pocos vecinos. Estos puntos son valores deseados, que en general son causados por el ruido y errores de medición, y su eliminación no solo hace que los cómputos sean más rapidos, por tener menos puntos, sino que ayuda a obtener valores más precisos. Esto es porque para esos puntos, no puede obtenerse información significativa de su entorno, dando lugar a estimaciones incorrectas de las normales de la superficie (ver Subsección 2.5.2), y, consecuentemente, descriptores con datos erróneos. Esos descriptores luego pueden ser falsamente asociados con otros puntos, dando lugar a correspondencias falsas.

4.2.5. Postprocesamiento

La estimación de la transformación utilizada para alinear el modelo con la instancia en la escena puede refinarse utilizando el algoritmo de *Iterative Closest Point (ICP)* [BM92], [Zha94]. **ICP** minimiza la diferencia entre dos nubes de puntos, iterativamente modifica la transformación necesaria para esto. Necesita una buena estimación inicial de esta transformación, de lo contrario no garantiza un buen resultado, ya que es un método de refinación local de la solución. Un ejemplo de alineación de nubes de puntos usando ICP

puede verse en la Figura 4.6, donde las dos nubes están inicialmente cerca, y después de haber sido alineadas terminan superpuestas.

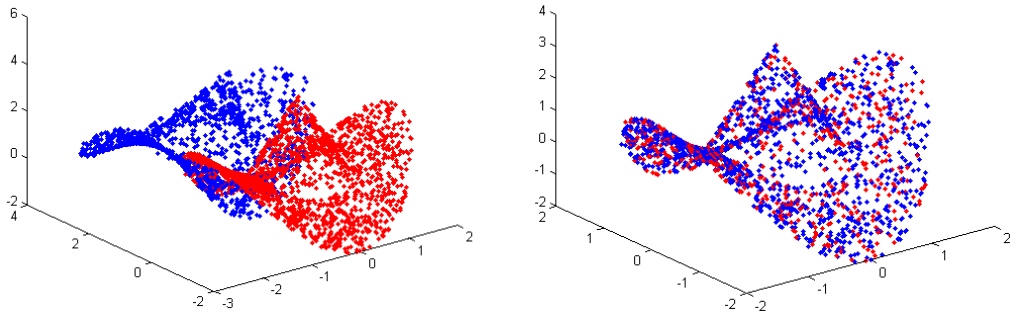


FIGURA 4.6: Ejemplo de alineación de nubes de puntos usando ICP.

En cada iteración, se busca el punto más cercano en la segunda nube por cada uno en la primera nube, luego se estima la combinación de rotación y traslación para alinear de la mejor manera posible los puntos. Por último se transforman todos los puntos de la primera nube.

4.3. Sistema implementado

El sistema fue implementado con el objetivo de poder reconocer objetos en ambientes complejos del mundo real, donde existe clutter y oclusión. Se define el *clutter* como la presencia de datos extra que no corresponden al objeto buscado, y la *oclusión* como la presencia de datos faltantes del objeto buscado. Estos problemas se tuvieron en cuenta en la elección de los métodos utilizados en cada etapa del sistema, y al momento de seleccionar los parámetros. Una discusión sobre los parámetros se encuentra en la Sección 5.5.

Es importante notar que cuando hablamos del sistema implementado en realidad nos referimos al **esquema de sistema** definido, no un único sistema particular, ya que las diferentes elecciones en las etapas del sistema determinan instanciaciones distintas, resultando en sistemas particulares potencialmente muy diferentes entre sí. Usar un descriptor distinto cambia el sistema completamente, y la modificación de los métodos de obtención de keypoints y de búsqueda de correspondencias también lo hacen. Cambiar el descriptor muchas veces implica cambiar la distancia asociada, y además puede pasar que un descriptor dependa del dector de keypoints utilizado. Por ejemplo, el descriptor NARF ([SRKB11]) necesita que los puntos a describir sean cercanos a bordes, para que el descriptor sea más informativo.

4.3.1. Estructuras de datos

Para almacenar los datos, para el modelo y la escena, definimos una estructura que contiene la siguiente información

- la **nube original**, conteniendo todos los puntos
- las **normales** de todos los puntos
- los **keypoints** seleccionados
- los **índices** en la nube original de los keypoints seleccionados
- los **descriptores** correspondientes a los keypoints seleccionados

Los keypoints son simplemente copias de los puntos de la nube original en las posiciones indicadas por los **índices**. Guardamos información redundante para facilitar el desarrollo del sistema.

4.3.2. Preprocesamiento

Preprocesamos las nubes removiendo puntos que no tienen suficientes vecinos en un radio de 1cm, para hacer al sistema resistente al ruido, usando distintos umbrales mínimos para la cantidad de vecinos. Este umbral depende de las características esperadas de las nubes de entrada, si el umbral es muy grande puede hacer que se pierdan detalles necesarios en las nubes; y por otro lado, si es muy chico la eliminación de ruido es menor. Este valor lo modificamos en el código según el caso.

4.3.3. Determinación de keypoints

Para seleccionar los puntos a utilizar, tenemos la opción de usar el detector de keypoints de Harris, y de realizar un submuestreo uniforme o aleatorio (Subsección 4.2.1). Para el submuestreo aleatorio se permite elegir el tamaño de cada posición de la grilla 3D utilizada, pudiendo ser distinto para el modelo y la escena.

Cuanto más grande sea ese tamano, menor será la cantidad de puntos seleccionados, y hay que tener en cuenta el tamaño del objeto buscado para poder determinarlo, por ejemplo si se seleccionan puntos cada 10 cm, entonces se perderán muchos detalles que hacen que el reconocimiento de objetos chicos sea muy dificil o imposible. Un tamaño más chico hace que se consideren más puntos, dando más detalle, pero también incrementando el tiempo total del sistema.

4.3.4. Elaboración de descriptores

Para elaborar los descriptores, primero estimamos las normales (Subsección 2.5.2), dado que todas las opciones de descriptores en nuestro sistema las necesitan, que son FPFH [RBB09], SHOT [TSDS10], CSHOT [TSDS11] y Spin Images [Joh97], descriptos en el Capítulo 3. Para la estimación usamos todos los vecinos de un punto en un radio dado, parametrizable. Este radio controla la cantidad de información de la superficie que se incorpora, si el valor es muy chico la estimación no será muy significativa, pero si es muy grande puede incoporar valores indeseados de ruido.

La estimación se hace para todos los puntos de las nubes, aunque los descriptores solo se elaboren en los keypoints, porque los descriptores utilizan información de los puntos de su entorno, sean o no keypoints.

Para elaborar los descriptores establecemos:

- Parámetros particulares al descriptor elegido
- La nube original
- Las normales de los puntos
- Los índices en los cuales queremos que se elaboren los descriptores
- El radio del entorno considerado para la elaboración de los descriptores

El proceso es básicamente el mismo para todas las opciones de descriptores, únicamente modificando los parámetros particulares, como por ejemplo el coseno del ángulo de soporte para Spin Images (ver Sección 3.4). La instanciación de dichos parámetros para cada opción de descriptor está comentada en el Capítulo 3.

4.3.5. Búsqueda de correspondencias

Al buscar correspondencias, usamos opcionalmente un árbol k-d (Sección 2.5.1) o simplemente una búsqueda exhaustiva de todas las opciones de pares de puntos. Esta búsqueda es desde el modelo hacia la escena porque sabemos que hay una sola instancia del modelo, y esto ayuda a reducir la aparición de correspondencias falsas.

Luego de la obtención de correspondencias, permitimos aplicar un rechazador de correspondencias basado en RANSAC (Sección 4.2.3), o alguno de los dos métodos de agrupamiento explicados previamente, Hough o Consistencia Geométrica (Sección 4.2.3). Los

métodos de agrupación devuelven un conjunto de clusters, donde consideramos como correcto aquel que tenga la mayor cantidad de correspondencias y menor promedio de las distancias al cuadrado entre las correspondencias.

Precisión

Por último, opcionalmente se muestra una medida de precisión, en el caso de existir información de ground truth se evalua como se explica en la Sección 5.4; y en caso de no existir se estima con el promedio de errores de correspondencias.

Capítulo 5

Resultados

En este capítulo analizaremos el rendimiento y la calidad del sistema (Sección 4.3) utilizando dos bases de datos públicamente disponibles con información de ground truth.

5.1. Introducción

Usando el esquema de sistema definido en la Capítulo 4, queremos evaluar el desempeño de los distintos sistemas que surgen de las diferentes elecciones de parámetros y métodos de selección de keypoints, descriptor, búsqueda de correspondencia y agrupación de las mismas. En la Sección 3.5 vimos el poder discriminativo de los descriptores utilizados, dando una idea de la utilidad de cada una, ahora nos interesa evaluar el comportamiento de todo el sistema en conjunto.

Consideramos dos aspectos del sistema:

- la **precisión**, calculada a partir de la información de ground truth de las bases, como se explica en la Sección 5.4, que mide cuán bien funciona el reconocimiento de objetos, es decir, cuán cercana es la solución encontrada a la real.
- el **tiempo** de las etapas de selección de keypoints, elaboración de descriptores, búsqueda de correspondencias, y agrupación o rechazo de correspondencias.

Los tiempos los separamos en etapas para poder encontrar el cuello de botella del sistema, y comparar más finamente el impacto que las distintas elecciones de parámetros y métodos tienen en el tiempo de las etapas. Estos dos aspectos están muy relacionados,

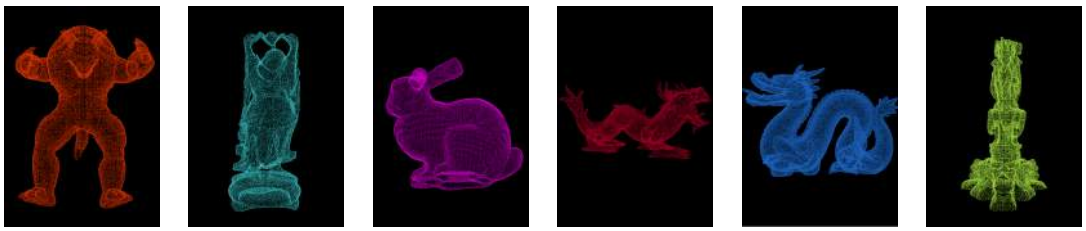
en general un incremento en la precisión lleva a que el sistema tarde más, y una disminución del tiempo implica una pérdida de precisión.

El sistema es evaluado usando dos bases de características muy diferentes, siendo una un caso que se puede considerar ideal, y la otra un caso real. La primera base, de CVLab, es el caso ideal, tiene modelos muy detallados representados con nubes muy densas, y las escenas son sintéticas. Además la información de ground truth provista es en forma de la *pose* (ver Subsección 2.5.3) de los objetos, permitiendo saber con exactitud la diferencia entre el resultado obtenido por el sistema y el esperado. La segunda base, de RGBD, es un caso real, contiene modelos y escenas provenientes de capturas del mundo real, conteniendo ruido y datos faltantes, además en las escenas los objetos no aparecen solos sino que existen datos de fondo que no interesan para el reconocimiento. En este caso la información de ground truth es en forma de *bounding boxes* para los objetos, que no provee tanta información como la pose. Además no hay garantías de que esa información sea precisa en todos los casos, puede pasar que las bounding boxes sean demasiado chicas o demasiado grandes, complicando la evaluación del sistema.

En el resto de este capítulo describimos las bases de datos utilizadas para los análisis de resultados, el método de evaluación utilizado, y los resultados obtenidos, presentando una discusión sobre precisión y tiempos.

5.2. Base de CVLab

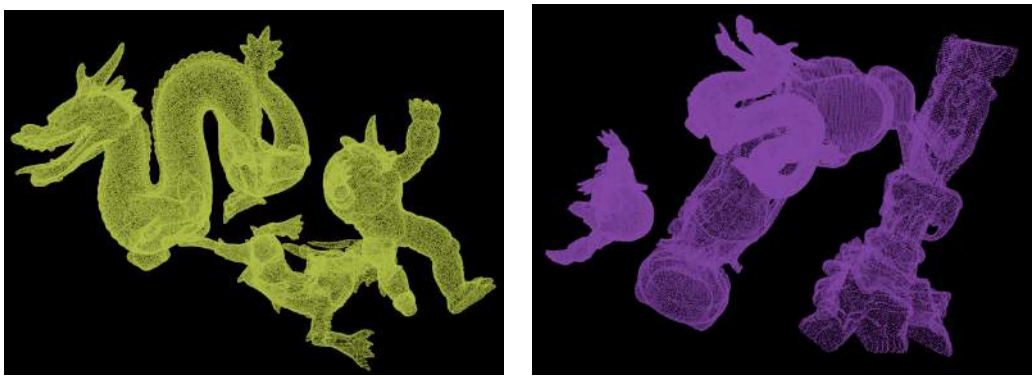
Utilizamos los datasets 1 y 2 del repositorio de CVLab [Bol], publicado junto con el artículo de Tombari, Salti y Stefano donde presentan el descriptor de SHOT [TSDS10], que contienen modelos del repositorio de Stanford [Sta].



(A) (B) (C) (D) (E) (F)

FIGURA 5.1: Vistas de los modelos de la base, de izquierda a derecha, Armadillo (A), Buddha (B), Bunny (C), Chinese Dragon (D), Dragon (E) y Statuette (F).

Cada una de estas bases está compuesta por 6 modelos y 45 escenas. Los modelos son los mismos para las dos bases, pueden verse vistas de ellos en la Figura 5.1. Cada escena contiene un subconjunto de los 6 modelos elegidos en orden aleatorio que luego son girados y trasladados de forma también aleatoria, resultando en escenas como las presentadas en la Figura 5.2. Además se agrega ruido gaussiano a las escenas. Para el segundo dataset, la densidad de los puntos varía, siendo $1/8$ de la original.



(A) Escena con 3 modelos: armadillo, chinese dragon y dragon

(B) Escena con 5 modelos: armadillo, buddha, bunny, dragon y statuette

FIGURA 5.2: Vistas de escenas con 3 y 5 modelos.

Por cada escena existe un archivo de configuración que indica la cantidad de modelos en dicha escena, y por cada uno de ellos se menciona el nombre y el archivo que contiene la información de ground truth. Esta información de ground truth consiste en la transformación T (compuesta por una rotación y una traslación) que hay que aplicarle al objeto para alinearlo con la instancia correspondiente en la escena, dada por los 16 números reales $t_{00}, \ldots, t_{33}$ que la conforman, de la siguiente forma:

$$T = \begin{pmatrix} t_{00} & t_{01} & t_{02} & t_{03} \\ t_{10} & t_{11} & t_{12} & t_{13} \\ t_{20} & t_{21} & t_{22} & t_{23} \\ t_{30} & t_{31} & t_{32} & t_{33} \end{pmatrix}$$

5.3. Base de objetos RGB-D

La base de datos de objetos RGB-D [1] [LBRF11a] es de gran escala, está organizada de forma jerárquica, tiene múltiples vistas de objetos y contiene datos visuales (color) y de profundidad para cada objeto. La base contiene 300 objetos organizados en 51

[1] http://www.cs.washington.edu/rgbd-dataset

categorías, que son comúnmente encontrados en una oficina y una casa. Las imágenes de los objetos tienen fondo (no se encuentra solo el objeto) pero se proveen máscaras de segmentación para poder separarlo del objeto en sí. En la Figura 5.3 se presentan vistas de objetos cuyo fondo ha sido removido.

FIGURA 5.3: Vistas de algunos objetos presentes en la base, sin el fondo.

Obtención de Imágenes

Para la recolección de datos se usaron dos cámaras calibradas: una cámara RGB-D , y una cámara RGB firewire grasshopper de Point Grey Research [2]. Para sincronizar las imágenes de ambas, se usan timestamps para asociar cada imagen de la cámara RGB con la correspondiente de la RGB-D (aquella que ocurra más cercana en el tiempo).

La Cámara RGB-D

Simultáneamente toma imágenes de color y de profundidad de una resolución de 640x480. Es decir, que cada píxel contiene cuatro canales: rojo, verde, azul y profundidad. La ubicación de cada píxel en el espacio se puede computar usando parámetros conocidos de los sensores y la fórmulas 2.1 presentadas en el Sección 2.4. Captura datos a una frecuencia de 20 Hz. El driver de la cámara se asegura que las imágenes RGB y de profundidad estén alineadas y sincronizadas en el tiempo.

La Cámara RGB

Está montada sobre la cámara RGB-D, y provee imágenes RGB a una resolución mayor (1600x1200), capturando datos a una frecuencia de aproximadamente 12 Hz.

[2]Point Grey Research http://www.ptgrey.com/

Captura de los modelos

Los objetos se ubican en una mesa giratoria, y se capturan secuencias de video mientras giran a una velocidad constante. Las cámaras están ubicadas a aproximadamente un metro de dicha mesa. Cada secuencia de video se captura a 20 Hz y contiene cerca de 250 frames, dando un total de 250.000 frames de RGB y profundidad.

Se tomaron datos con las cámaras montadas a 3 alturas diferentes respecto de la mesa: a aproximadamente $30°$, $45°$ y $60°$ sobre la línea horizontal. Se capturó una revolución de cada objeto a cada altura. La Figura 5.4 muestra capturas de una de las instancias del modelo de una gorra, puede verse como el objeto va rotando, haciendo que se vean distintas partes de él según el ángulo.

