Universidad Zaragoza
1542

Proyecto Fin de Carrera

Learning of Objects Models for RoboEarth database

Autor/es

Antonio Esteban Lansaque

Director

Arjen den Hamer

Codirector

Juan Domingo Tardós Solano

Ingeniería informática

Escuela de Ingeniería y Arquitectura

2013

Learning of Objects Models for RoboEarth database

Resumen

Philips Innovation Services y el grupo de robótica de la Universidad de Zaragoza están cooperando en el proyecto europeo de RoboEarth, cuyo objetivo es crear una base de datos en Internet donde diferentes tipos de robots puedan compartir modelos y estrategias de manipulación de objetos. El objetivo de este trabajo es el de construir una herramienta capaz de reconstruir modelos en 3D de diferentes tipos de objetos para almacenarlos en la base de datos de RoboEarth.

Los modelos son obtenidos escaneando los objetos con cámaras RGB-D como la kinect que proveen imágenes y nubes de puntos en 3D. La librería VSLAM (Visual Simultaneous Localization and Mapping) desarrollada por UNIZAR se usa para calcular la trayectoria de la cámara y para obtener un conjunto de keyframes. Las nubes de puntos pertenecientes a los keyframes son las que una vez registradas forman el modelo en 3D del objeto.

Las nubes de puntos que son utilizadas para reconstruir los modelos no solo contienen los puntos correspondientes al objeto sino que también contienen puntos del entorno donde fueron tomadas. Por tanto, el primer paso para reconstruir el objeto es el de eliminar de las nubes los puntos que pertenecen al entorno y no al objeto. Una vez que los puntos del entorno han sido eliminado de las nubes, se realiza el proceso de registrado. Para reconstruir los modelos en 3D de objetos se han estudiado dos métodos diferentes. El primero de los métodos es un método basado en puntos de interés como SURF o ORB. El segundo de los métodos se basa en el conocido Iterative Closest Point (ICP).

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2012 under grant agreement number 248942 RoboEarth.

Índice general

Capítulo 1

Introducción

1.1. Motivación

El incremento de la capacidad computacional y de la posibilidad de encontrar sensores de bajo coste han impulsado el desarrollo de nuevas aplicaciones de visión por computador. Un campo en particular que se beneficia de esta tendencia es la robótica. Un ejemplo práctico de una de estas aplicaciones podría ser una aspiradora autónoma que reconoce objetos para averiguar el lugar donde se encuentra y así ajustar el modo en el que tiene que limpiar.

Un problema importante pero a la vez difícil de solucionar en robótica es el de obtener información de entornos en aplicaciones no industriales donde el sistema opera en entornos desconocidos. Una aproximación para que robots puedan interactuar con el entorno es que estos pudieran compartir conocimiento mediante una base de datos que les permitiera obtener información como mapas de navegación, estrategias de manipulación de objetos o modelos de objetos.

RoboEarth [17] es una red y base de datos donde los robots pueden compartir información y aprender unos de otros. (Figura 1.1). El propósito principal de RoboEarth es el de proporcionar a los robots el conocimiento necesario para interactuar con entornos desconocidos.

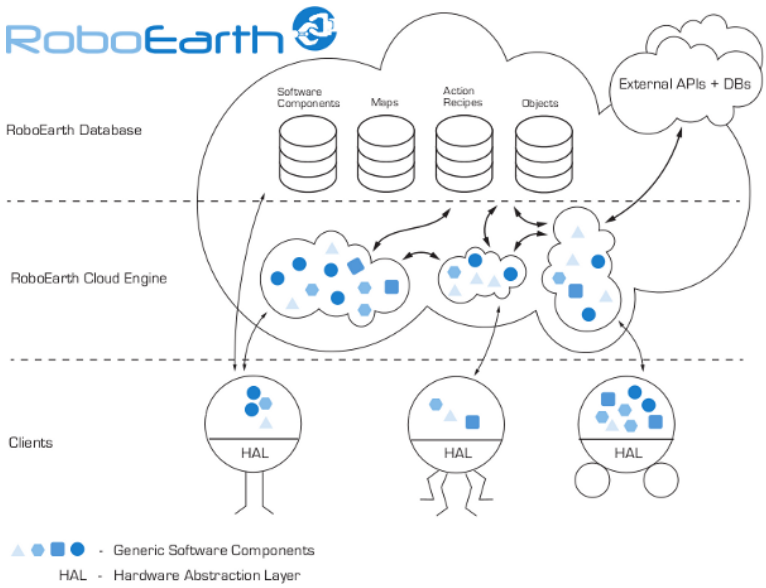

Figura 1.1: Estructura de RoboEarth.

El objetivo que se quiere conseguir con el trabajo desarrollado en esta memoria es el da la creación de modelos en 3D de objetos. Estos modelos serán utilizados para almacenarlos en la base de datos de

RoboEarth. Una vez que estos objetos hayan sido añadidos a la base de datos de RoboEarth, diferentes tipos de robots podrán conectarse a la base de datos para recuperar estos modelos. De este modo, los robots podrán reconocer los objetos en un entorno cualquiera y así ser capaces de manipularlos.

1.2. Conceptos básicos para registrar y segmentar nubes de puntos

El propósito de esta sección es introducir algunos conceptos importantes que serán necesarios en los siguientes capítulos de esta memoria. Al lector al que le sean familiares estos conceptos, esta sección le puede servir para refrescar conceptos o puede continuar directamente con la siguiente sección del capítulo. Los conceptos que serán desarrollados son: nube de puntos, matriz de transformación, registrado de nube de puntos y segmentación de nubes de puntos.

1.2.1. Nube de puntos

Una nube de puntos es el punto de partida para la mayoría de algoritmos descritos en esta memoria. Una nube de puntos P es simplemente un conjunto de puntos en 3D que están definidos por tres coordenadas (X, Y, Z). Por lo tanto, una nube de puntos puede ser vista como un conjunto de puntos que están definidos en el espacio euclídeo.

$$P = \{(x, y, z)_0, ..., (x, y, z)_i, ..., (x, y, z)_n\} \tag{1.1}$$

Las nubes de puntos pueden ser obtenidos de escáneres o cámaras que detectan la profundidad como la kinect. Depende del escáner si hay información extra en la nube de puntos a parte de la información espacial. Por ejemplo, cuando se usan escáneres RGB-D para crear la nube de puntos, la información RGB de los puntos también está presente en la nube de puntos.

$$P_{rgb} = \{(x, y, z, rgb)_0, ..., (x, y, z, rgb)_i, ..., (x, y, z, rgb)_n\} \tag{1.2}$$

1.2.2. Matriz de transformación

La matriz de transformación T es una matriz que mapea un punto p en otro punto p' de acuerdo con la siguiente ecuación

$$\begin{pmatrix} p' \\ 1 \end{pmatrix} = T \begin{pmatrix} p \\ 1 \end{pmatrix} \tag{1.3}$$

donde $p = [x, y, z]^T$ y $p' = [x', y', z']^T$.

Una matriz de transformación describe una rotación y una traslación de un punto en el espacio al mismo tiempo. Así, la matriz de transformación T definida en la ecuación 1.3 puede ser desglosada en términos de una rotación y una translación

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \tag{1.4}$$

donde la matriz 3x3 izquierda superior (R) se corresponde con la matriz de rotación y el vector 3x1 superior derecho (t) se corresponde con el vector de traslación.

1.2.3. Registrado de nubes de puntos

Cuando una cámara toma imágenes mientras se mueve a través de un entorno, las imágenes son tomadas desde diferentes puntos de vista. En muchos casos, puede ser interesante representar las nubes de puntos en un sistema de referencia común para compensar el movimiento de la cámara.

El proceso de encontrar y aplicar la transformación que convierte diferentes nubes de puntos en un mismo sistema de referencia (sistema que define inequívocamente todos los puntos que pertenecen al

sistema) se llama registrado de nubes de puntos.

Dadas dos nubes de puntos M (conjunto modelo) y D (conjunto a registrar) obtenidas en un mismo entorno pero desde diferente puntos de vista, se buscan puntos correspondientes entre ambas nubes de puntos. Así, el registrado de las nubes de puntos (Figura 1.2) se reduce a encontrar la transformación T que minimiza el error de alineación entre los puntos correspondientes

$$\underset{T}{\operatorname{argmin}} \left\{ \sum_{i=0}^{n} ||T \cdot d_i - m_i||^2 \right\} \tag{1.5}$$

donde d_i y m_i representan los puntos de la correspondencia i-ésima entre las nubes de puntos D y M.

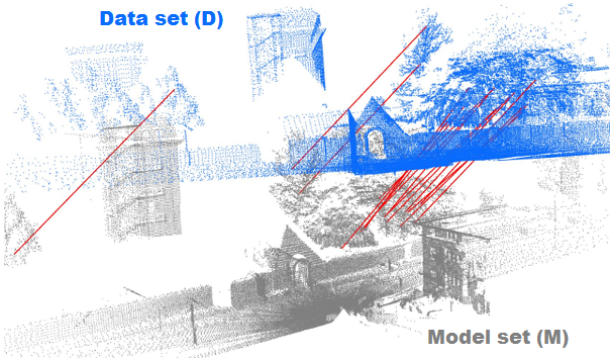

Figura 1.2: Ejemplo de un conjunto modelo, de un conjunto a registrar y de las correspondencias entre los conjuntos.

Una vez encontrada la transformación T, el nuevo modelo quedará formado por la unión del modelo anterior y la nube de puntos:

$$M = M \cup T \cdot P_i \tag{1.6}$$

En el caso de utilizar como referencia del modelo la de la última nube de puntos, el modelo quedará:

$$M = P_i \cup T^{-1} \cdot M \tag{1.7}$$

1.2.4. Segmentación de nubes de puntos

El objetivo de esta memoria es generar modelos 3D de objetos. Sin embargo, los objetos están presente en un entorno que no forma parte del objeto. Para separar el objeto del entorno, se realiza un proceso llamado segmentación.

Segmentación es el proceso de separar una nube de puntos en distintos grupos. En esta memoria, se define segmentación como el proceso de eliminar todo punto que pertenece al entorno y no al objeto de una nube de puntos.

1.3. Objetivo del proyecto

El objetivo de este proyecto es el de reconstruir modelos en 3D de objetos (Figura 1.3) que serán almacenado en la base de datos de RoboEarth. El modelo es creado registrando N nubes de puntos obtenidas desde diferentes puntos de vista. Estas nubes de puntos se encuentran en un entorno que se segmenta para que las N nubes de puntos solo contengan los puntos pertenecientes al objeto.

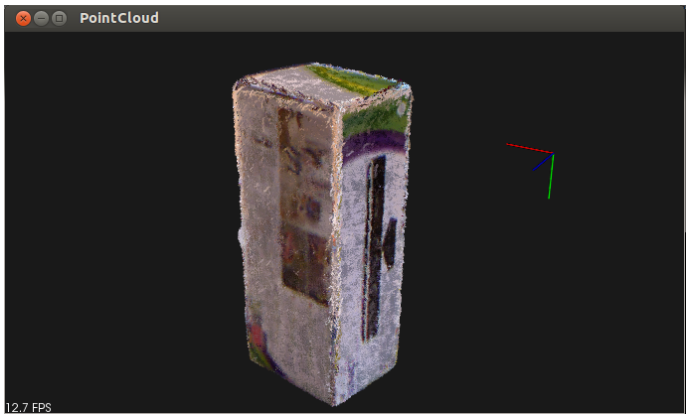

Figura 1.3: Example of a reconstructed box.

1.4. Resumen

En esta sección se da una visión general de como es el proceso de reconstrucción 3D de un objecto mediante dos de los métodos estudiados en esta memoria. Un método es el Iterative Closest Point (ICP) y el otro es un método diseñado especialmente para este proyecto basado en puntos de interés.

Las entradas para ambos métodos son N nubes de puntos $(P_0, ..., P_i, ..., P_{N-1})$ las cuales tienen una estimación de la posición y orientación de la cámara con respecto a la cual fueron obtenidas. Esta estimación es calculada usando el algoritmo VSLAM [9] (Visual Simultaneous Tracking and Mapping) desarrollado por un grupo de la universidad de Zaragoza. Antes de empezar el proceso de reconstrucción del modelo, las N nubes de puntos son preprocesadas de tal forma que todo punto del entorno que no forme parte del objeto es eliminado.

El resultado final obtenido de ambos métodos son las N nubes de puntos en un mismo sistema de coordenadas. El conjunto de las N nubes de puntos transformadas a este sistema de coordenadas forma el modelo 3D del objeto. El sistema de coordenadas con respecto al cual el modelo está definido es el sistema de coordenadas de la cámara que tomo la ultima nube de puntos que es registrada.