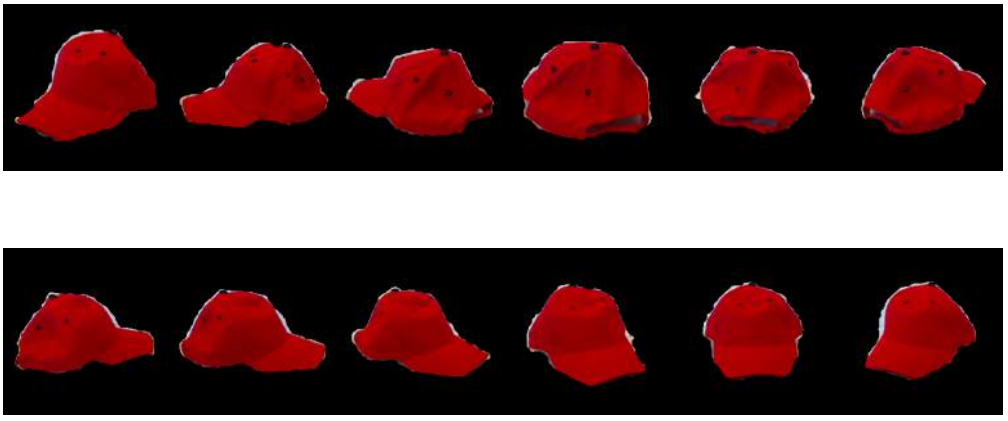


FIGURA 5.4: Capturas a distintos ángulos de rotación del modelo *cap*.

La mesa giratoria tiene marcas para poder determinar el ángulo cuando va girando. Este ángulo está entre $0°$ y $360°$.

Captura de las escenas

La base de datos también incluye 8 secuencias de video de escenas. Estas escenas son de ambientes interiores comunes como áreas de trabajo, áreas de reunión y de cocina. Algunos cuadros de estas escenas pueden verse en las Figuras 5.5, 5.6, 5.7, y 5.8.

Estas escenas fueron grabadas con la cámara RGB-D a aproximadamente el nivel del ojo humano mientras se caminaba alrededor del lugar. En cada escena hay varios objetos que están en la base de datos de objetos. Dichos objetos son visibles de múltiples puntos de vista y distancias y pueden estar parcial o totalmente ocluidos en algunos cuadros.

FIGURA 5.5: Cuadros de la escena desk_1

FIGURA 5.6: Cuadros de la escena desk_2

FIGURA 5.7: Cuadros de la escena kitchen_small_1

FIGURA 5.8: Cuadros de la escena table_1

Anotaciones en las Escenas de Video

La información de ground truth para cada cuadro de las escenas de la base de datos consiste en las coordenadas del rectángulo *bounding box*, que contiene a cada objeto presente en el cuadro.

desk_1 desk_2

kitchen_small_1 table_1

FIGURA 5.9: Escenas 3D reconstruídas.

Para anotar las escenas se une toda la secuencia de video para crear una reconstrucción 3D de toda la escena, mientras se sigue la posición de la cámara en cada cuadro. Para reconstruir las escenas se usa una técnica de mapeo RGB-D propuesta en el trabajo de Henry, Krainin, Herbst, Ren y Fox [HKH+10] que consiste de dos componentes principales: alineamiento espacial de frames sucesivos y alineamiento global consistente de toda la secuencia. Algunas escenas reconstruídas pueden verse en la Figura 5.9.

Luego, se anotan los objetos en esta reconstrucción a mano, y se proyectan las anotaciones a las posiciones conocidas de la cámara en cada cuadro, resultando en las *bounding boxes* de cada escena.

5.4. Método de evaluación

Obtención de datos

Para cada modelo, consideramos todas las escenas en las que esté presente para cada base de datos, y evaluamos el sistema con ese modelo y cada una de las escenas. Se puede saber en qué escenas está presente un objeto a partir de los archivos de ground truth de las escenas. Por cada par modelo-escena, obtenemos 8 medidas de tiempos y 1 o más medidas de precisión.

En detalle, esta información es:

t_1 tiempo de selección de keypoints para el modelo

t_2 tiempo de selección de keypoints para la escena

t_3 tiempo de estimación de normales para el modelo

t_4 tiempo de generación de descriptores para el modelo

t_5 tiempo de estimación de normales para la escena

t_6 tiempo de generación de descriptores para la escena

t_7 tiempo de búsqueda de correspondencias entre descriptores

t_8 tiempo de agrupación de correspondencias

medida(s) de precisión del resultado

Dentro de agrupación de correspondencias consideramos rechazo usando RANSAC con todas las correspondencias, consistency grouping y votación de Hough según el caso (ver Subsección 4.2.3).

Generación de gráficos

A partir de la información obtenida generamos dos gráficos de barras representando los tiempos, separando el procesamiento del modelo con el reconocimiento en la escena. Para el primer gráfico utilizamos tres valores que provienen del promedio de los tiempos t_1, t_3 y t_4, es decir los que corresponden al modelo, estos tienen muy poca varianza porque el modelo siempre es el mismo y el proceso también. Para el segundo gráfico, utilizamos cinco valores que provienen del promedio de los tiempos de t_2, t_5, t_6, t_7 y t_8, es decir los que corresponden a la escena; consideramos el tiempo de búsqueda de correspondencias y de agrupamiento para este gráfico ya que varían para cada escena.

En los gráficos mostramos el tiempo de selección de keypoints, el tiempo de generación de descriptores, dentro del cual marcamos en un color más oscuro el tiempo de estimación de normales, el tiempo de búsqueda de correspondencias y de agrupación, en el caso de las escenas.

Con la precisión, dada por una (Ecuación 5.2) o dos medidas (Ecuación 5.1), también realizamos un promedio.

Medidas de precisión

La forma de medir la precisión varía según la base, para la base de CVLab consiste de dos medidas de error y para la de objetos de RGBD de una medida de superposición, ambas explicadas a continuación.

Precisión para la base de CVLab

Dada la transformación T_{gt} leida del archivo de ground truth, y T_s, la obtenida por el sistema, las comparamos para tener dos medidas de precisión. Primero obtenemos la transformación $T_\delta \in \mathcal{R}^{4x4}$ que representa la diferencia entre transformaciones

$$T_\delta = T_s . {T_{gt}}^{-1} = \begin{pmatrix} R_\delta & t_\delta \\ 0 & 1 \end{pmatrix}$$

En el caso ideal, la transformación encontrada es igual a la de ground truth entonces T será la identidad.

Consideramos para la precisión una medida de error de rotación y una medida de error para la traslación [PCSM13] definidas como

$$\begin{aligned} err_R &= \text{acos}\left(\frac{\text{traza}\,(R_\delta) - 1}{2}\right) \\ err_t &= ||t_\delta||_2 \end{aligned} \tag{5.1}$$

Si T_δ es la matríz identidad, entonces $R_\delta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, y $t_\delta = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$. La traza de R_δ es 3, por lo tanto $err_R = \text{acos}\left(\frac{3-1}{2}\right) = \text{acos}\ (1) = 0$, y err_t es la norma de t_δ cuyos valores son todos 0, por lo tanto vale 0 también. Entonces valores más chicos de err_R y err_t se corresponden con una mayor cercanía de T_δ a la identidad, indicando una mayor precisión del resultado.

La idea es medir la magnitud de la diferencia entre la pose obtenida y la dada por el ground truth. err_t mide el tamaño de la traslación, y err_R el ángulo de la rotación ([Ma04]). Entonces err_t toma valores mayores o iguales a 0 por ser una norma (euclídea en este caso), y err_R toma valores entre 0 y π porque es el ángulo de la rotación en radianes.

Precisión para la base de objetos RGBD

Una vez conseguida la transformación rígida T, podemos obtener la bounding box b_s del sistema siguiendo los pasos descriptos a continuación.

Consideramos los keypoints del modelo $\mathcal{K}$ y los transformamos usando T, obteniendo $\mathcal{K}_T$. Luego, para cada uno de los puntos p en $\mathcal{K}_T$ buscamos cuál es el punto q más cercano a él en la escena (ver Subsección 2.5.1), resultando en una nueva nube de puntos $\hat{\mathcal{K}}$.

$\hat{\mathcal{K}}$ contiene puntos en la escena que representan la instancia del modelo encontrada, calcular la bounding box para esta nube es lo que buscamos. Los puntos de $\hat{\mathcal{K}}$ son tridimensionales y la información de ground truth es en dos dimensiones, para poder comparar ambas tenemos que trabajar en las mismas dimensiones. Decidimos primero proyectar los puntos de la nube a dos dimensiones y luego calcular el rectángulo correspondiente a la bounding box para la nube de puntos proyectados.

Para realizar esta proyección de 3D a 2D, cada punto $p = \{p_x, p_y, p_z\}$ de $\hat{\mathcal{K}}$, de manera inversa a como generamos la nube a partir de imágenes RGBD (ver Ecuación 2.1)

$$\begin{aligned} p_x &= ((p.x + f) \,/\, p.z) + c_x \\ p_y &= ((p.y + f) \,/\, p.z) + c_y \\ p_z &= 0 \end{aligned}$$

donde c_x, c_y son los centros de la cámara en x y en y respectivamente, y f es la distancia focal, que es igual en x y en y. Finalmente, obtenemos b_s teniendo en cuenta los valores mínimos y máximos de x e y para determinar las esquinas del rectángulo.

Consideramos la intersección entre la bounding b_{gt} leida del archivo de ground truth, y b_s, que será un rectángulo. La medida de precisión será la superposición, o el *overlap*, determinada por el área de la intersección sobre el área de la unión [EVGW+10]:

$$\text{superposicion}\,(b_s, b_{gt}) = \text{▲}\,(b_s \cap b_{gt}) \,/\, \text{▲}\,(b_s \cup b_{gt}) \tag{5.2}$$

5.5. Parámetros

Analizamos el rendimiento del sistema desarrollado con ambas bases para poder determinar conjuntos de parámetros útiles. Constantemente nos encontramos con la dificultad de balancear la precisión y el tiempo, una mejora en la precisión la mayoría de las veces implica más demora en el sistema, y una ganancia de velocidad significa una pérdida de precisión. Un sistema puede ser muy rápido pero para nada preciso, fallando constantemente, haciendo que no sea útil; y por otro lado, un sistema extremadamente preciso que tarda horas en emitir un resultado tampoco es apropiado.

Buscamos un sistema adecuado para las situaciones distintas dadas por cada una de las bases. La base de CVLab tiene escenas sintéticas y modelos con mucha definición, en la cual es simple obtener un buen resultado, en ella nos interesa ganar velocidad, mientras que el sistema en todos los casos devuelva un resultado aceptable. La definición de qué valores de error constituyen resultados aceptables fue determinada experimentalmente, observando los resultados. Consideramos como aceptables valores de err_t menores a $0{,}1$, correspondiente a 10 cm, y valores de err_R menores a $0{,}52$, correspondiente a aproximadamente 30 grados, ya que valores más grandes hacen que el reconocimiento no sea tan eficiente, como puede verse en las Figuras 5.11 y 5.12. Además en la Figura 5.10 se puede observar un ejemplo de reconocimiento exitoso con valores muy bajos de error.

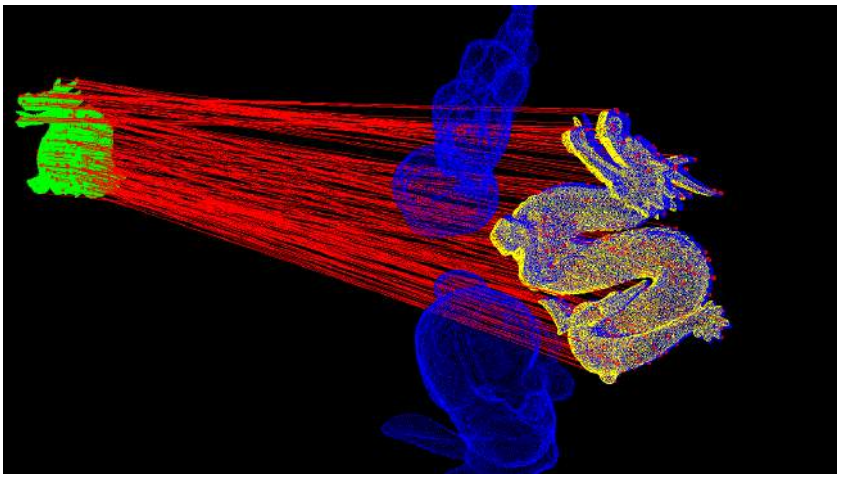


FIGURA 5.10: Resultado de reconocimiento del modelo *Dragon* en una de las escenas del *dataset_1*, con valores de precision $err_R = 0{,}03$ y $err_t = 0{,}07$.

En el caso de la base de objetos RGBD, nos interesa conseguir precisión, sin insumir excesivo tiempo. En esta base es muy difícil obtener buenos resultados, existe mucho ruido, poca definición, muchos objetos, datos de más y datos faltantes (clutter y oclusión).

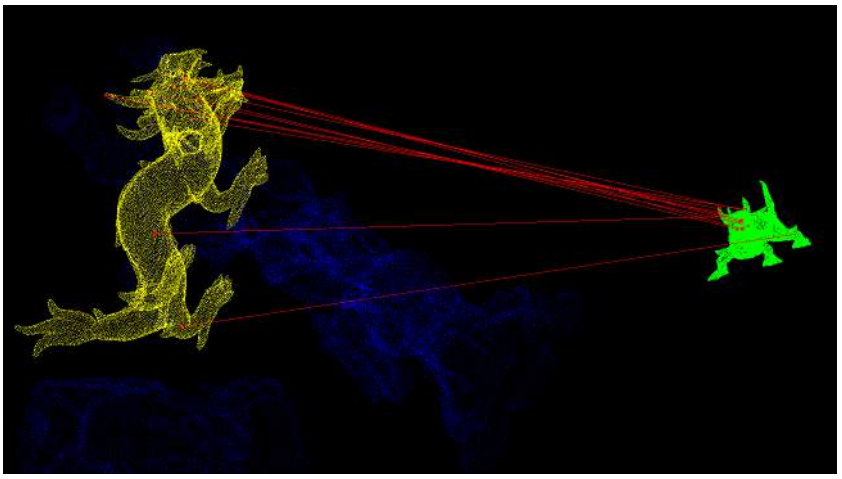


FIGURA 5.11: Resultado de reconocimiento del modelo *Chinese Dragon* en una de las escenas del *dataset_1*, con valores de precision $err_R = 0{,}59$ y $err_t = 0{,}27$.

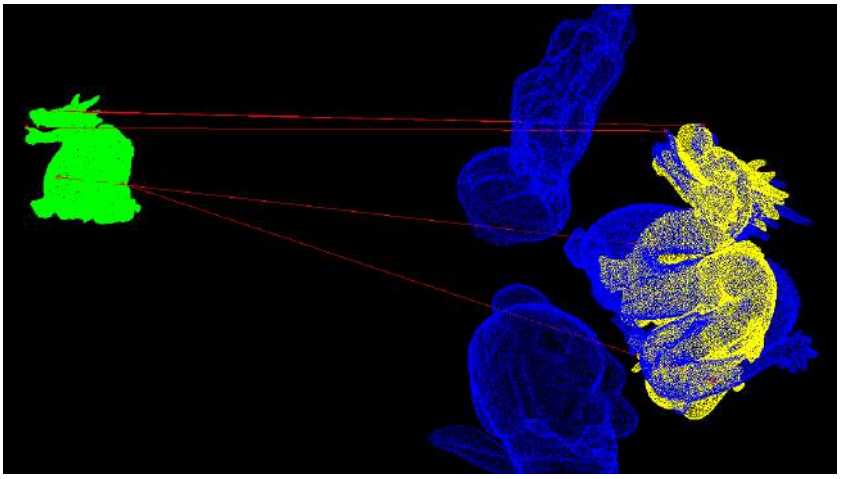


FIGURA 5.12: Resultado de reconocimiento del modelo *Dragon* en una de las escenas del *dataset_1*, con valores de precision $err_R = 0{,}69$ y $err_t = 0{,}12$.

Como solo probamos situaciones en las cuales sabemos que el objeto se encuentra en la escena, todos los resultados deberían ser exitosos. En la base de CVLab con un conjunto de parámetros que hace que las medidas de precisión se mantengan dentro de valores aceptables, en ningún caso ocurre que el objeto no es encontrado en la escena. En la base de objetos de RGBD no es posible utilizar esta información como criterio, ya que en algunos casos aunque el objeto esté presente en la escena, casi ni es visible, o es visible pero hay mucho ruido, y el reconocimiento puede fallar. Para esta base, el valor de porcentaje considerado como bueno es diferente, porque depende de la calidad del ground truth, puede ocurrir que el porcentaje es bajo aunque el objeto esté cerca de donde debería estar, como se observa en las Figuras 5.13, 5.14 y 5.15.