1.4.1. Método basado en puntos de interés

En esta sección vamos a explicar el proceso que sigue una nube de puntos P_i para ser registrada con un modelo M previamente calculado que contiene fusionadas las nubes $P_0, ..., P_{i-1}$ $(M = \{P_0, ..., P_{i-1}\})$. Cada nube de puntos $P_0, ..., P_{i-1}$ tiene asociados los puntos de interés y descriptores calculados de cada una de sus imágenes.

La primera iteración del algoritmo, el modelo M con el que registraríamos la primera nube de puntos P_0 no contiene ningún punto, por lo que en esta primera iteración la única acción que se realiza es convertir la nube P_0 en el modelo M $(M = P_0)$. El proceso (Figura 1.4) para el resto de nube de puntos es el siguiente:

1. Se obtienen puntos de interés (ORB o SURF) de la imagen i-ésima. Solo los keypoints que tengan información 3D son tenidos en cuenta. Además, los puntos que no pertenecen al objetos tampoco son tenidos en cuenta ya que son eliminados en la etapa de pre procesamiento.

2. Usando las características obtenidas, se calcula el vecino más cercano de la imagen i en la $i-1$ y de la imagen $i-1$ en la imagen i. Si para ambas búsquedas los keypoints coinciden, estos keypoints forman una posible coincidencia (Sección 3.1.1). Este proceso se repite para las r nubes de puntos registradas anteriores al registro de la nube P_i por lo que de este paso obtenemos r conjuntos de posibles coincidencias.

3. RANSAC se aplica a cada uno de los r conjuntos de posibles coincidencias. Para ello, se obtienen aleatoriamente conjuntos de 3 coincidencias y se calcula la transformación que los mapea. La transformación calculada se aplica a todo el conjunto de posibles coincidencias y se comprueba cuales han sido mapeadas con un cierto umbral de aceptación (Sección 3.1.2). Este proceso se realiza un número determinado de veces suficientemente grande y para cada uno de los r posibles conjuntos de coincidencias se conserva el conjunto que más coincidencias contenga.

4. En este punto tenemos r conjuntos de coincidencias que son fusionados en un mismo conjunto que contenga todos los emparejamientos. Usando este conjunto de correspondencias, se calcula la transformación T que minimiza la distancia entre todas las coincidencias vía mínimos cuadrados (Sección 3.1.3).

5. Una vez se calcula la transformación T, el modelo M que se utiliza en la siguiente iteración del método es actualizado de la siguiente forma $M = T^{-1} \cdot M \cup P_i$.

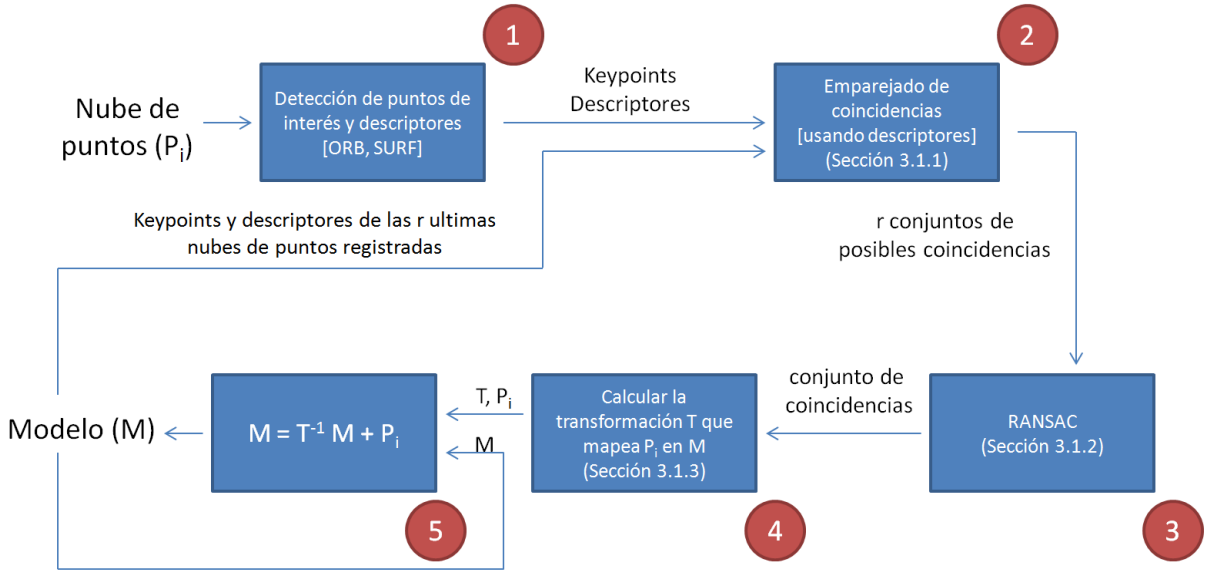

Figura 1.4: Figura que muestra gráficamente el proceso de obtención del modelo en 3D vía puntos de interés.

1.4.2. Método basado en Iterative Closest Point

En esta sección vamos a explicar el proceso que sigue una nube de puntos $P_i = \{p_0, ..., p_n\}$ para ser registrada con un modelo M que contiene fusionadas las nubes $P_0, ..., P_{i-1}$. En este caso vamos a realizar el proceso de registrado utilizando el ICP para obtener la transformación que mapea P_i en M. Como en el método anterior, en la primera iteración del método, el modelo M con el que registraríamos la primera nube de puntos P_0 no contiene ningún punto, por lo que en esta primera iteración la única acción que se realiza es la de convertir la nube P_0 en el modelo M ($M = P_0$).

El ICP utiliza una estimación inicial T para el cálculo de las correspondencias. Esta estimación inicial la obtenemos de la posición y orientación de la cámara calculada por el VSLAM. El proceso (Figura 1.5) de registrado de una nube de puntos P_i es el siguiente:

1. Para cada punto de P_i buscamos su vecino más cercano en M usando las estimación inicial T.

2. Recalcular la transformación T' que minimiza las distancias entre los puntos de P_i y sus vecinos más cercanos

$$\underset{T}{\operatorname{argmin}}\{\sum_{j=0}^{n} ||T \cdot p_j - m_j||^2\} \tag{1.8}$$

donde m_j es el vecino más cercano del punto p_j en el modelo M. Si la diferencia entre las transformaciones de dos iteraciones consecutivas es menor que un umbral o se ha llegado a un número

máximo de iteraciones se considera que el algoritmo ha convergido a la transformación T deseada. Sino se vuelve al paso uno hasta que el algoritmo converge.

3. Una vez que se ha calculado la transformación T, utilizamos la transformación T^{-1} para transformar el sistema de referencia del modelo en el sistema de referencia de la nube de puntos P_i.

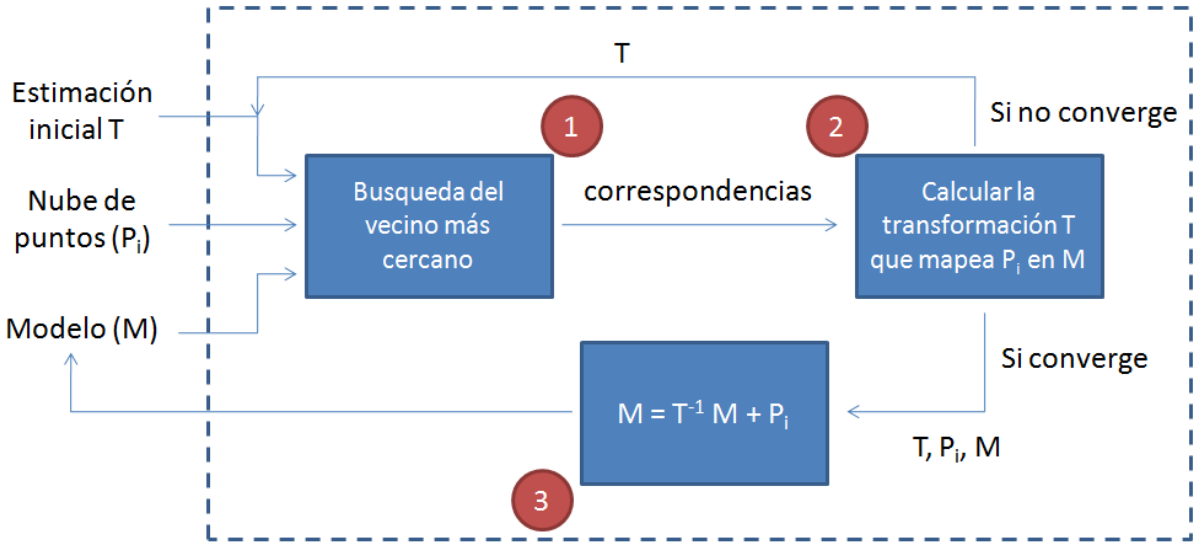

Figura 1.5: Figura que muestra gráficamente el proceso de obtención del modelo en 3D vía ICP.

Capítulo 2

Segmentación de nubes de puntos

En este proyecto, asumimos que el objeto a reconstruir se encuentra dentro de un entorno. Como no todos los puntos del entorno pertenecen al objeto, necesitamos eliminar estos puntos para quedarnos únicamente con el objeto. Este proceso de eliminación de puntos se llama segmentación. El proceso de segmentación se realiza en dos pasos. El primero y más importante se realiza antes de registrar las nubes de puntos y el segundo después de registrarlas.

2.1. Segmentación antes del registrado

El proceso de segmentación que se realiza antes del registrado consta de dos pasos:

1. eliminar los puntos de la nube que pertenecen al fondo (Figura 2.4). Este primer paso para deshacernos del fondo se explica en la sección 2.1.1.

2. eliminar los puntos de la nube que pertenecen a la superficie donde descansa el objeto que estamos reconstruyendo (Figura 2.5). Este segundo paso se explica en la sección 2.1.2.

Para dar un ejemplo práctico de como se segmenta un objeto en esta primera fase, vamos a tomar como ejemplo la nube de puntos de la Figura 2.1. A partir de esta, iremos viendo como se realiza la eliminación del fondo y del plano.

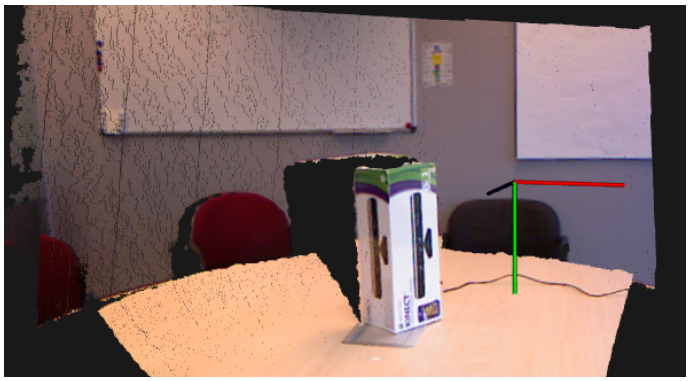

Figura 2.1: Estado inicial de una nube de puntos a segmentar.

2.1.1. Eliminación del fondo

La herramienta diseñada para reconstruir los modelos asume que todas las nubes de puntos y sus posiciones están almacenadas en disco. La herramienta también asume que el usuario se mueve alrededor del objeto. Según la definición del sistema de coordenadas de la kinect, el plano X-Z es paralelo al plano donde descansa el objeto. Por lo tanto, podemos detectar el valor mínimo y máximo de las posiciones de la cámara en X y Z (Figura 2.2). Los puntos cuyo valor en X no este entre el valor mínimo y máximo calculado

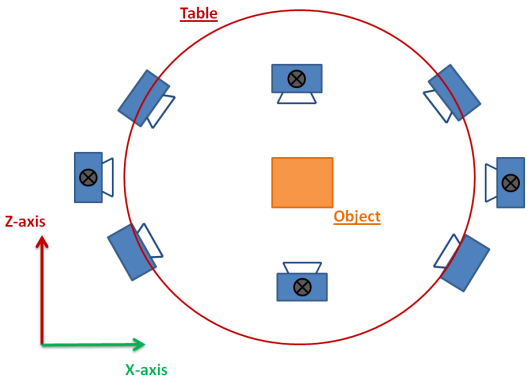

Figura 2.2: Esta figura muestra un posible escenario donde las cruces negras representan el valor mínimo y máximo en X y Z de las posiciones de la cámara que capturó las nubes de puntos a registrar.

entorno al eje X son eliminados de la nube de puntos. Lo mismo ocurre para el eje Z respectivamente. La eliminación de estos puntos implica que la eliminación del fondo de la nube de puntos como puede verse en la Figura 2.3.