En la mayoría de los casos, valores mayores a 20 % representan resultados buenos, donde el objeto fue encontrado muy cercano a su posición real. Pueden verse en las Figuras

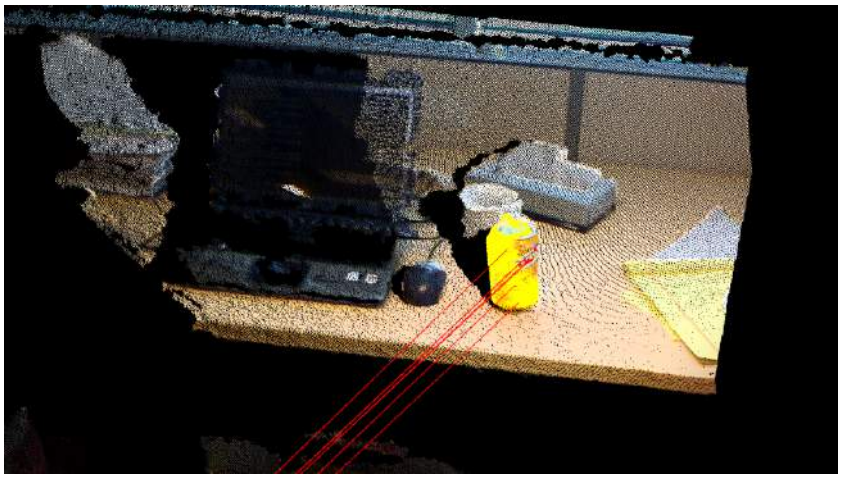


FIGURA 5.13: Resultado de reconocimiento del modelo *soda_can_6* en una de las escenas de *desk_1*, con porcentaje de superposición de $0\,\%$.

FIGURA 5.14: Resultado de reconocimiento del modelo *coffee_mug_1* en una de las escenas de *desk_1*, con porcentaje de superposición de $5\,\%$.

5.16, 5.17 y 5.18 ejemplos exitosos de reconocimiento con distintos porcentajes de superposición.

5.5.1. Preprocesamiento

Para la eliminación de puntos según la cantidad de vecinos, usamos un radio de 1 cm, y la cantidad mínima de vecinos la determinamos según la situación. Las nubes de la base de objetos RGBD tienen poca densidad de puntos e introducir restricciones muy fuertes hace que se pierdan datos útiles, puede pasar por ejemplo que los puntos del objeto que se está buscando sean eliminados con el filtrado, y este no es el efecto deseado. Usamos

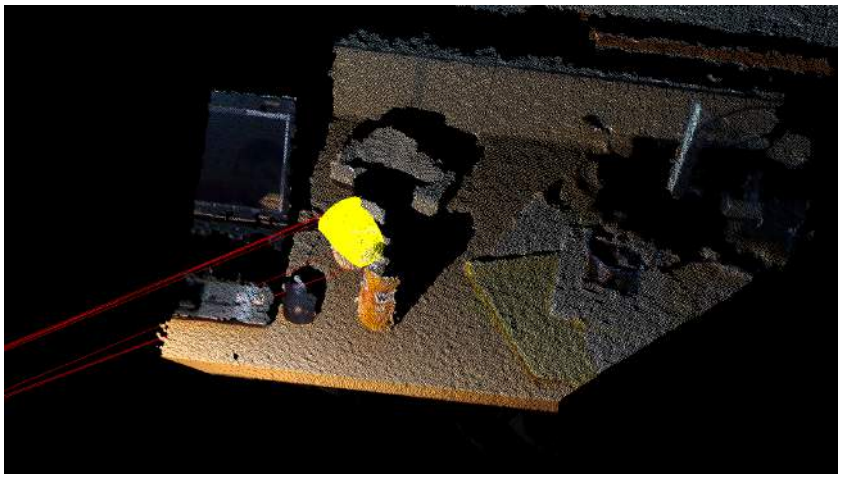


FIGURA 5.15: Resultado de reconocimiento del modelo *coffee_mug_1* en una de las escenas de *desk_1*, con porcentaje de superposición de $16\,\%$.

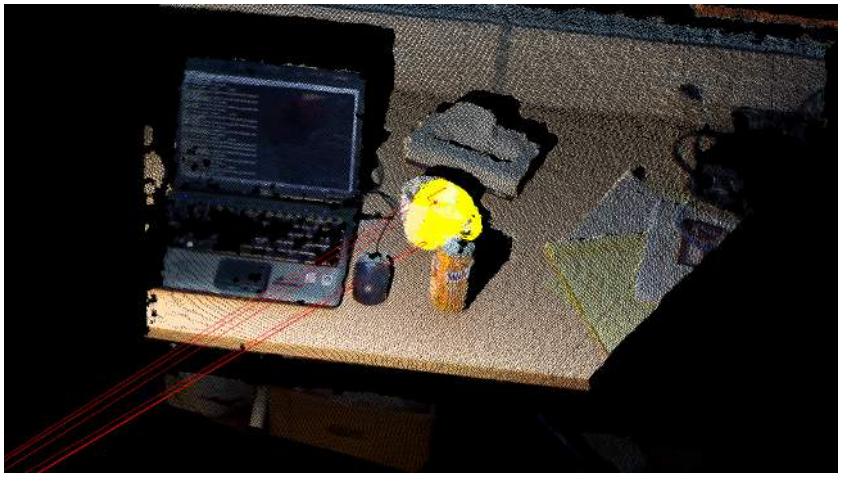


FIGURA 5.16: Resultado de reconocimiento del modelo *coffee_mug_1* en una de las escenas de *desk_1*, con porcentaje de superposición de $24\,\%$.

entonces una restricción conservadora de 5 vecinos. Para valores más grandes, de hasta 20 vecinos, el sistema sigue funcionando en la mayoría de los casos, aunque ocurre el problema de la sobre eliminación de puntos para algunas escenas. Encontramos que aplicar este paso previo de preprocesamiento mejora el reconocimiento, aunque no sea una restricción muy fuerte, porque se deshace de muchos puntos atípicos con muy pocos vecinos.

Para la base de CVLab usamos una restricción mucho más fuerte, de 200 vecinos en ese radio de 1 cm, ya que los modelos son muy densos y las escenas son sintéticas, por lo cual no se remueven puntos de más. Este paso no es estrictamente necesario en este caso porque no hay valores atípicos, lo hacemos solamente para simplificar las nubes.

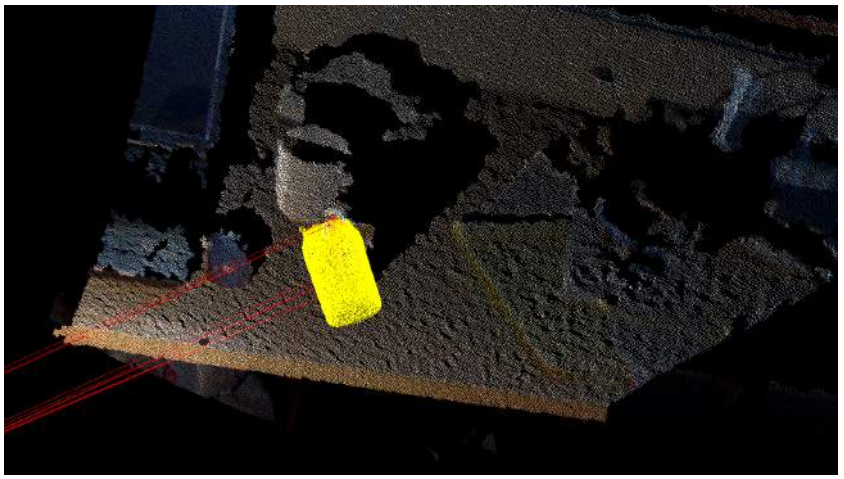


FIGURA 5.17: Resultado de reconocimiento del modelo *soda_can_6* en una de las escenas de *desk_1*, con porcentaje de superposición de $50\,\%$.

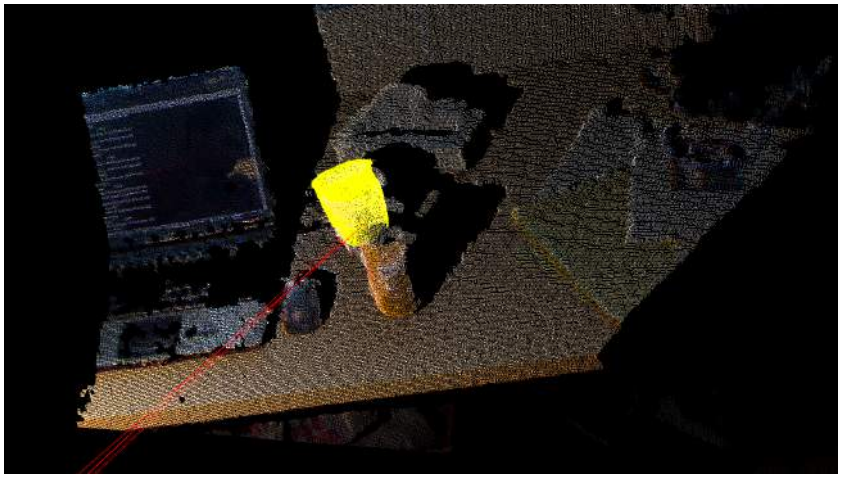


FIGURA 5.18: Resultado de reconocimiento del modelo *soda_can_6* en una de las escenas de *desk_1*, con porcentaje de superposición de $55\,\%$.

5.5.2. Selección de Keypoints

En nuestro sistema tenemos la opción de usar el detector de keypoints de Harris, o de submuestrear de manera uniforme o aleatoria. El dector de Harris es lento, necesita analizar los puntos para poder determinar cuán buenos son como keypoints, y necesita calcular las normales para este análisis, lo cual es computacionalmente costoso.

En la base de CVLab usar keypoints obtenidos con Harris mejora ligeramente la precisión pero empeora el tiempo, comparado con un submuestreo. En la base de RGBD no se observó esta mejora, los datos de entrada no son lo suficientemente buenos como para poder determinar con precisión los keypoints, en algunos casos ocurre que se seleccionan muy pocos puntos, y esos no son significativos, o se seleccionan demasiados, haciendo

que sea desventajoso usarlos todos. Por esta razón descartamos el uso de Harris para la base RGBD.

El submuestreo, es una alternativa rápida y fácil para la selección de keypoints, aunque los puntos submuestreados no tienen ninguna característica especial, pueden ser parte del fondo o de los objetos.

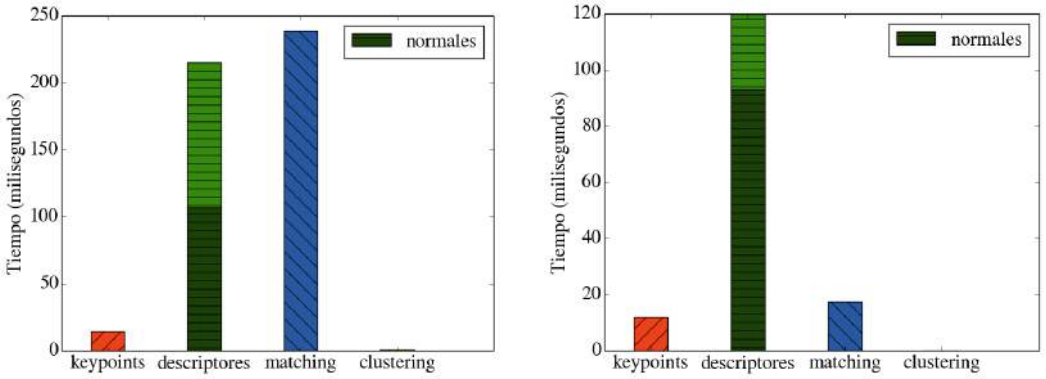
El submuestreo aleatorio trae mucha variabilidad, y puede resultar en muy pocos puntos útiles, es decir sobre el modelo a buscar, por lo cual no fue utilizado.

Submuestreo Uniforme

El submuestreo uniforme otorga resultados mucho más esperables, según el tamaño de muestreo, un radio en nuestro caso. Cada posición de la grilla 3D que se considera para dividir la nube tiene el radio de muestreo, por lo tanto cuando menor sea el tamaño más posiciones habrá, y como consecuencia más puntos resultantes. En general disminuir el radio mejora el reconocimiento, porque se consideran más puntos, incrementando la posibilidad de que haya puntos pertenecientes a la instancia del modelo en la escena, que son los que interesan para poder obtener correspondencias válidas. La desventaja de usar un radio más chico es el tiempo que lleva trabajar con todos estos puntos, generar descriptores para todos ellos y compararlos. Si el radio es demasiado chico, se pierde el sentido de haber realizado un submuestreo.

Consideremos por ejemplo variar el tamaño del submuestreo para el caso del modelo de conejo de la base de CVLab (ver Figura 5.1C) en las escenas del segundo dataset, dejando los tamaños de elaboración de descriptor y estimación de normales fijos en 3cm.

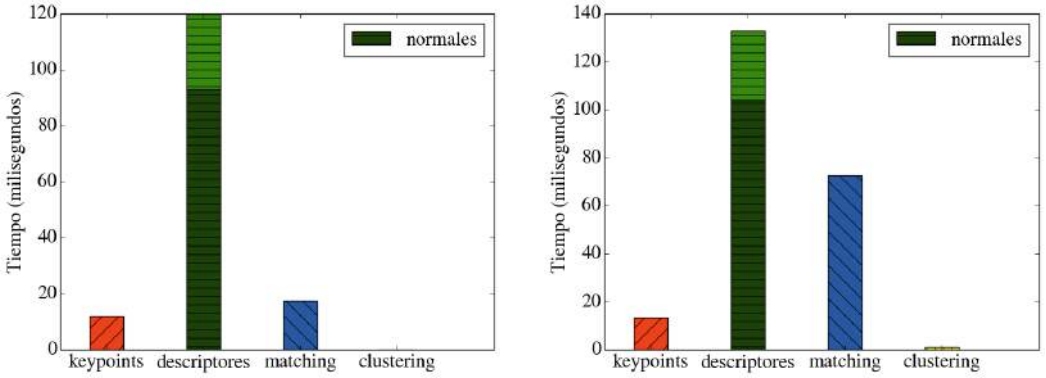
Usando un submuestreo uniforme de 1cm tarda 468ms, que es más del triple que usando uno cada 2cm, que demora 149ms, pudiendo observarse la distribución de los tiempos en etapas en la Figura 5.19. Las etapas de elaboración de descriptores y búsqueda de correspondencias tardan más por la mayor cantidad de puntos, siendo más notoria la diferencia en esta última etapa, dando la idea de que es más computacionalmente costoso en este caso la búsqueda de vecino más cercano que la elaboración del descriptor. La precisión es mejor, siendo $err_R = 0{,}096$ y $err_t = 0{,}023$ usando 1cm, y $err_R = 0{,}163$ y $err_t = 0{,}045$ usando 2cm. Ambos valores de precisión son buenos, la rotación es menor a 10 grados en ambos casos, y la traslación menor a 5cm.



(A) (B)

FIGURA 5.19: Tiempo promedio de cada etapa para el modelo *bunny* en las escenas de *dataset_2*, usando un submuestreo uniforme cada 1cm ($\blacktriangle$) y cada 2cm (B), tanto en el modelo como en la escena para los dos casos.

Existe también la posibilidad de modificar el muestreo en el modelo y en la escena, una buena opción es muestrear de forma más fina en el modelo que en la escena. Usando el mismo ejemplo de antes, vemos la diferencia entre usar un submuestreo uniforme de 2cm tanto para el modelo como para la escena, y usar 1cm en el modelo y 2cm en la escena. Nuevamente el tiempo se modifica por la cantidad de puntos utilizados, tardando 149ms en el primer caso, y 219ms en el segundo. Puede observarse en la Figura 5.20 que, al igual que en el caso anterior, la etapa de búsqueda de correspondencias es la más afectada. La precisión también cambia, siendo $err_R = 0{,}163$ y $err_t = 0{,}045$ en el primer caso de igual muestreo en el modelo y en la escena, y $err_R = 0{,}158$ y $err_t = 0{,}038$ en el segundo caso donde variamos el muestreo. Los dos casos otorgan desempeños similares tanto en el tiempo como en la precisión, siendo buenos los resultados (ver las Figuras 5.10, 5.11 y 5.12 para interpretar los valores).



(A) (B)

FIGURA 5.20: Tiempo promedio de cada etapa para el modelo *bunny* en las escenas de *dataset_2*, usando un submuestreo uniforme cada 2cm en el modelo y en la escena ($\blacktriangle$), y cada 1cm en el modelo y 2cm en la escena (B).

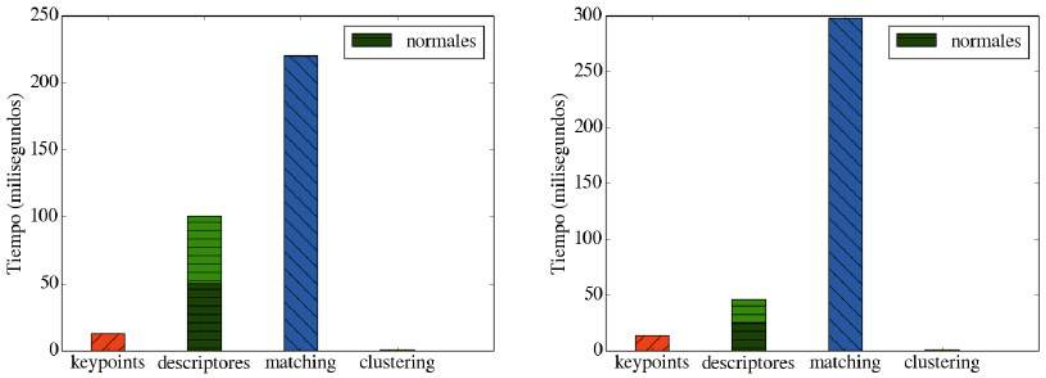
5.5.3. Elaboración de Descriptores

Una parte importante de la elaboración de descriptores es la estimación de normales, éstas proveen información de la forma local de la superficie. Es importante contar con una buena estimación de normales para poder construir descriptores informativos, por lo tanto no podemos dejar de lado demasiado la precisión de las normales. Como se vio en la Subsección 2.5.2 la estimación de normales involucra el cálculo de autovectores y autovalores, que es computacionalmente costoso, haciendo que el cálculo de las normales sea lento.