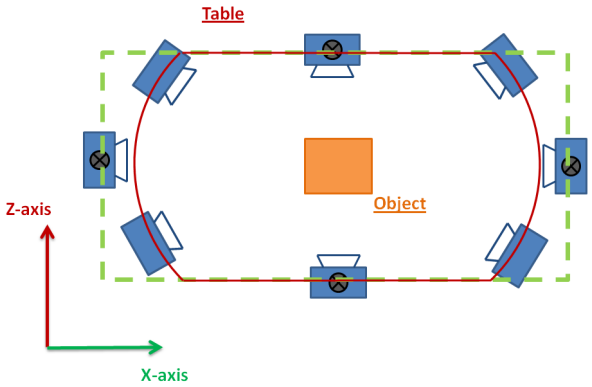

Figura 2.3: Esta figura muestra el resultado después de aplicar el método explicado en esta sección.

Aplicando el método descrito en esta sección a la Figura 2.1 podemos observar en la Figura 2.4 como efectivamente se ha eliminado el fondo de la nube de puntos.

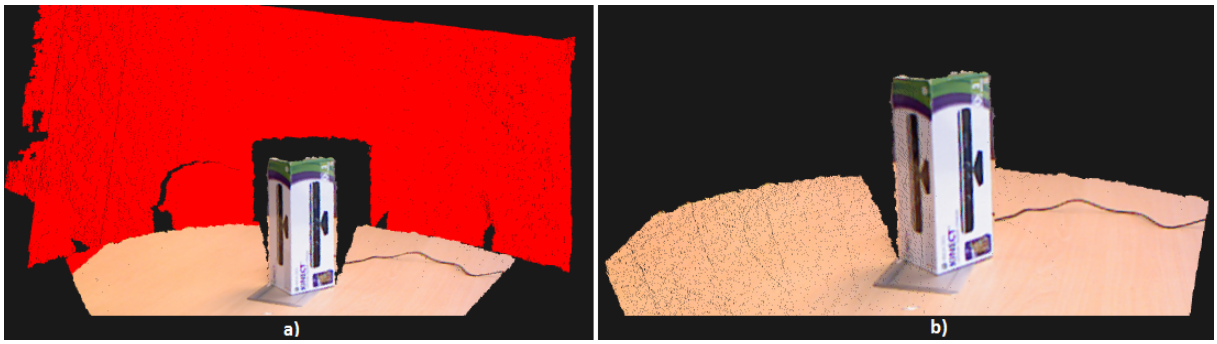

Figura 2.4: a) Fondo detectado y teñido de rojo. b) Resultado de eliminar el fondo de la nube de puntos.

2.1.2. Eliminación del plano

Los objetos que queremos reconstruir normalmente descansan sobre una superficie plana que puede ser tanto el suelo como una mesa. Podemos hacer uso de esta suposición para detectar la superficie y

así eliminar los puntos que pertenecen a esta superficie de la nube de puntos.

Para detectar el plano en la nube de puntos se usa RANSAC. El proceso de detección del plano se describe en los siguientes puntos:

1. seleccionar tres puntos aleatorios y estimar los parámetros del plano que contiene esos tres puntos.
2. comprobar todos los puntos que pertenecen al plano. Se considera que un punto pertenece al plano si la distancia del punto al plano es menor que un umbral u definido por el usuario.
3. los dos primero pasos se repiten N veces. Una vez que se haya iterado las N veces, se eliminan los puntos del plano que más puntos tuviera a lo largo de las N iteraciones.

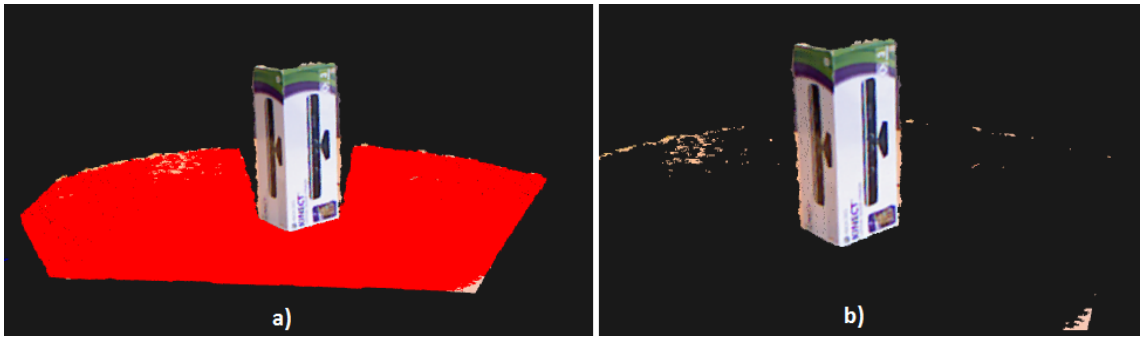

Figura 2.5: a) Plano donde reside el objeto detectado y teñido de rojo. b) Resultado de eliminar el plano donde reside el objeto de la nube de puntos.

2.2. Segmentación después del registrado

El objetivo de este segundo paso de la segmentación es eliminar algunas imperfecciones que pueden aparecer en el modelo reconstruido. Este proceso se muestra en la Figura 2.6. Estas imperfecciones son debidas a dos hechos que no podemos evitar. El primero hecho es que la kinect no es perfecta en la estimación de la profundidad por lo que en las nubes de puntos devueltas por la kinect aparecen espurios. El segundo hecho es que la detección del plano donde el objeto descansa tampoco es perfecta como puede verse en la Figura 2.5.

Como se puede ver en la imagen a) de la Figura 2.6, los espurios que queremos eliminar están muy esparcidos. Así, aprovechándonos de esta descentralización de los espurios, eliminaremos los puntos que en un radio r tengan menos de N vecinos. Por tanto, fijamos un número de vecinos N y un radio r suficientemente grande para que estos espurios sean eliminados pero los puntos del objeto no (Figura 2.6).

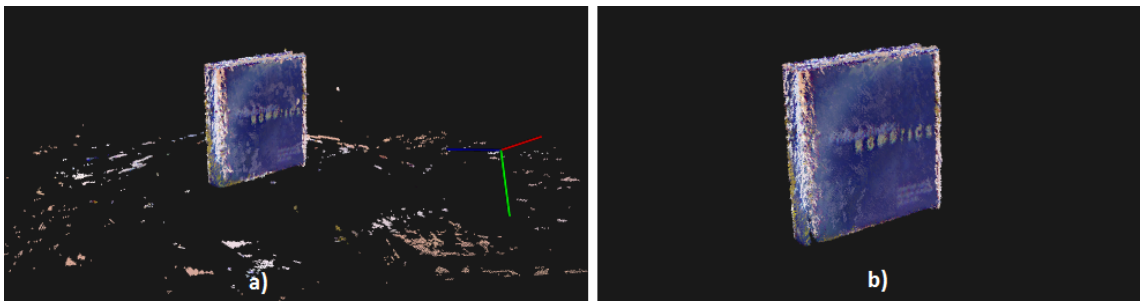

Figura 2.6: a) Modelo en 3D reconstruido con espurios. b)Modelo en 3D reconstruido sin espurios.

Capítulo 3

Registrado de nubes de puntos

El propósito de esta sección es el de definir métodos que reconstruyan modelos de objetos en 3D registrando nubes de puntos desde diferentes puntos de vista. Para registrar dos nubes de puntos, se tiene que encontrar una matriz de transformación T que mapee los puntos de las nubes de puntos con respecto al mismo sistema de referencia. Se han investigado dos métodos que calculan esta transformación T. El primer método usa keypoints y sus descriptores para obtener correspondencias entre las nubes de puntos y así calcular la transformación T. El método desde un punto de vista global se explica en la sección 3.1. El segundo método se basa en el conocido algoritmo ICP (Iterative Closest Point) y se describe en la sección 3.2.

3.1. Método basado en puntos de interés

En la Figura 1.4, se presenta una visión general del proceso que registra varias nubes de puntos usando keypoints. Además, en la sección 1.2.1 de esta memoria se describe una version esquematizada del método que será explicada al detalle en esta sección.

3.1.1. Emparejamiento inicial entre dos nubes de puntos

Las nubes de puntos obtenidas por la kinect no contienen información espacial para todos los puntos pero si contienen la información RGB. Por lo tanto, la imagen que usaremos para obtener los puntos de interés y los descriptores la podemos recuperar directamente de las nubes de puntos a registrar.

Una vez hemos obtenido la imagen y la hemos transformado a escala de grises, se obtienen los puntos de interés de esta ultima. El tipo de puntos de interés que hemos elegido para la reconstrucción de los modelos 3D de objetos son SURF [1] y ORB [13]. Los puntos de interés se obtienen de la imagen 2D. Pero dada una coordenada en la imagen 2D, podemos obtener la posición en 3D. Por esta razón, a partir de ahora usaremos las coordenadas 3D para la construcción del conjunto de emparejamientos iniciales.

Una vez que se calculan los puntos de interés en 3D y los descriptores correspondientes, el emparejamiento inicial entre los puntos de interés de dos nubes de puntos diferentes puede comenzar. El proceso para obtener el emparejamiento inicial se realiza entre dos nubes de puntos diferentes ($nube1$ y $nube2$), y sus respectivos puntos de interés y descriptores. La búsqueda de los emparejamientos consta de los siguientes pasos:

1. se busca el vecino más próximo de todos los puntos de interés de la $nube1$ en la $nube2$ usando sus respectivos descriptores. Esta búsqueda se realiza atendiendo a la norma L_2 en el caso de puntos de interés SURF y atendiendo a la distancia de Hamming en el caso de puntos de interés ORB.

2. se realiza el mismo proceso que en el caso anterior pero en sentido contrario. Es decir, se busca el vecino más próximo de todos los puntos de interés de la $nube2$ en la $nube1$.

3. si el vecino mas próximo de un punto de interés p_{12} de la $nube1$ en la $nube2$ y el vecino más próximo de un punto de interés p_{21} de la $nube2$ en la $nube1$ coinciden, estamos hablando de que

hemos encontrado una posible coincidencia entre los puntos de interés p_{12} y p_{21}. El conjunto de emparejamientos iniciales está formado por todos los puntos de interés que satisfacen la misma condición que los puntos de interés p_{12} y p_{21}.

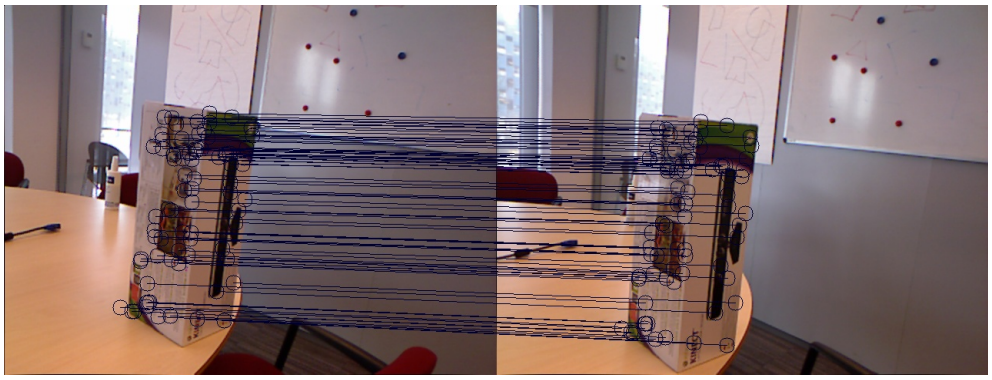

Figura 3.1: Ejemplo de emparejamientos iniciales usando SURF.

3.1.2. RANSAC

En la sección anterior se explica como obtener un conjunto de emparejamiento inicial. Sin embargo en la práctica, es muy probable que aparezcan espurios que pueden hacer que la estimación de la matriz de transformación T no sea la estimación óptima. Para evitar la aparición de estos espurios y obtener una estimación correcta, primero se aplica RANSAC al conjunto inicial de emparejamientos.

RANSAC [6] es el algoritmo más conocido para estimar los parámetros de modelos matemáticos de un conjunto de datos que contiene espurios. RANSAC fue definido por Martin A. Fischler y Robert C. Bolles en 1981. La razón por la que RANSAC es tan usado es porque RANSAC puede estimar los parámetros de un modelo aunque tenga un número significante de espurios. Como se muestra en la Figura 3.2, la presencia de espurios puede provocar estimaciones erróneas. Sin embargo, usando RANSAC, los espurios de un modelo pueden ser detectados y eliminados para obtener así un modelo mucho más preciso.