En general no hay una única forma correcta de saber qué radio elegir, un radio chico acelera la estimación, pero disminuye la cantidad de información incluida, también puede pasar que un radio muy grande deje detalles chicos de la superficie afuera.

En escenas del mundo real los objetos no existen en el vacío sino que hay mucha información de fondo que perjudica el proceso de reconocimiento, y al realizar un descriptor se pueden incorporar datos que no son del objeto en sí. Al reducir el radio se previene la incorporación de datos extra, pero también se pierde capacidad informativa del descriptor.

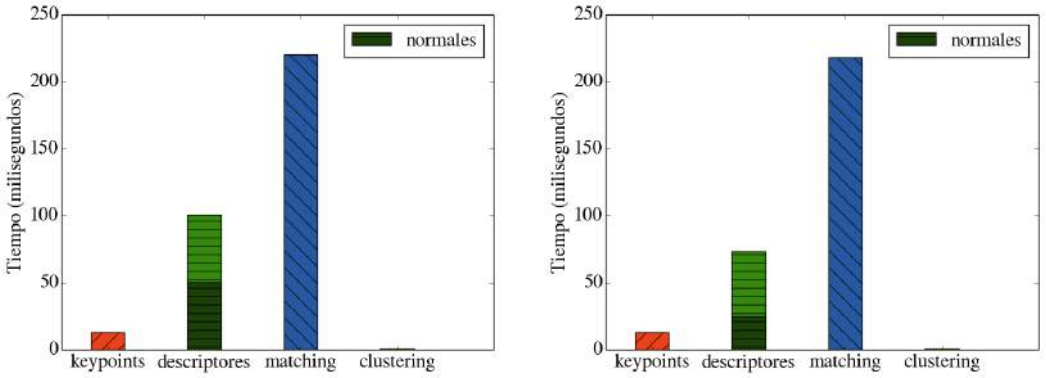
Volviendo al ejemplo del conejo, dejando fijos ahora los tamaños del submuestreo en 1cm tanto para el modelo como para la escena, probamos cambiar los radios usados por los descriptores y las normales. Consideramos primero un radio de 2cm tanto para las normales como para los descriptores, y de 1cm, obteniendo un sistema más lento al disminuir los radios y menos preciso. El tiempo total pasa de 335ms a 358ms, siendo poca la diferencia. Puede verse en la Figura 5.21, que el tiempo de elaboración de descriptores disminuye pero el de búsqueda de correspondencias aumenta. Este se debe a que al reducir el radio, los descriptores son menos informativos y es más dificil encontrar pares de puntos que estén asociados. Los valores de precisión son $err_R = 0{,}200$ y $err_t = 0{,}048$ en el primer caso de radios de 2cm, y $err_R = 1{,}180$ y $err_t = 0{,}383$ en el caso de radios de 1cm. En este caso esta reducción de tamaño no es buena, ya que la precisión empeora mucho, llegando a valores no aceptables, de más de 30 cm de traslación y más de 67 grados de rotación.



(A) (B)

FIGURA 5.21: Tiempo promedio de cada etapa para el modelo *bunny* en las escenas de *dataset_2*, usando radios de estimación de normales y generación de descriptores de 2cm (A), y de 1cm (B).

También podemos considerar tener distintos valores para estos radios, el radio de la normal puede disminuirse para acelerar la estimación. Consideramos ahora el caso de ambos radios en 2cm, comparado con el caso en el que se reduce el radio de la normal a 1cm. Se ve en la Figura 5.22, que al igual que antes el tiempo de la etapa de descriptores es menor, pero el de la etapa de matching es mayor, por las mismas razones. El tiempo pasa de 335ms a 305ms, dando una ligera mejora de velocidad. Los valores de precisión son $err_R = 0{,}200$ y $err_t = 0{,}048$ en el primer caso de radios de 2cm, y $err_R = 0{,}164$ y $err_t = 0{,}034$ en el segundo caso donde disminuimos el radio de la normal a 1cm. En este caso la precisión también mejora, siendo un buen cambio de parámetros.



(A) (B)

FIGURA 5.22: Tiempo promedio de cada etapa para el modelo *bunny* en las escenas de *dataset_2*, usando radios de estimación de normales y generación de descriptores de 2cm (A), y radios de estimación de normales de 1cm y de generación de descriptores de 2cm (B).

5.5.4. Búsqueda de Correspondencias

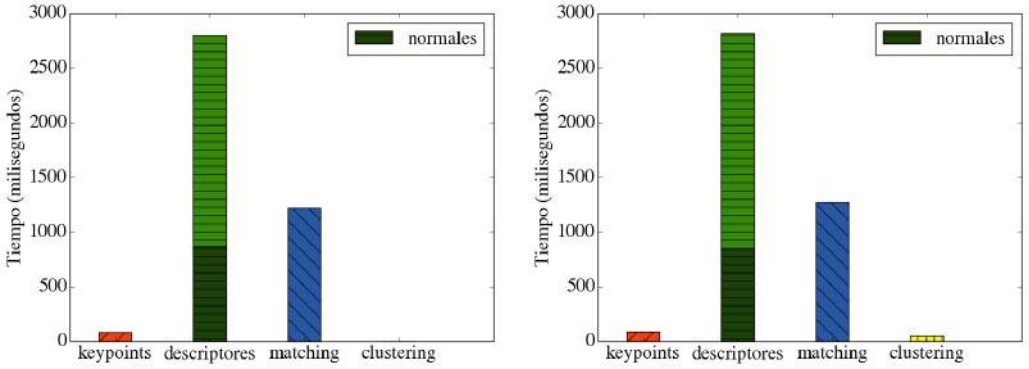
Usamos el método del cociente para determinar las correspondencias, como fue explicado en la Sección 3.5, con valores de umbral de 0.90 en ambas bases, determinado experimentalmente. Buscamos los dos descriptores más cercanos usando un árbol k-d ya que es más adecuado para las búsquedas de vecino más cercano que simplemente hacer una búsqueda lineal entre todos los descriptores.

Como vimos en la Subsección 4.2.3, para cada descriptor del modelo, buscamos los dos más cercanos en la escena para obtener las correspondencias, ya que sabemos que el modelo aparece una sola vez en la escena.

Agrupación de Correspondencias

Para agrupar las correspondencias, tenemos la opción de usar consistencia geométrica, votación de Hough o rechazo usando RANSAC con todas las correspondencias (ver Sección 4.2.3).

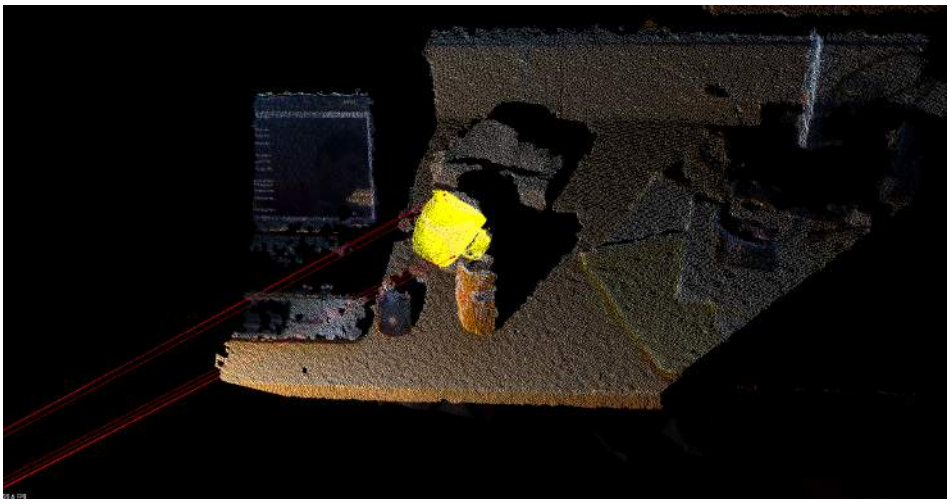
En experimentos iniciales, vimos que el uso de votación de Hough resultaba más lento y otorgaba peor precisión que usar consistencia geométrica, por lo cual no lo utilizamos. El rechazo con RANSAC también es más lento que consistencia geométrica pero otorga mejores resultados. Para la base de CVLab, en la cual nos importa el tiempo, usamos consistencia geométrica, con una cantidad mínima de 4 correspondencias por grupo, y una distancia máxima entre pares de correspondencias de 3 cm.



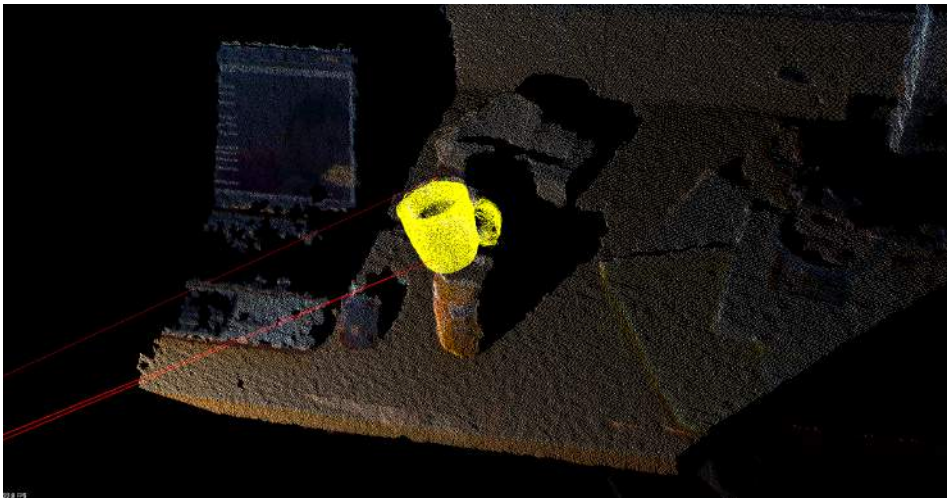
(A) (B)

FIGURA 5.23: Tiempo promedio de cada etapa para el modelo *coffee_mug_1* en las escenas de *desk_1*, usando consistencia geométrica con una distancia máxima de 1cm y un mínimo de 4 correspondencias por grupo (A), y usando rechazo de correspondencias con RANSAC con un umbral de inliers de 1cm (B).

En la base de RGB-D el agrupamiento con consistencia geométrica en muchos casos no encontraba una solución o encontraba una solución no tan buena como la dada por RANSAC. Para esta base decidimos entonces usar rechazo de correspondencias con RANSAC, porque otorgaba mayor precisión, y no incrementaba mucho el tiempo.



(A)



(B)

FIGURA 5.24: Resultado de reconocimiento para el modelo *coffee_mug_1* en una de las escenas de *desk_1*, usando consistencia geométrica con una distancia máxima de 1cm y un mínimo de 4 correspondencias por grupo (A), y usando rechazo de todas las correspondencias con RANSAC con un umbral de inliers de 1cm (B).

Por ejemplo, consideramos el modelo de la taza de café (ver Figura 5.35) en las escenas del escritorio (ver Figura 5.5), usando tanto agrupación con consistencia geométrica como rechazo con RANSAC, vemos en la Figura 5.23 que usando RANSAC el sistema demora un poco más, cambiando solamente el tiempo de la etapa de clustering, pasando de 0ms a 48 ms. El tiempo de clustering es muy poco en comparación a la etapa de elaboración de descriptores, por lo cual no afecta demasiado al rendimiento, en especial

si con solo un poco más de tiempo obtenemos una mejor precisión. Los valores de precisión para este caso son $70{,}47\,\%$ de superposición para consistencia geométrica, y $99{,}58\,\%$ para rechazo con RANSAC. Podemos ver en la Figura 5.24 cómo varía la precisión de la detección para estos dos casos.

Conclusión

Experimentamos instanciando el esquema de sistema (descripto en la Sección 4.3) para analizar cómo cambia el sistema final en cuanto al tiempo y la precisión otorgada. Encontramos que usar submuestreo uniforme es más rápido que usar el detector de Harris, y además resulta en más puntos. También usar un tamaño de submuestreo más chico otorga mejores precisiones a costa de mayores tiempos. El radio de descriptores y normales usado depende de la situación, radios chicos aceleran la generación de descriptores pero pueden hacer que la búsqueda de correspondencias sea más lenta. Sobre la agrupación, vimos que es más lento pero también más preciso usar RANSAC directamente sobre todas las correspondencias encontradas, en vez de primero agrupar consistentemente y después hacer RANSAC sobre cada cluster encontrado.

5.6. Evaluación del sistema usando la base de CVLab

En todos los casos se submuestrea uniformemente cada 1cm en el modelo, y cada 2cm en la escena, se estiman las normales usando un radio de 1 cm, y se elaboran los descriptores usando un radio de 2cm. Para comparar los descriptores, se usa el método del cociente visto en la Subsección 3.5.3, con un umbral de 0.90 para todos los descriptores. Por último, se agrupan las correspondencias con consistencia geométrica (ver Sección 4.2.3) con una distancia máxima de 3cm y tamaño mínimo de 4 correspondencias por grupo. Para la elaboración de descriptores solamente modificamos los parámetros de Spin Images, como fue visto en el Capítulo 3 siendo el máximo coseno entre ángulos de normales permitido 0.5, dando un ángulo de 60 grados, y la cantidad mínima de puntos de 1.

5.6.1. Precisión

En la Tabla 5.1 se pueden ver los errores de rotación y traslación promedio usando Spin Images, SHOT (notar que es sin color porque las nubes no poseen esta información), y FPFH (ver Capítulo 3), para evaluar las diferencias de precisión. Las filas corresponden

a distintos pares de modelo-escena, para cada uno de ellos se muestran los errores de rotación y traslación en distintas columnas, usando un descriptor particular.

modelo	escena	descriptor					
		Spin Images		**SHOT**		**FPFH**	
		err_R	err_t	err_R	err_t	err_R	err_t
Armadillo							
	Dataset1	0.142	0.034	**0.133**	**0.028**	0.189	0.041
	Dataset2	**0.192**	**0.039**	0.243	0.057	0.236	0.051
Buddha							
	Dataset1	0.174	0.034	**0.155**	**0.038**	0.167	0.036
	Dataset2	**0.257**	**0.063**	0.361	0.108	0.290	0.077
Bunny							
	Dataset1	0.130	0.031	**0.109**	**0.028**	0.127	0.030
	Dataset2	0.371	0.109	0.478	0.118	**0.139**	**0.035**
Chinese Dragon							
	Dataset1	**0.138**	**0.030**	0.170	0.042	0.166	0.042
	Dataset2	**0.253**	**0.052**	0.516	0.165	0.426	0.157
Dragon							
	Dataset1	**0.094**	**0.023**	**0.094**	**0.023**	0.109	0.027
	Dataset2	0.143	0.035	0.149	0.037	**0.129**	**0.029**
Statuette							
	Dataset1	**0.402**	**0.087**	0.509	0.140	0.868	0.218
	Dataset2	**0.407**	**0.075**	0.781	0.184	0.740	0.201

TABLA 5.1: Errores promedio de rotación (en radianes), y traslación (en metros), entre la transformación obtenida y la provista como ground truth. En negrita se muestra la mejor precisión en cada caso (menor valor), y en rojo los valores que superan los umbrales máximos de error aceptable.

Para el primer dataset, excepto para el modelo de la estatua, los valores de precisión son similares para todos los descriptores, esto puede deberse al alto detalle de las nubes, la densidad es alta y, tanto los descriptores como las normales son muy informativos de la superficie subyacente, facilitando el reconocimiento. Spin Images y SHOT son los que menores errores otorgan, y con FPFH se obtienen peores valores de precisión en comparación, aunque igualmente la diferencia sea muy chica entre el mejor y el peor caso, llegando a ser de 1.3 veces.

En el segundo dataset, que tiene considerablemente menos puntos que el primero y en el cual los efectos del ruido son más notorios; la variabilidad de precisiones dadas por los distintos descriptores fue más notoria. En este caso SHOT fue el que peor precisión otorgó, llegando a ser casi 4 veces peor que el mejor caso. Con Spin Images y FPFH se obtuvieron los mejores valores, con Spin Images en más casos que con FPFH, dando mejoras de hasta 1.8 veces frente a FPFH. Por otro lado, al usar el modelo del conejo, solamente con FPFH se pudieron lograr buenos resultados, casi 3 veces mejores que los dados por Spin Images, y más de 3.4 veces mejores que usando SHOT.

Finalmente, con el modelo *Statuette* se obtuvieron los peores valores de precisión para los dos datasets y con todos los descriptores, siendo Spin Images el único que logra obtener resultados aceptables. Esto puede deberse a la complejidad y detalle de dicho modelo en comparación a los otros, como puede verse en la Figura 5.1F. En comparación, con el modelo del dragon se obtuvieron los mejores valores en todos los casos, que puede deberse a la alta densidad de la nube modelo y la cantidad de puntos muestreados en ésta, como se ve en la Tabla 5.2 que es mayor que para los otros modelos utilizados.

5.6.2. Tiempos

A continuación se presentan los gráficos con los tiempos promedio de cada etapa del sistema (ver Sección 4.3), obtención de keypoints, elaboración de descriptores, búsqueda de correspondencias, y agrupación de correspondencias.

Por simplicidad se presenta un subconjunto de todos los gráficos obtenidos, solo los correspondientes a los modelos *Armadillo* y *Statuette*, ya que para cada descriptor los gráficos obtenidos con distintos pares modelo-escena son muy similares, y no consideramos significativo analizar cada caso por separado. Por esta razón consideramos mostrar los gráficos para estos dos modelos, usando los dos datasets, puesto que entre datasets los tiempos sí se modifican de manera relevante.

Primero se muestran los gráficos correspondientes a los tiempos de los modelos, y después los de las escenas, y en las Tablas 5.2 y 5.3, se puede ver la cantidad de puntos y keypoints seleccionados de cada uno de los modelos y escenas usados respectivamente, para dar una idea de la cantidad de descriptores que se elaboran en cada caso, y el tiempo estimado que tarda en generarse un solo descriptor, para Spin Images, CSHOT y FPFH.

			Tiempo de un descriptor (ms)		
Modelo	Puntos	Keypoints	**Spin Images**	**SHOT**	**FPFH**
Armadillo	36.100	525	0.20	0.45	6.65
Buddha	32.300	640	0.30	0.35	3.10
Bunny	35.950	760	0.10	0.20	2.65
Chinese Dragon	31.100	470	0.35	0.50	5.95
Dragon	100.200	925	0.50	0.85	14.90
Statuette	40.200	540	0.30	0.50	6.50

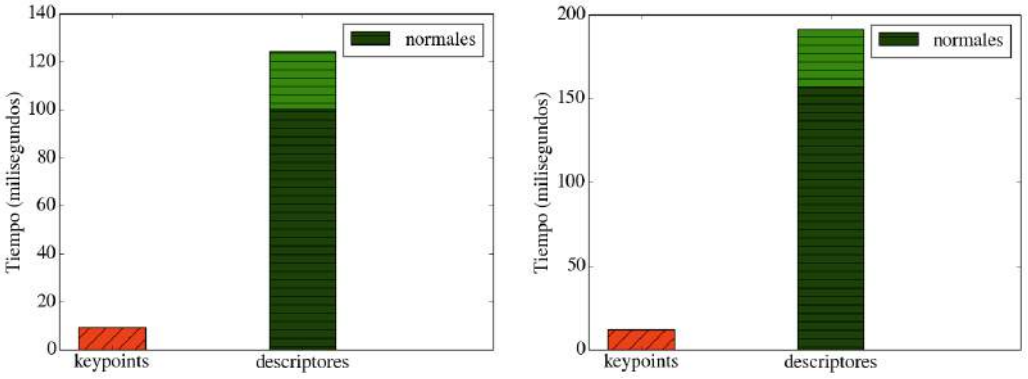
TABLA 5.2: Aproximaciones de cantidades de puntos y keypoints seleccionados, y tiempos de generación del descriptor para un punto en milisegundos, para los modelos.

			Tiempo de un descriptor (ms)		
Escenas	Puntos	Keypoints	**Spin Images**	**SHOT**	**FPFH**
Dataset1	175.000	585	1.35	2.05	42.75
Dataset2	22.1000	580	0.10	0.12	0.85

TABLA 5.3: Aproximaciones de cantidades de puntos y keypoints seleccionados, y tiempos de generación del descriptor para un punto en milisegundos, para las escenas de cada dataset.

Para los modelos, la mayor parte del tiempo se consume en la elaboración de descriptores, dentro de este tiempo, el porcentaje consumido por la estimación de las normales varía según el descriptor (el tiempo que se tarda en estimar las normales para un mismo modelo es el mismo en todos los casos, pero como el tiempo de elaboración de descriptores cambia, el porcentaje el distinto).

Para Spin Images y SHOT, este porcentaje es mayor al $50\,\%$, como puede verse en las figuras 5.25 y 5.26. El caso de FPFH es el que más demora en elaborar descriptores, tardando alrededor de 4 segundo como se ve en la Figura 5.27, cuando para los otros descriptores se tarda menos de 300 milisegundos.



(A) (B)

FIGURA 5.25: Tiempo promedio de cada etapa para los modelos *armadillo* (A) y *statuette* (B) usando *Spin Images*

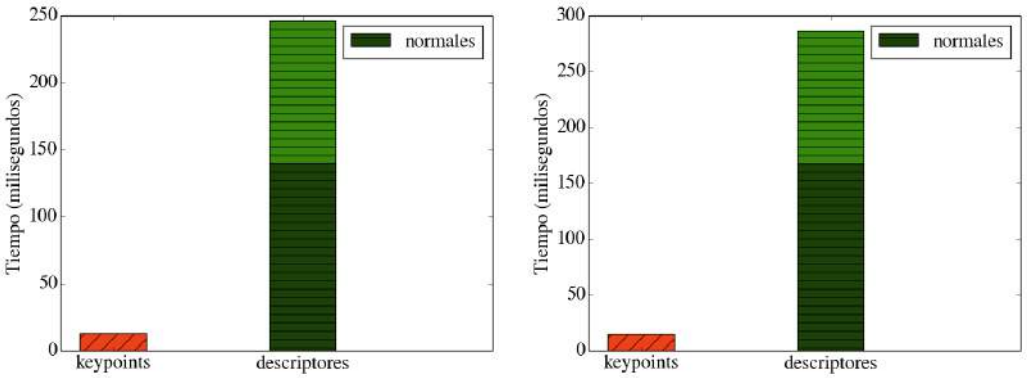


FIGURA 5.26: Tiempo promedio de cada etapa para los modelos *armadillo* (A) y *statuette* (B) usando *SHOT*

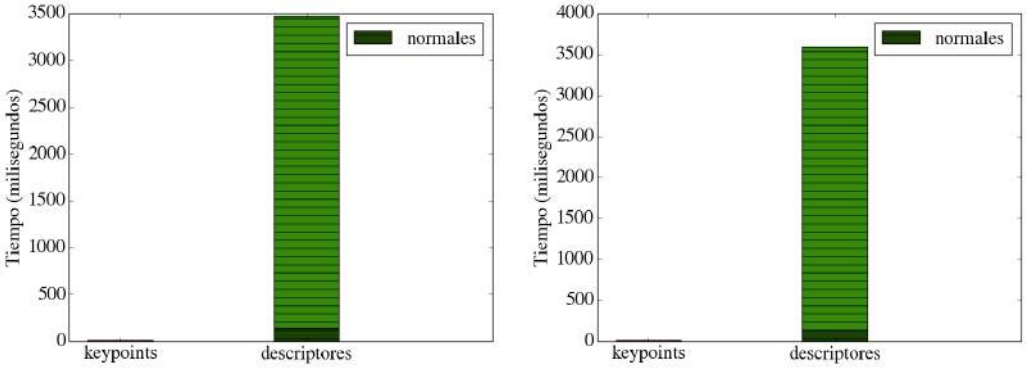


FIGURA 5.27: Tiempo promedio de cada etapa para los modelos *armadillo* (A) y *statuette* (B) usando *FPFH*

Para las escenas, podemos separar los casos de los dos datasets, el primero tiene una alta densidad de puntos, ofreciendo un detalle fino, mientras que el segundo tiene una densidad muy gruesa en la cual muchos detalles se pierden, y el ruido influye más en el proceso de reconocimiento, haciéndolo más dificil. Además hay una gran diferencia en la cantidad de puntos, haciendo que el sistema tarde más en reconocer objetos en las escenas del dataset 1 que en las del 2. Esta diferencia se ve más marcada en la obtención de descriptores.

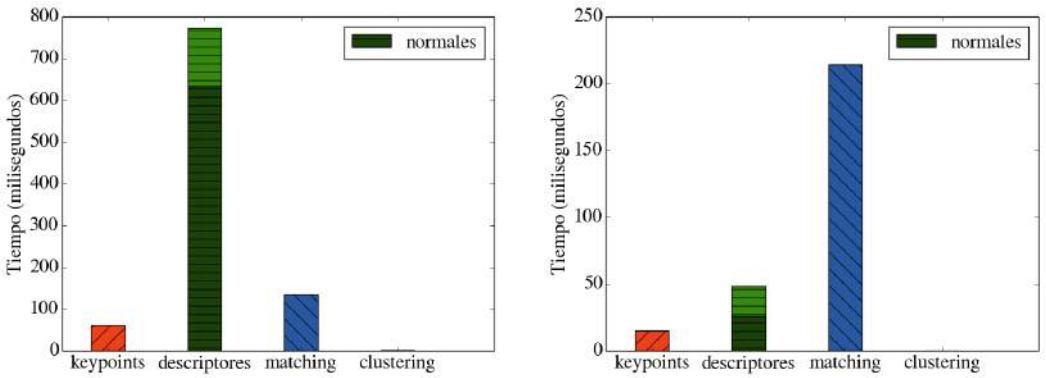
La etapa de obtención de keypoints tarda lo mismo sin importar el descriptor porque solo depende de la nube utilizada. Usando Spin Images y SHOT también ocurre que el porcentaje de estimación de las normales dentro de la generación de descriptores es alto, mayor a la mitad, como puede verse en las Figuras 5.28 y 5.29 para Spin Images,

y 5.30 y 5.31, para SHOT. Con FPFH no ocurre porque nuevamente la elaboración de descriptores es lenta, como se ve reflejado en las Figuras 5.32 y 5.33.

El uso de los descriptores SHOT y Spin Images otorga tiempos parecidos, siendo Spin Images más rápido, especialmente al generar descriptores, como puede verse en la Tabla 5.3, siendo más marcada la diferencia para el primer dataset.

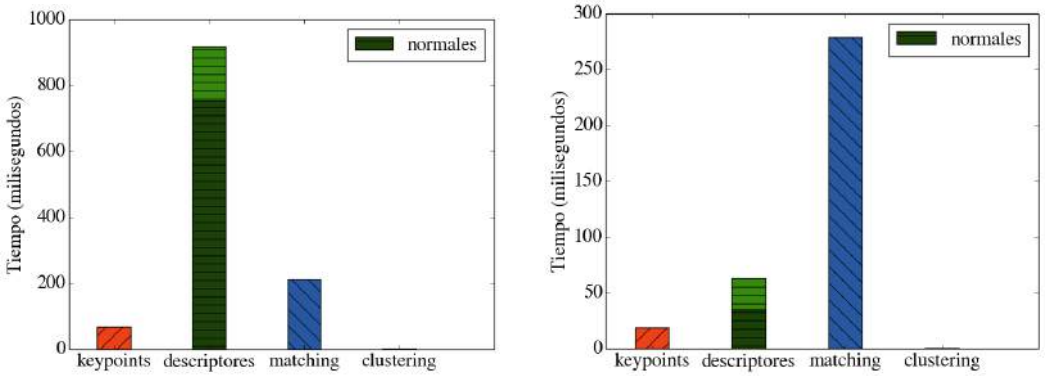
Para el segundo dataset la etapa de búsqueda de correspondencias demora más que para el primero, porque los descriptores tienen menos información de la superficie, lo que hace que la búsqueda de puntos similares lleve más tiempo. Esta diferencia de información se debe al cambio de la densidad de las nubes de las escenas.

Usar FPFH es casi 30 veces más lento que usar Spin Images o SHOT para el primer dataset, mientras que para el segundo dataset demora más de 4 veces más. La generación de descriptores para FPFH es más lenta, pero la etapa de matching es más rápida, esto puede deberse al hecho de que el vector que se utiliza para representar al descriptor FPFH tenga muchas menos posiciones que los que se utilizan para Spin Images y SHOT (33 posiciones contra 153 y 352 respectivamente), que hace que la búsqueda de vecino más cercano se acelere. Igualmente la diferencia en el tiempo de descriptores es muy grande como para compensar la ganancia en la búsqueda de vecinos más cercanos.



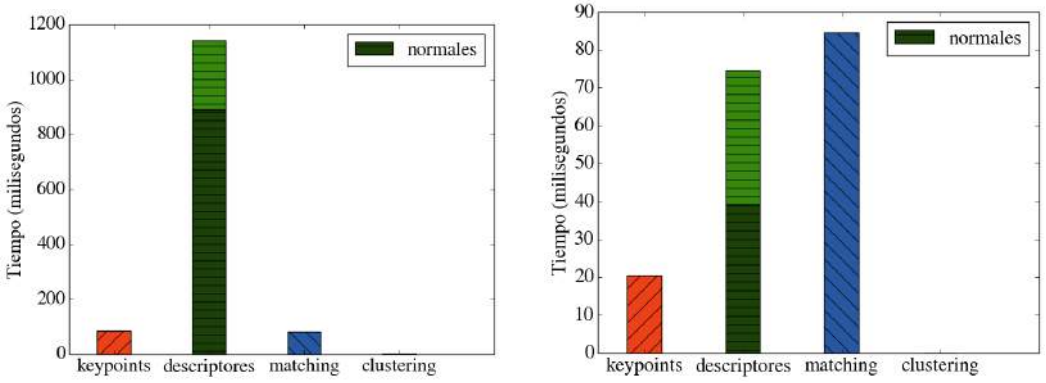
(A) (B)

FIGURA 5.28: Tiempo promedio de cada etapa para el modelo *armadillo* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *Spin Images*



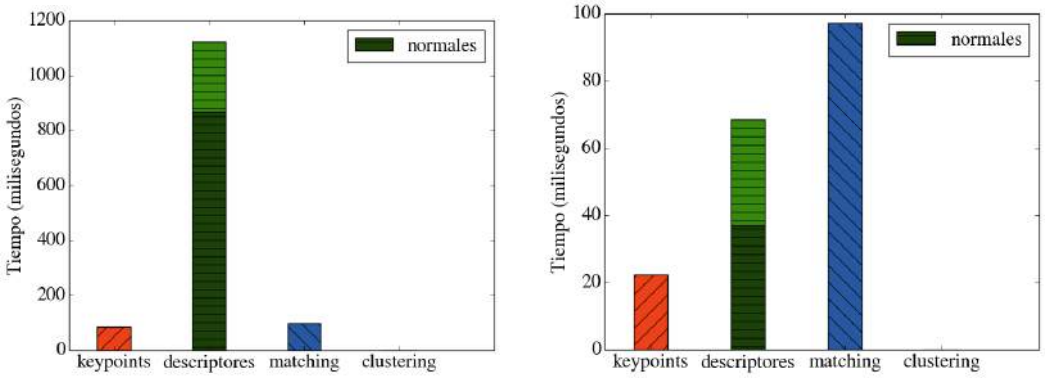
(A) (B)

FIGURA 5.29: Tiempo promedio de cada etapa para el modelo *statuette* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *Spin Images*



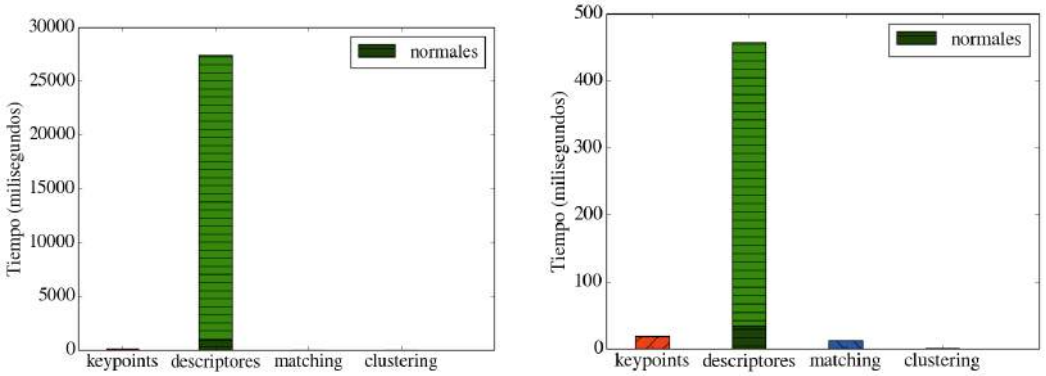
(A) (B)

FIGURA 5.30: Tiempo promedio de cada etapa para el modelo *armadillo* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *SHOT*



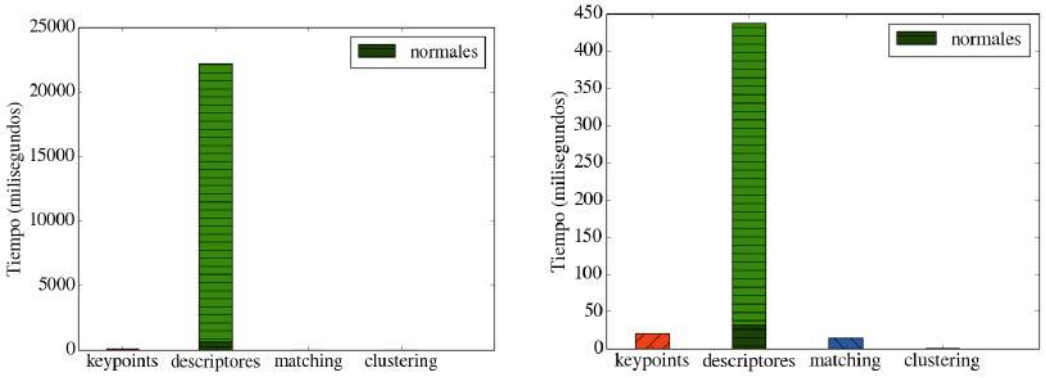
(A) (B)

FIGURA 5.31: Tiempo promedio de cada etapa para el modelo *statuette* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *SHOT*



(A) (B)

FIGURA 5.32: Tiempo promedio de cada etapa para el modelo *armadillo* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *FPFH*



(A) (B)

FIGURA 5.33: Tiempo promedio de cada etapa para el modelo *statuette* en las escenas de *dataset_1* (A) y *dataset_2* (B) usando *FPFH*

Como varía no solo la cantidad sino también la densidad, la estimación de normales para el primer dataset consume un porcentaje muy alto de la elaboración de descriptores, haciendo que todo el sistema sea más lento.