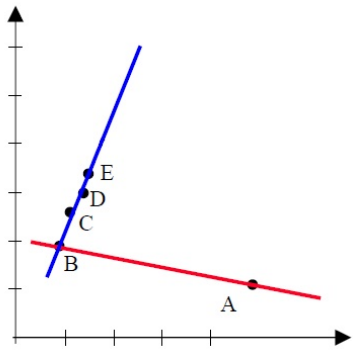

Figura 3.2: La linea roja representa el modelo que minimiza el error total. La linea azul describe el modelo obtenido usando RANSAC.

La idea básica para calcular el conjunto de correspondencias sin espurios entre dos nubes de puntos aplicando RANSAC es la siguiente:

1. elegir tres emparejamientos aleatorios del conjunto de emparejamientos inicial
2. obtener la transformación T que mapea los tres emparejamientos elegidos aleatoriamente.
3. aplicar la transformación T a todos los emparejamientos y ver cuantos de ellos casarían con un cierto umbral de aceptación.

4. los pasos anteriores se repiten N veces. RANSAC devuelve el conjunto que mas emparejamientos mapeados contiene de las N iteraciones.

El algoritmo explicado con mas detalle se presenta en el Algoritmo 1.

Algorithm 1 RANSAC

Require: Conjunto de coincidencias iniciales (*emparejamientos*)
Número de iteraciones que RANSAC se ejecutará (N)
Umbral para descartar espurios (d_{min})
Número de correspondencias mínimas para obtener T ($numEmparejamientosNecesarios$)
Ensure: Conjunto de coincidencias sin espurios (*mejorConjunto*)
$mejorConjunto \leftarrow \emptyset$
for $i \leftarrow 0$ **to** N **do**
$muestras \leftarrow$ seleccionar $numEmparejamientosNecesarios$ aleatoriamente de *emparejamientos*
$T \leftarrow$ calcular transformación T (usando el conjunto *muestras*)
$conjuntoTransformado \leftarrow$ aplicar T a todo punto origen del conjunto *emparejamientos*
for all *coincidencia* **in** $conjuntoTransformado$ **do**
Calcular la distancia euclídea (d) entre coorespondencias
if $d < d_{max}$ **then**
$inliers \leftarrow inliers + coincidencia$
end if
end for
if $|inliers| > |mejorConjunto|$ **then**
$mejorConjunto \leftarrow inliers$
end if
end for
return *mejorConjunto*

No todos los puntos devueltos por la kinect tienen información espacial. Por lo tanto, existen puntos de interés que podrían tener una correspondencia en 2D pero no en 3D. Por consiguiente, hay menos emparejamientos en 3D de los que habría en 2D. Para solucionar esta falta de puntos de interés, si queremos registrar una nube de puntos P_i con el modelo $M = \{P_0, ..., P_{i-1}\}$, el proceso de emparejamiento inicial se realiza entre la la nube de puntos P_i y las r (con r definido por el usuario) nubes de puntos registradas en pasos anteriores del algoritmo $P_{i-1}, ..., P_{i-r}$. De este modo conseguimos un mayor número de emparejamientos iniciales que permite obtener mejores resultados. Una vez que está definido como obtener un conjunto de correspondencias sin espurios entre dos nubes de puntos diferentes, a continuación se describe el algoritmo completo para obtener la transformación T que mapea una nube de puntos P_i con un modelo $M = \{P_0, ..., P_{i-1}\}$

Algorithm 2 Algoritmo para obtener la transformación T

Require: Nubes de puntos a registrar (P_i)
Modelo ($M = \{P_0, ..., P_{i-1}\}$)
Numero de nubes con respecto a las que buscar emparejamientos (r)
Ensure: transformacion (T) que mapea P_i en M
$allInliers \leftarrow \emptyset$
for all *nube* **in** $P_{i-r}..P_{i-1}$ **do**
$correspondencias \leftarrow$ encontrarCorrespondencias(P_i,*nube*)
$inliers \leftarrow$ RANSAC($correspondencias$)
$allInliers \leftarrow allInliers \cup inliers$
end for
$T \leftarrow$ calcularTransformacion($allInliers$)
return T

3.1.3. Mínimos cuadrados iterativos

Como fue descrito en el Capitulo 1, el objetivo del registrado de nubes de puntos es el de encontrar la transformación T que mapea una nube de puntos D (conjunto a registrar) en el modelo M

$$\underbrace{\begin{pmatrix} t_{00} & t_{01} & t_{02} & t_{03} \\ t_{10} & t_{11} & t_{12} & t_{13} \\ t_{20} & t_{21} & t_{22} & t_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{T} \underbrace{\begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}}_{D} = \underbrace{\begin{pmatrix} x'_i \\ y'_i \\ z'_i \\ 1 \end{pmatrix}}_{M} \tag{3.1}$$

El objetivo de esta sección es la de obtener todos los parámetros t_{ij} de la matriz T de la ecuación 3.1. Por lo tanto, el problema de encontrar la transformación T se reduce a resolver el sistema de ecuaciones 3.1 donde $[x_i, y_i, z_i, 1]^T$ y $[x'_i, y'_i, z'_i, 1]^T$ representan los puntos de cada una de las coincidencias calculadas de acuerdo con la sección anterior.

La matriz T contiene 12 parámetros desconocidos pero el sistema es un sistema con 6 grados de libertad (GDL). Estos 6 GDL son los ángulos de Euler (α, β and γ) y el vector de traslación(x, y, z). Los ángulos de Euler describen la orientación de la cámara que tomo la nube de puntos a registrar y el vector de traslación describe la posición de esta misma. Esta transformación T no es un operador lineal debido a que la mayoría de los elementos de la matriz están compuestos por senos y cosenos. Las funciones seno y coseno son funciones que no presentan cambios bruscos en sus valores, por eso el sistema de ecuaciones 3.1 puede ser reordenado en términos de los parámetros α, β, γ, x, y y z. Esto puede se puede realizar linealizando previamente la transformación T. Una vez que la matriz T es linealizada y el sistema de ecuaciones reordenado de tal forma que α, β, γ, x, y y z se conviertan en las nuevas incógnitas, el sistema es resuelto usando un método que iterativamente calcula la transformación usando mínimos cuadrados.