Se ve en la Tabla 5.3 que para las escenas, el promedio de keypoints elegidos es casi el mismo, haciendo que la cantidad de descriptores a elaborar sea igual, pero sin embargo varía mucho la cantidad de puntos originales (en el dataset 1 hay aproximadamente 8 veces más puntos), lo que hace que la estimación de normales y el cálculo de descriptores demoren más, ya que utilizan todos los puntos en un entorno. Esto es porque la estimación de normales se hace en función a un radio, y no una cantidad, en un mismo radio la cantidad de puntos presentes varía mucho según la densidad de la nube.

El hecho de que la cantidad de puntos elegidos, y por lo tanto de descriptores varíe, hace que la distribución relativa de tiempos cambie. El tiempo de la etapa de matching depende de esta cantidad, y el tiempo de clustering depende de la cantidad de correspondencias encontradas, que a su vez varía según el descriptor. Sin embargo, el tiempo absoluto de esta etapa es muy parecido para los casos de los dos datasets, pero relativo a las otras etapas muy distinto.

Soluciones posibles al problema del tiempo de estimación de normales podrían tener en cuenta descriptores que no utilizaran normales, o podrían calcular las normales de antemano, en una etapa de preprocesamiento anterior al sistema, y recibir como entrada nubes con normales.

5.6.3. Rendimiento Computacional

En el caso del primer dataset la diferencia se ve más marcadamente en los tiempos, con FPFH el sistema demora casi 30 veces más, y la precisión no es 30 veces mejor sino que hasta es peor que con los otros descriptores. En este caso Spin Images y SHOT son una mejor opción.

Entre Spin Images y SHOT, las precisiones son similares, siendo mejor uno que el otro según el modelo utilizado, con diferencias no mayores a 1.3 veces uno del otro. En cuanto a los tiempos, SHOT demora casi 1.5 veces más que Spin Images, haciendo que la elección de Spin Images sea mejor.

Para el segundo dataset, la diferencia es más marcada en la precisión, donde hay mucha variabilidad, y ahora SHOT entrega las peores precisiones. FPFH resulta más preciso que Spin Images en solo algunos casos, pero es siempre más lento, llegando a tardar más de 4 veces más que usando Spin Images. Spin Images resulta más rápido en todos los casos y en particular para el modelo de la estatua es el único que otorga resultados aceptables, siendo una mejor elección.

5.7. Evaluación del sistema usando la base RGB-D

Para esta base usamos el descriptor CSHOT en vez de SHOT porque las nubes tienen información de color. Antes de presentar las precisiones y tiempos obtenidos, discutiremos el preprocesamiento de la imágenes de la base, y los modelos utilizados.

5.7.1. Preprocesamiento

La base de datos provee la informacion de los modelos y las escenas como imágenes RGBD, a partir de las cuales obtenemos nubes de puntos ya que es el formato usado por el sistema implementado (Sección 4.3). Conseguimos las nubes a partir de las imágenes RGBD como describimos en la Sección 2.4.

Generación de modelos 3D para los objetos

Las nubes de los modelos se corresponden con vistas parciales de los objetos a distintos ángulos y no al modelo tridimensional completo del objeto. Para facilitar el reconocimiento construimos estos modelos 3D a partir de sus vistas.

Elaboramos un procedimiento que, dado un conjunto de vistas de una instancia de un modelo, devuelve la nube correspondiente al objeto tridimensional de esa instancia. Alineamos las vistas de a pares, obteniendo para cada par una nube con los puntos de ambas de forma alineada, que agregamos a la nube resultado.

Para alinear cada par, usamos *ICP* (ver Subsección 4.2.5) con una transformación inicial elegida experimentalmente. En algunos casos se refinaron las alineaciones de forma manual. Usamos un subconjunto de las vistas (1 cada 16) para cada instancia para no introducir demasiado ruido a la nube final. Por ejemplo, usando las vistas de la taza mostradas en la Figura 5.34, obtuvimos el modelo tridimensional visible en la Figura 5.35.

FIGURA 5.34: Vistas del modelo *coffee_mug*, instancia 1

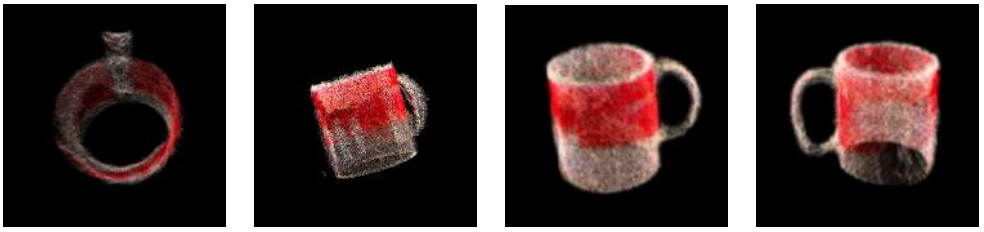


FIGURA 5.35: Modelo tridimensional *coffee_mug_1* construido a partir de las diferentes vistas presentadas en la Figura 5.34

Simplificación de nubes

Como los datos de la base provienen de capturas usando Kinect, las nubes son ruidosas y las medidas de distancia no son exactas. Por eso, decidimos desarrollar un método de preprocesamiento que suaviza las nubes para facilitar el reconocimiento. Se utiliza un método de remuestreo, que trata de recrear las partes faltantes de la superficie dada por los puntos mediante interpolación polinómica usando *Moving Least Squares (MLS)* [LS81]. En la Figura 5.36 puede verse un ejemplo de simplificación de superficie y reducción de ruido para el modelo de una taza.

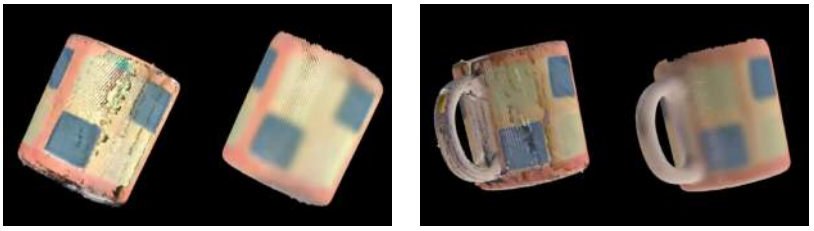
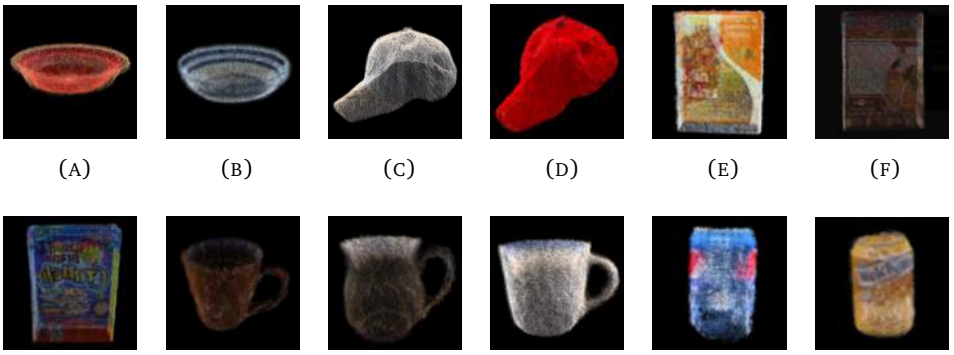


FIGURA 5.36: Modelo de una taza antes (izquierda) y después (derecha) de haber sido simplificado.

5.7.2. Modelos utilizados

Aunque las escenas contienen muchos modelos, solo para unos pocos de ellos existe información de ground truth. Como en las pruebas realizadas hacemos uso de esta información, solamente usamos estos modelos, que son los pertenecientes a las categorías de Bowl, Cap, Cereal Box, Coffee Mug y Soda Can. Algunos ejemplos se muestran en la Figura 5.37.



(A) (B) (C) (D) (E) (F)

(G) (H) (I) (J) (K) (L)

FIGURA 5.37: Algunos de los modelos tridimensionales utilizados, de izquierda a derecha y de arriba a abajo son, *bowl_1* (A), *bowl_5* (B), *cap_1* (C), *cap_4* (D), *cereal_box_1* (E), *cereal_box_2* (F), *cereal_box_4* (G), *coffee_mug_2* (H), *coffee_mug_3* (I), *coffee_mug_4* (J), *soda_can_1* (K), y *soda_can_6* (L).

5.7.3. Precisión

En todos los casos se submuestrea uniformemente cada 1cm tanto en el modelo como en la escena, se estiman las normales usando un radio de 2cm y se elaboran los descriptores usando un radio de 5cm. Para obtener las correspondencias se usa el método del cociente descripto en la Subsección 3.5.3 con un umbral de 0.90 para todos los descriptores. Por último se rechazan correspondencias utilizando RANSAC (ver Sección 4.2.3) con todas las correspondencias con un umbral de inliers de 1cm.

modelo	escena	descriptor					
		Spin Images		**CSHOT**		**FPFH**	
		precisión	aciertos	precisión	aciertos	precisión	aciertos
coffee_mug_5							
	desk_1	**44.347**	**77.631**	40.786	63.157	38.176	17.105
	kitchen_small_1	20.107	35.714	**33.777**	**42.857**	-	-
soda_can_6							
	desk_1	55.706	**73.529**	**64.292**	42.647	80.024	14.705
	kitchen_small_1	**33.968**	**33.333**	78.875	9.523	-	-
cap_4							
	desk_1	49.050	**82.926**	**63.707**	63.414	58.536	70.731
	table_1	**25.809**	**32.608**	12.973	31.521	-	-
	table_small_2	30.306	**91.578**	**47.870**	34.736	-	-
bowl_3							
	desk_2	32.103	42.727	53.561	47.272	**54.272**	**50.909**
	meeting_small_1	42.360	23.943	70.153	24.647	-	-
bowl_4							
	desk_3	31.208	85.517	38.226	**88.275**	**44.779**	53.793
	table_small_1	**41.317**	35.384	36.939	**51.538**	-	-
	kitchen_small_1	28.064	**65.625**	**33.682**	62.5	-	-
cereal_box_1							
	desk_3	33.262	**48.0**	**55.373**	44.666	-	-
	table_small_1	31.266	**71.895**	**35.842**	49.019	-	-

TABLA 5.4: Superposición promedio (porcentaje) entre la bounding box obtenida y la provista como ground truth. En negrita se muestran los mayores valores de superposición y de aciertos, y en rojo los valores que están por debajo de los umbrales definidos. Si para un caso el porcentaje de aciertos es menor al 30 %, no consideramos el valor de superposición, aunque sea mayor que el dado por otros descriptores.

En las tablas 5.4 y 5.5 se pueden ver los porcentajes de superposición promedio usando distintos descriptores así como también los porcentajes de aciertos. Consideramos aciertos a los casos en los cuales el porcentaje de superposición fue mayor o igual al 5 %, el porcentaje de aciertos será entonces la cantidad de cuadros con aciertos sobre la cantidad total de cuadros en los cuales aparece un objeto. El porcentaje de superposición promedio se realizó solamente sobre los aciertos. Las filas de las tablas corresponden a distintos pares de modelo-escena, para cada uno de ellos se muestra el porcentaje de superposición (precisión) y de aciertos, usando un descriptor particular.

Consideramos como resultado aceptable que el porcentaje de aciertos sea mayor a 30 %, y que el de superposición sea mayor a 20 %, basándonos en los porcentajes de casos exitosos de reconocimientos analizados, y lo visto en la Sección 5.5. Si el porcentaje de aciertos es muy bajo, ya no consideramos el valor de superposición reportado. El valor de superposición sirve como indicador de la calidad del reconocimiento, y si el porcentaje de aciertos es bajo entonces el sistema no será bueno, es decir que con ese descriptor el sistema no será bueno detectando objetos pero si lo detecta lo puede hacer bien, con un porcentaje de superposición alto.

modelo	escena	descriptor			
		Spin Images		**CSHOT**	
		precisión	aciertos	precisión	aciertos
cap_1					
	table_1	**25.951**	**82.882**	66.114	15.315
	kitchen_small_1	**33.291**	**92.0**	58.994	16.0
	meeting_small_1	**40.531**	**63.225**	56.200	18.709
cereal_box_4					
	table_1	**39.742**	56.140	33.467	**67.543**
	table_small_1	21.692	27.835	**36.664**	**84.536**
coffee_mug_1					
	table_small_1	**55.387**	**30.215**	41.051	27.338
coffee_mug_4					
	table_1	51.803	**39.560**	**79.490**	32.967
soda_can_3					
	table_small_1	42.353	27.407	15.933	20.0
soda_can_1					
	table_small_2	49.308	29.702	79.394	6.9306
	kitchen_small_1	**23.058**	**31.578**	7.894	7.894
cereal_box_2					
	kitchen_small_1	29.298	63.265	**60.530**	**83.673**
cap_3					
	meeting_small_1	39.892	**88.775**	**43.629**	40.816

TABLA 5.5: Superposición promedio (porcentaje) entre la bounding box obtenida y la provista como ground truth. En negrita se muestran los mayores valores de superposición y de aciertos, y en rojo los valores que están por debajo de los umbrales definidos. Si para un caso el porcentaje de aciertos es menor al 30 %, no consideramos el valor de superposición, aunque sea mayor que el dado por otros descriptores.

Usando FPFH solo obtuvimos algunas medidas de tiempo, porque resultó ser excesivamente lento, como se verá más adelante. Funciona mejor que Spin Images y CSHOT solo en un solo caso, usando el modelo bowl_3 en las escenas de desk_2, en el resto el porcentaje de superposición es bueno pero el de aciertos suele ser menor que el dado por los otros descriptores, llegando en dos casos a estar por debajo del umbral definido. Como demora mucho y las mejoras de precisión no fueron evidentes, decidimos descartar este descriptor como opción para esta base.

Al usar Spin Images y CSHOT, las precisiones fueron comparables, fallando SHOT en más casos, es decir que el porcentaje de aciertos fue menor a 30 %. Algunas veces la precisión dada por CSHOT fue mejor que la dada por Spin Images pero con menor cantidad de aciertos, por ejemplo en el caso de soda_can_6 con las escenas de desk_1, donde la precisión de CSHOT es mejor (1.15 veces) pero el porcentaje de aciertos es peor (1.7 veces). En casos como estos podemos decir que el comportamiento de Spin Images fue mejor porque la diferencia de aciertos fue más grande que la de la precisión.

Vimos que, la mayoría de los modelos de la base tienen poca información de color, siendo de un mismo color o a lo sumo dos. El caso en el cual Spin Images otorga la mayor mejora de precisión respecto a CSHOT es usando la primera instancia del modelo de la gorra, que tiene un solo color, como se puede observar en la Figura 5.38A; y por otro lado, el uso de CSHOT resulta en una mejora respecto a Spin Images al usar la cuarta instancia del modelo de la caja de cereal, que contiene información significativa y puede verse en la Figura 5.38B.

(A) (B)

FIGURA 5.38: Dos instancias de los modelos de la base, el primero (A), *cap_1* teniendo poca textura, y el segundo (B), *cereal_box_4* teniendo mucha información de color.

Podemos decir entonces que, para estos modelos, la forma es más importante que el color para el reconocimiento, y que además en esta situación en la cual el color no es tan importante, el uso de Spin Images resulta en mayor precisión que el de CSHOT. Esto da la idea de que Spin Images describe de mejor manera la forma para estos objetos.

Hubieron casos donde ni Spin Images ni CSHOT otorgaron resultados aceptables, estos son usando el modelo *bowl_3* en las escenas de *meeting_small_1*, y al usar los modelos de las latas de gaseosa *soda_can_1* y *soda_can_3*. Esto puede deberse a que las latas de gaseosa son objetos muy chicos sin mucha información de forma, que pueden ser facilmente confundidos con el fondo o con otros objetos, y en las escenas de *meeting_small_1* existen dos instancias de los modelos Bowl, complicando el reconocimiento.

5.7.4. Tiempos

A continuación se presentan los gráficos con los tiempos promedio de cada etapa del sistema (ver Sección 4.3), obtención de keypoints, elaboración de descriptores, búsqueda de correspondencias, y agrupación de correspondencias.

Por simplicidad se presenta un subconjunto de todos los gráficos obtenidos, solo los correspondientes a los modelos *bowl_3*, *cap_4*, *coffee_mug_5* y *soda_can_6* para los modelos, ya que para cada descriptor los gráficos obtenidos con distintos pares modelo-escena son muy similares, es decir los tiempos relativos de cada etapa respecto a las demás se mantienen semejantes, y no consideramos significativo analizar cada caso por separado. El resto de los gráficos son similares a los presentados.

Primero se muestran los gráficos correspondientes a los tiempos de los modelos, y después los de las escenas, y en las Tablas 5.6 y 5.7, se puede ver la cantidad de puntos y keypoints seleccionados de cada uno de los modelos y escenas usados respectivamente, para dar una idea de la cantidad de descriptores que se elaboran en cada caso, y el tiempo estimado que tarda en generarse un solo descriptor, para Spin Images, CSHOT y FPFH.

			Tiempo de un descriptor (ms)		
Modelo	Puntos	Keypoints	**Spin Images**	**CSHOT**	**FPFH**
bowl_3	82.400	395	6.3	8.8	37.9
bowl_4	86.000	409	7.3	9.7	32.9
cap_1	174.300	1.100	4.1	5.4	-
cap_3	196.600	1.300	5.2	6.5	-
cap_4	167.300	1.080	4.6	6.4	20.8
cereal_box_1	276.500	2.400	3.5	5.4	-
cereal_box_2	185.000	2.200	1.8	2.6	-
cereal_box_4	278.300	2.100	3.8	5.2	-
coffee_mug_1	58.500	450	3.5	3.7	-
coffee_mug_4	58.000	470	2.9	3.6	-
coffee_mug_5	65.200	570	2.8	4.3	136.8
coffee_mug_6	65.900	580	2.9	4.3	-
soda_can_1	58.800	380	3.9	4.4	-
soda_can_3	34.000	400	3.2	3.7	-
soda_can_5	68.400	390	4.1	6.4	-
soda_can_6	69.600	400	4.2	5.2	225

TABLA 5.6: Aproximaciones de cantidades de puntos y keypoints seleccionados, y tiempos de generación del descriptor para un punto en milisegundos, para los modelos.

			Tiempo de un descriptor (ms)		
Escenas	Puntos	Keypoints	**Spin Images**	**CSHOT**	**FPFH**
desk_1	230.000	19.000	0.15	0.30	3.50
desk_2	270.000	16.000	0.10	0.30	3.05
desk_3	270.000	16.000	0.20	0.30	3.40
kitchen_small_1	270.000	15.000	0.20	0.35	-
meeting_small_1	270.000	28.000	0.10	0.20	-
table_1	280.000	22.000	0.10	0.25	-
table_small_1	260.000	27.000	0.05	0.15	-
table_small_2	260.000	27.000	0.10	0.20	-

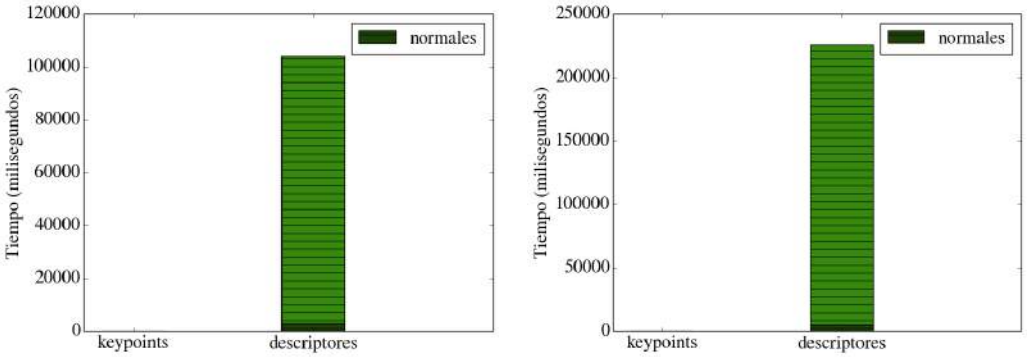
TABLA 5.7: Aproximaciones de cantidades de puntos y keypoints seleccionados, y tiempos de generación del descriptor para un punto en milisegundos, para los cuadros de las escenas.

Para los modelos, la mayor parte del tiempo se consume en la elaboración de descriptores, siendo el tiempo de obtención de keypoints despreciable frente a éste. Puede verse

en las Figuras 5.39 y 5.40 que usando FPFH el sistema demora más de 1 minuto (60.000 milisegundos) para todos los modelos, en obtener los descriptores, que es mucho tiempo, en especial en comparación con los otros descriptores, el sistema usando FPFH es más de 30 veces más lento que usando Spin Images, y más de 25 veces más lento que usando CSHOT, pudiendo verse también la diferencia entre el tiempo de generación de descriptores en la Tabla 5.6.

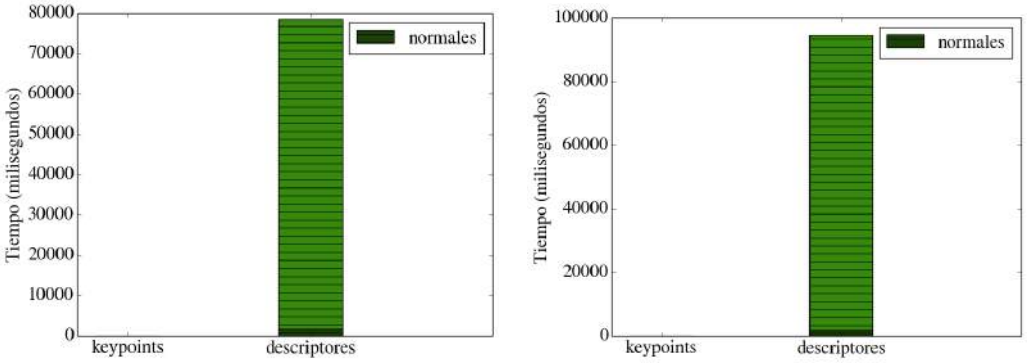
Usando Spin Images y CSHOT, el sistema demora entre 2 y 8 segundos dependiendo del modelo, lo que puede verse en las Figuras 5.41, 5.42, 5.43, 5.44. Entre los tiempos dados por usar Spin Images y CSHOT hay una diferencia chica, siendo CSHOT ligeramente más lento, aproximadamente 1.5 veces.

Como en la base de CVLab, ocurre que el porcentaje de la estimación de normales dentro de la etapa de la elaboración de descriptores es alto, excepto para FPFH que demora mucho. Para Spin Images y CSHOT este porcentaje es mayor al 50 %, y cambia según el modelo porque varía la densidad de puntos, por ejemplo para cap_4 ese porcentaje es más chico que para bowl_3, porque la densidad de puntos es mayor, la cantidad de keypoints y puntos totales mostrados en la Tabla 5.6 pueden dar una idea de la densidad.



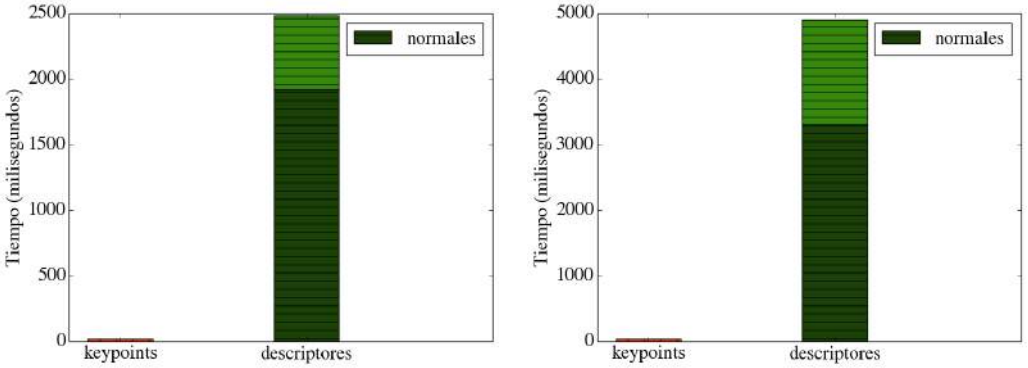
(A) (B)

FIGURA 5.39: Tiempo promedio de cada etapa para los modelos *bowl_3* (A) y *cap_4* (B) usando *FPFH*



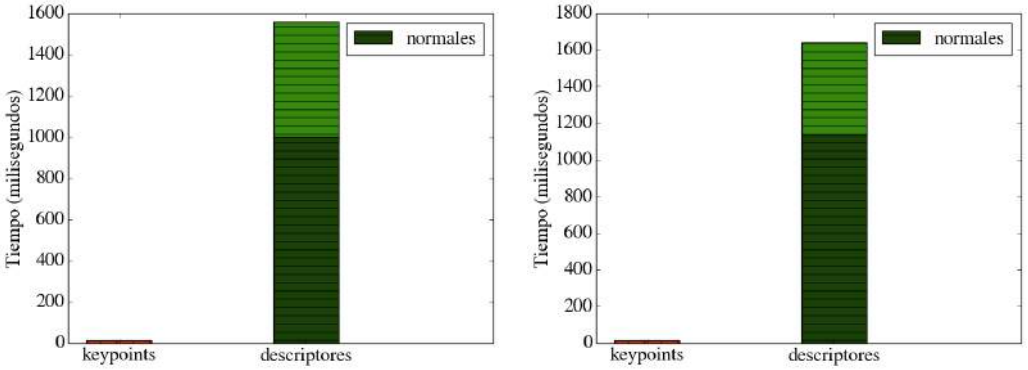
(A) (B)

FIGURA 5.40: Tiempo promedio de cada etapa para los modelos *coffee_mug_5* (A) y *soda_can_6* (B) usando *FPFH*



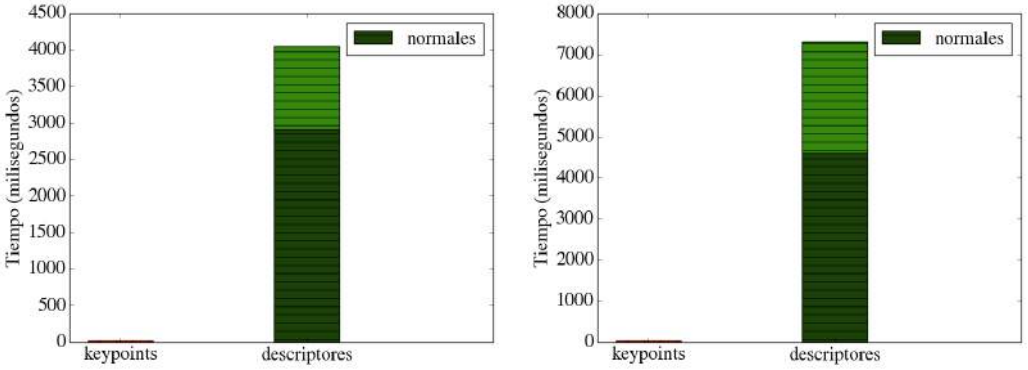
(A) (B)

FIGURA 5.41: Tiempo promedio de cada etapa para los modelos *bowl_3* (A) y *cap_4* (B) usando *Spin Images*



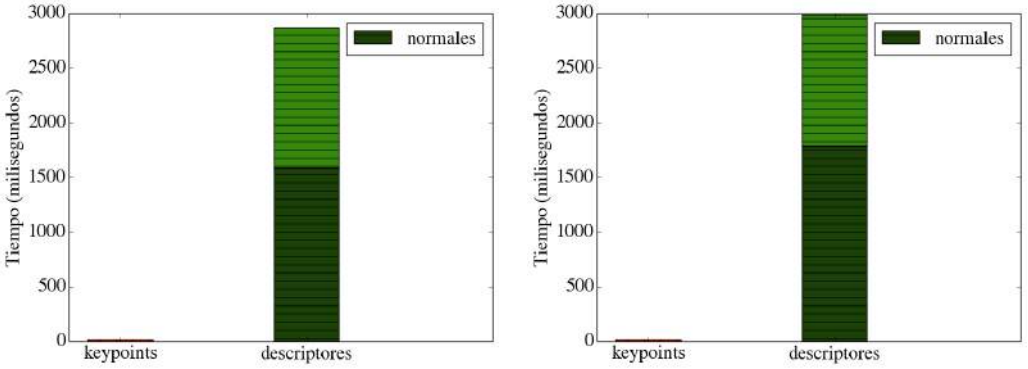
(A) (B)

FIGURA 5.42: Tiempo promedio de cada etapa para los modelos *coffee_mug_5* (A) y *soda_can_6* (B) usando *Spin Images*



(A) (B)

FIGURA 5.43: Tiempo promedio de cada etapa para los modelos *bowl_3* ($\blacktriangle$) y *cap_4* (B) usando *SHOT*



(A) (B)

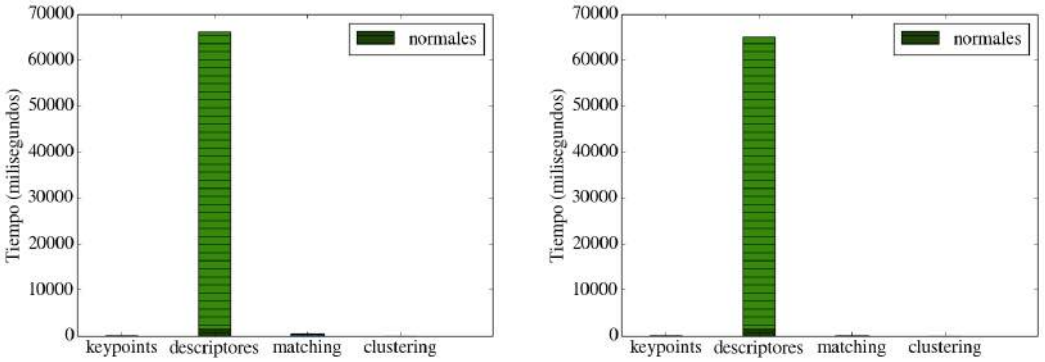
FIGURA 5.44: Tiempo promedio de cada etapa para los modelos *coffee_mug_5* ($\blacktriangle$) y *soda_can_6* (B) usando *SHOT*

Para las escenas, mostramos solamente los gráficos correspondientes a los modelos cap_4 y coffee_mug_5, por los mismos motivos de antes. En cada escena la cantidad de objetos es distinta, y algunas tienen varias instancias del mismo objeto, haciendo que la detección sea más dificil, por ejemplo en la escena *table_1* hay dos instancias del modelo de la gorra, y del a taza de café, y en la escena *kitchen_small_1* hay dos instancias del modelo de la lata de gaseosa.

Para FPFH nuevamente ocurre que la elaboración de descriptores es muy lenta, solo mostramos los tiempos modelos en las escenas de desk_1, que se pueden ver en la Figura 5.45 haciendo que el sistema sea más de 14 veces más lento que usando Spin Images (ver Figuras 5.46 y 5.47); y más de 5 veces más lento que usando SHOT (ver Figuras 5.48 y 5.49), considerando el tiempo total del sistema.

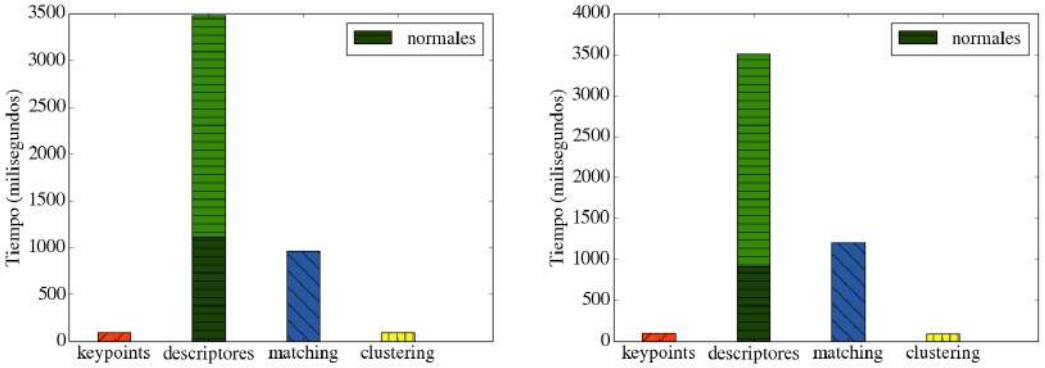
Se puede observar que el porcentaje de estimación de normales dentro de la elaboración de descriptores no es tan grande como para los modelos, esto puede deberse a la diferencia de densidades de las nubes, los nubes correspondientes a los cuadros de las escenas son más esparsas que las de los modelos reconstruidos como se puede ver en las tablas 5.6 y 5.7, y por lo tanto en el radio usado para la estimación de normales no habrán tantos puntos en las nubes de las escenas como en las de los modelos.

Para CSHOT, la etapa de búsqueda de correspondencias es la más lenta, tardando aún más que la elaboración de descriptores, como puede verse en las Figuras 5.48, y 5.49. En cambio para Spin Images, donde la etapa de elaboración de descriptores es la que más demora, como se observa en las Figuras 5.46 y 5.47, que en general tarda aproximadamente la mitad que para en el caso de CSHOT.



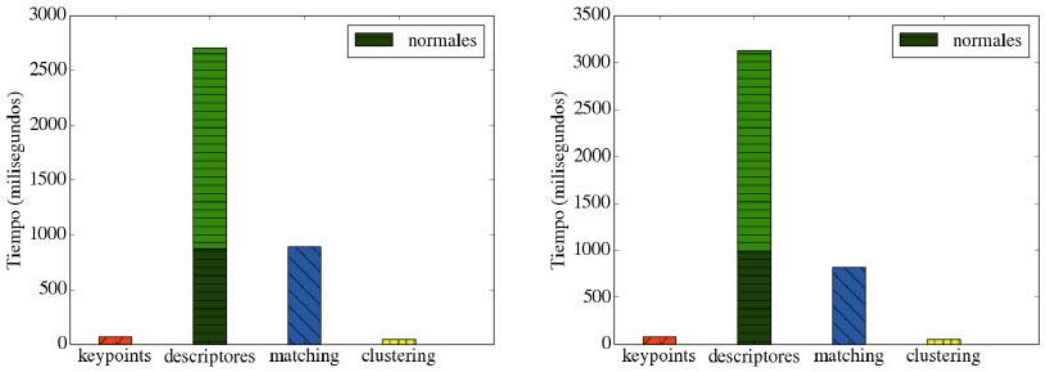
(A) (B)

FIGURA 5.45: Tiempo promedio de cada etapa para el modelo *cap_4* $(\blacktriangle)$ y *coffee_mug_5* (B) en las escenas de *desk_1* usando *FPFH*



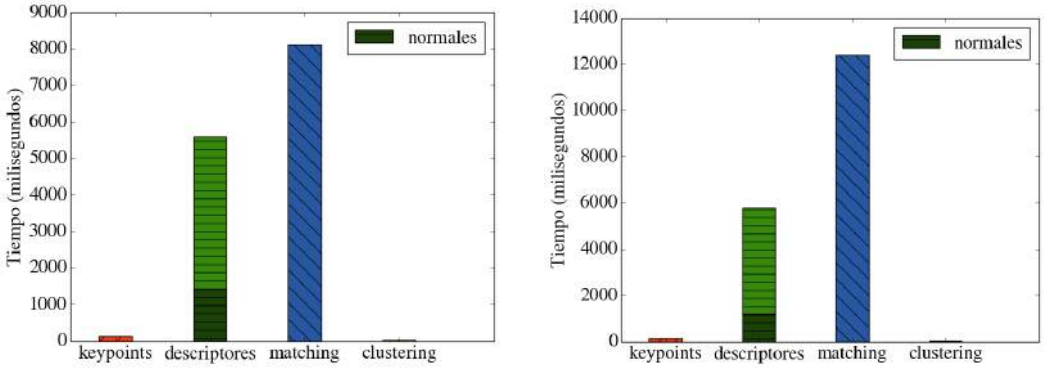
(A) (B)

FIGURA 5.46: Tiempo promedio de cada etapa para el modelo *cap_4* en las escenas de *desk_1* $(\blacktriangle)$ y *table_1* (B) usando *Spin Images*



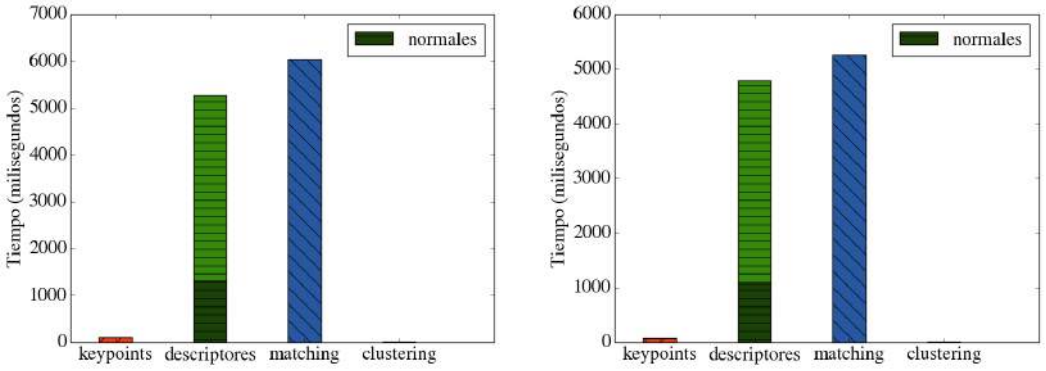
(A) (B)

FIGURA 5.47: Tiempo promedio de cada etapa para el modelo *coffee_mug_5* en las escenas de *desk_1* ($\blacktriangle$) y *kitchen_small_1* (B) usando *Spin Images*



(A) (B)

FIGURA 5.48: Tiempo promedio de cada etapa para el modelo *cap_4* en las escenas de *desk_1* ($\blacktriangle$) y *table_1* (B) usando *CSHOT*



(A) (B)

FIGURA 5.49: Tiempo promedio de cada etapa para el modelo *coffee_mug_5* en las escenas de *desk_1* ($\blacktriangle$) y *kitchen_small_1* (B) usando *CSHOT*

5.7.5. Rendimiento Computacional

El rendimiento del sistema usando Spin Images y CSHOT es parecido, siendo en el caso de Spin Images más preciso en la mayoría de los casos teniendo en cuenta los aciertos y la superposición, y más rápido, aproximadamente 1.5 veces más veloz que usando CSHOT en el caso de los modelos, y más de 2 veces más rápido en el caso de las escenas. Encontramos que el uso de CSHOT otorgaba mejor precisión en el caso de la caja de cereal que tiene más información de color que los otros modelos de la base, pero para el resto de los objetos no, porque tiene solo uno o dos colores. En comparación, al usar FPFH los resultados para esta base fueron muy distintos, vimos que demora mucho y su precisión no es en general comparable con la dada por los otros descriptores.

Como ya vimos, en los datos particulares de esta base, como importa más la forma, Spin Images funciona mejor, y, cuando hay información significativa de color CSHOT otorgaba mejor precisión, por lo cual sería una buena opción modificar el cómputo del descriptor de Spin Images para que incorpore color ([BAGC05]), y permita obtener valores de precisión aún mejores que Spin Images y CSHOT.

5.8. Comparación del rendimiento

En la base de CVLab las escenas son generadas de forma sintética, y los modelos son de muy alta calidad. Esto hace que el reconocimiento de objetos se simplifique significativamente. En la escena los objetos pueden aparecen rotados y trasladados, pero no tapados por otros objetos ni con partes faltantes. Los objetos nunca se superponen, y siempre son totalmente visibles. Además, la información de la transformación que debe usarse para localizar los objetos en las escenas permite evaluar precisamente la exactitud del reconocimiento. En este caso todos los descriptores dieron buenos resultados, siendo Spin Images más preciso en la mayoría de los casos y además más rápido.

En la base RGB-D, tanto las vistas de los modelos como las de las escenas provienen de capturas del mundo real. Los datos tienen ruido y hay información faltante, las nubes de puntos obtenidas no son tan detalladas como las existentes en la base de CVLab. En la escena los objetos pueden aparecer no solo rotados y traslados, sino que también parcialmente ocultos, con partes faltantes, y junto a otros objetos de forma similar. La información del rectángulo dónde debería encontrarse el objeto no es precisa en algunos casos, puede ser que sea muy chico, dejando partes del objeto afuera, o muy grande, abarcando grandes partes de la imagen donde no está el objeto. En este caso el uso de cualquier descriptor resultó en un buen rendimiento para la mayoría de los casos, y con

Spin Images se obtuvo un mejor rendimiento, tardando menos que CSHOT y FPFH, y dando mejores precisiones en casi todos los casos.

Una particularidad de los objetos de esta base es que tienen poca textura, excepto para las cajas de cereal que tienen poca información de forma pero mucha textura, haciendo que el uso de un enfoque que combina color y forma, como CSHOT, no ofrezcan mejores resultados que solamente utilizando la forma. Notamos, como un gran cambio, que el uso de FPFH otorgaba resultados mucho peores que para los otros descriptores, el sistema resultaba muy lento, tardando más de 25 veces más en el caso de los modelos, y más de 8 veces para las escenas. Además la precisión no fue comparable a la dada con los otros descriptores, como para justificar esta gran diferencia en el tiempo.

Una opción para mejorar la exactitud de las medidas de precisión de esta base sería revisar manualmente la información de ground truth, que llevaría mucho tiempo porque la verificación de los valores debería hacerse cuadro a cuadro.

Otros trabajos con la base RGB-D

Existen muchos trabajos de reconocimiento de objetos con esta misma base, como por ejemplo los de Bo, Lai, Ren y Fox [BLRF11], [LBRF11b], [LBRF11a], y el de Alexandre [Ale12].

Nos interesa en particular el trabajo de Alexandre [Ale12], donde se realiza una evaluación comparativa de descriptores 3D, usando SHOT, CSHOT, FPFH y otros (sin considerar Spin Images), y se encuentra que CSHOT era el que mejor balance otorgaba entre precisión y complejidad temporal. El sistema que se utiliza consiste en obtener keypoints, calcular descriptores y de una etapa de matching. A diferencia del sistema implementado en esta tesis (Sección 4.3) que busca correspondencias entre puntos, el matching se realiza usando machine learning y considerando distancias entre conjuntos de correspondencias. La distancia entre conjuntos está dada por la suma de la distancia L1 (ver Ecuación 2.2) entre los centroides más la distancia L1 entre los desvios standard en cada dimensión. Para medir la precisión, solo se consideran los modelos, entrenando al sistema con vistas de estos desde dos puntos de vista, y evaluándolo con un tercer punto de vista.

Por otra parte, en el paper de Bo, Lai, Ren y Fox [LBRF11a], además de presentar la base analizan técnicas reconocimiento usando machine learning, y las evalúan por categoría y por instancia. El reconocimiento a nivel de categoría involucra la clasificación de objetos como pertenecientes a la misma categoría de otros objetos ya vistos (usados en el entrenamiento). Por ejemplo una categoría puede ser la taza de café. El reconocimiento

a nivel de instancia consiste en identificar si un objeto es físicamente el mismo que otro ya visto. Es decir si es la misma instancia, por ejemplo para las tazas de café existen 6 distintas instancias, similares entre ellas por ser tazas, pero cada una con características distintivas.

Capítulo 6

Conclusiones y Trabajo Futuro

Desarrollamos un esquema de sistema de reconocimiento de objetos que permite utilizar distintos descriptores, Spin Images, SHOT, CSHOT y FPFH, y una variedad de métodos para instanciar distintos sistemas, y adecuarlo a situaciones particulares. Analizamos la capacidad informativa de los descriptores y sus distancias asociadas, obteniendo que usar el cociente entre los dos vecinos más cercanos da mejores resultados que el uso de la distancia al vecino más cercano.

Experimentamos detalladamente cambiando cada etapa del sistema, variando sus parámetros y propiedades, viendo su influencia en la precisión del sistema y los tiempos. Obtuvimos que el uso de un submuestreo uniforme es más rápido que usar el detector de keypoints de Harris, y resulta en más puntos. Además un tamaño más chico de submuestreo otorga mejores precisiones a costa de mayores tiempos. Al reducir el radio de los descriptores y normales se disminuye el tiempo de generación de descriptores pero se incrementa el tiempo de búsqueda de correspondencias. Finalmente, el uso de RANSAC sobre todas las correspondencias es más preciso que el uso de agrupación con consistencia geométrica.

Para evaluar el rendimiento de los distintos sistemas usamos dos bases de datos públicamente disponibles, con información de ground truth, una siendo un caso ideal y la otra un caso real. Vimos que el sistema daba buenos valores de precisión y tiempos bajos para todos los descriptores en la mayoría de los casos de la primera base, y para la base con datos reales solo usando los descriptores de CSHOT y Spin Images. En particular la mayoría de los objetos de esta base no tienen mucha textura, haciendo que el uso de la información de color no otorgara mejoras. Sin embargo vimos que en los casos donde sí existía textura, el rendimiento de CSHOT resultó mejor que usando Spin Images. Esto da la idea de que Spin Images describe de mejor manera la forma para estos objetos.

En el rendimiento influye mucho la densidad de las nubes, y la cantidad de puntos elegidos, haciendo que se modifique la distribución de tiempos de las etapas y la precisión final. El tiempo de la etapa de matching depende de la cantidad de puntos elegidos, y el tiempo de clustering depende de la cantidad de correspondencias encontradas, que a su vez varía según el descriptor. Al disminuir la densidad de las nubes se incorpora menos información en los descriptores, haciendo que sean menos informativos. Por otro lado, nubes muy densas hacen que el reconocimiento sea mucho más preciso pero que demore más por la cantidad extra de información que se utiliza para estimar las normales y elaborar descriptores.

Trabajo futuro

La cantidad de temas posibles para futuros trabajos es grande, dado que el reconocimiento de objetos involucra muchos conceptos y no es un problema simple.

Una opción es analizar el rendimiento del sistema en otras situaciones, como por ejemplo por instancia y categoría, como se presenta en el paper de la base de datos RGBD [LBRF11a]. También se puede considerar la utilización de otras bases de datos de objetos tridimensionales.

Pueden considerarse otros descriptores además de CSHOT, FPFH y Spin Images, en particular aquellos que consideren tanto la información de la forma como la del color, por ejemplo Spin Images con color [BAGC05], y el descriptor de *MeshHOG* [ZBVH09].

También es una opción modificar el sistema para precalcular normales de todas las nubes que se desee considerar, y luego recibir las nubes de los modelos y las escenas con normales precalculadas, haciendo que ese tiempo de estimación de normales no afecte el tiempo total del sistema de reconocimiento.

Por último, puede hacerse una comparación con el trabajo de Alexandre [Ale12], donde encuentran que CSHOT otorga mejor balance otorgaba entre precisión y complejidad temporal, pero no consideran Spin Images. En nuestro trabajo vimos que el rendimiento de Spin Images y CSHOT fue comparable, por lo cual sería interesante contrastar ambos.

Bibliografía

[Ale12] Luıs A Alexandre. 3D descriptors for object and category recognition: a comparative evaluation. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Color-Depth Camera Fusion*. Citeseer, 2012.

[AMT+12] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Point Cloud Library. *IEEE Conference on Robotics and Automation (ICRA)*, 1070(9932/12), 2012.

[BAGC05] Nicola Brusco, Marco Andreetto, Andrea Giorgi, and Guido Maria Cortelazzo. 3D registration by textured spin-images. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 262–269. IEEE, 2005.

[Bal81] Dana H Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[BLRF11] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1729–1736. IEEE, 2011.

[BM92] Paul J Besl and Neil D McKay. A method for registration of 3-D shapes. 14(2):586–606, 1992.

[BMP02] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(4):509–522, 2002.

[Bol] University Of Bologna. Computer Vision LAB. SHOT database `http://www.vision.deis.unibo.it/research/80-shot`.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[Fai13] Mark D Fairchild. *Color appearance models*. John Wiley & Sons, 2013.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[HKH+10] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *International Symposium on Experimental Robotics (ISER)*. Citeseer, 2010.

[HRBB09] Andreas Holzbach, Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast geometric point labeling using conditional random fields. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7–12. IEEE, 2009.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Plessey Research, 1988.

[JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449, 1999.

[Joh97] Andrew Edie Johnson. *Spin-images: A Representation For 3-D Surface Matching*. PhD thesis, The Robotics Institute, Carnegie Mellon University, 1997.

[Kin] Microsoft Kinect. `http://www.xbox.com/en-us/kinect`.

[LBRF11a] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2011.

[LBRF11b] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *IEEE Conference on Robotics and Automation (ICRA)*, pages 4007–4013. IEEE, 2011.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157. IEEE, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[LS81] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[Ma04] Yi Ma. *An invitation to 3-D vision: From images to geometric models*, volume 26. Springer Science & Business Media, 2004.

[MBO10] Ajmal Mian, Mohammed Bennamoun, and R Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision (IJCV)*, 89(2-3):348–361, 2010.

[Mü] Technische Universität München. Computer Vision Group, SLAM database https://vision.in.tum.de/data/datasets/rgbd-dataset/download.

[NOC+12] Erickson R Nascimento, Gabriel L Oliveira, Mario Fernando Montenegro Campos, Antônio Wilson Vieira, and William Robson Schwartz. BRAND: a robust appearance and depth descriptor for RGB-D images. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1720–1726. IEEE, 2012.

[PCSM13] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE Conference on Robotics and Automation (ICRA)*, pages 3212–3217. IEEE, 2009.

[RBMB08] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3384–3391. IEEE, 2008.

[RBTH10] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2155–2162. IEEE, 2010.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011.

[RHBB09] Radu Bogdan Rusu, Andreas Holzbach, Michael Beetz, and Gary Bradski. Detecting and segmenting objects for mobile manipulation. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 47–54. IEEE, 2009.

[Rus10] Radu Bogdan Rusu. Semantic 3D object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[SEE+12] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/SRJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012.

[SRKB11] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *IEEE Conference on Robotics and Automation (ICRA)*, pages 2601–2608. IEEE, 2011.

[Sta] University Of Stanford. Computer Graphics Lab. The Stanford 3D scanning repository http://graphics.stanford.edu/data/3Dscanrep/.

[STDS11] Samuele Salti, Federico Tombari, and Luigi Di Stefano. A performance evaluation of 3D keypoint detectors. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 236–243. IEEE, 2011.

[TSDS10] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*, pages 356–369. Springer, 2010.

[TSDS11] Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3D feature matching. In *IEEE International Conference on Image Processing (ICIP)*, pages 809–812. IEEE, 2011.

[TSDS13] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision (IJCV)*, 102(1-3):198–220, 2013.

[Was] University Of Washington. RGB-D Database http://rgbd-dataset.cs.washington.edu/.

[Xti] Asus Xtion. http://www.asus.com/Multimedia/Xtion/.

[ZBVH09] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 373–380. IEEE, 2009.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision (IJCV)*, 13(2):119–152, 1994.