Linealización

Cada elemento t_{ij} de la matriz de transformación T se define como una función no lineal (Ecuación 3.2) con 6 GDL ($\Theta = \{\alpha, \beta, \gamma, x, y, z\}$). Para obtener la expresión en términos de Θ cada uno de los elementos t_{ij} de la matriz T es linealizado [11]. Para realizar esta linealización se usan series de Taylor de primer orden. La serie de Taylor de primer orden alrededor del punto $\Theta_0 = \{\alpha_0, \beta_0, \gamma_0, x_0, y_0, z_0\}$ de cada uno de los elementos t_{ij} se define en la ecuación 3.2

$$t_{ij}(\alpha,\beta,\gamma,x,y,z) = t_{ij}(\Theta_0) + \frac{\partial t_{ij}}{\partial \alpha} d\alpha + \frac{\partial t_{ij}}{\partial \beta} d\beta + \frac{\partial t_{ij}}{\partial \gamma} d\gamma + \frac{\partial t_{ij}}{\partial x} dx + \frac{\partial t_{ij}}{\partial y} dy + \frac{\partial t_{ij}}{\partial z} dz \tag{3.2}$$

donde

$$\begin{array}{lll} d\alpha = \alpha - \alpha_0 & d\beta = \beta - \beta_0 & d\gamma = \gamma - \gamma_0 \\ dx = x - x_0 & dy = y - y_0 & dz = z - z_0 \end{array} \tag{3.3}$$

Una vez definida la linealización, sustituimos la ecuación 3.2 en cada uno de los elementos de la matriz 3.1 y reagrupamos términos de tal forma que el vector de incógnitas pasen a ser las variables $d\alpha$, $d\beta$, $d\gamma$, dx, dy y dz. Este nuevo sistema de ecuaciones esta definido a continuación

$$A \cdot d\Theta = b \Rightarrow$$

$$\begin{pmatrix} \frac{\partial t_{00}}{\partial \alpha} x_i + \frac{\partial t_{01}}{\partial \alpha} y_i + \frac{\partial t_{02}}{\partial \alpha} z_i & \frac{\partial t_{00}}{\partial \beta} x_i + \frac{\partial t_{01}}{\partial \beta} y_i + \frac{\partial t_{02}}{\partial \beta} z_i & \frac{\partial t_{00}}{\partial \gamma} x_i + \frac{\partial t_{01}}{\partial \gamma} y_i + \frac{\partial t_{02}}{\partial \gamma} z_i & 1 & 0 & 0 \\ \frac{\partial t_{10}}{\partial \alpha} x_i + \frac{\partial t_{11}}{\partial \alpha} y_i + \frac{\partial t_{12}}{\partial \alpha} z_i & \frac{\partial t_{10}}{\partial \beta} x_i + \frac{\partial t_{11}}{\partial \beta} y_i + \frac{\partial t_{12}}{\partial \beta} z_i & \frac{\partial t_{10}}{\partial \gamma} x_i + \frac{\partial t_{11}}{\partial \gamma} y_i + \frac{\partial t_{12}}{\partial \gamma} z_i & 0 & 1 & 0 \\ \frac{\partial t_{20}}{\partial \alpha} x_i + \frac{\partial t_{21}}{\partial \alpha} y_i + \frac{\partial t_{22}}{\partial \alpha} z_i & \frac{\partial t_{21}}{\partial \beta} x_i + \frac{\partial t_{22}}{\partial \beta} y_i + \frac{\partial t_{23}}{\partial \beta} z_i & \frac{\partial t_{21}}{\partial \gamma} x_i + \frac{\partial t_{22}}{\partial \gamma} y_i + \frac{\partial t_{23}}{\partial \gamma} z_i & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d\alpha \\ d\beta \\ d\gamma \\ dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} x'_i - t_{00} - t_{01} - f_{02} \\ y'_i - t_{10} - t_{11} - f_{12} \\ z'_i - t_{20} - t_{21} - f_{22} \end{pmatrix} \tag{3.4}$$

El nuevo sistema de ecuaciones 3.4 define las ecuaciones para un emparejamiento i-ésimo ya que el sistema de ecuaciones completo tiene 3 ecuaciones por cada una de las correspondencias del conjunto de emparejamientos sin espurios (una ecuación por cada una de las componentes de los puntos en 3D).

Una vez que se ha calculado la version linealizada del sistema de ecuaciones 3.1, podemos usar métodos de álgebra lineal para iterativamente obtener los valores de la variable Θ.

Mínimos cuadrados

Como fue descrito en la sección anterior, cada una de las coincidencias calculadas añade tres ecuaciones al sistema de ecuaciones 3.4. Por tanto, el sistema esta formado por un número de ecuaciones mucho mayor que el número de incógnitas. Por esa razón, el sistema de ecuaciones es resuelto usando mínimos cuadrados para encontrar el parámetro $d\Theta$ que minimizan las distancia entre las correspondencias de dos nubes de puntos diferentes. La distancia que hemos elegido para minimizar el problema ha sido la distancia euclídea:

$$||[x'_0 - x_0, y'_0 - y_0, z'_0 - z_0, \ \dots \ , x'_n - x_n, y'_n - y_n, z'_n - z_n]^T||^2 \tag{3.5}$$

donde $[x'_i, \ y'_i, \ z'_i]^T$ y $[x_i, \ y_i, \ z_i]^T$ representan las coordenadas de un emparejamiento i-ésimo entre el modelo M y la nube de puntos a registrar D respectivamente.

Ahora podemos reescribir el sistema de ecuaciones 3.4 como un problema de mínimos cuadrados en función de $d\Theta$. Así, la función que queremos minimizar es

$$f(x) = \ ||b - A \cdot d\Theta||^2 \tag{3.6}$$

De acuerdo con el sistema de ecuaciones 3.4, la solucion $d\Theta$ que minimiza la funcion se define como un pequeño paso en la búsqueda del mínimo global. Por lo tanto, el método se ejecuta iterativamente hasta que se encuentra el mínimo global de la función.

Hay que enfatizar que para aplicar el método de mínimos cuadrados se ha realizado una linealización de la matriz T. Como resultado, la solución del método depende fuertemente del punto inicial $\Theta_0 = \{\alpha_0, \beta_0, \gamma_0, t_{x_0}, t_{y_0}, t_{z_0}\}$ usado para la linealización que debería ser un valor cercano al valor optimo deseado, ya que de otro modo podríamos obtener un mínimo local en vez del mínimo global.

Resumiendo, dada una estimación inicial (Θ_0) cercana al valor óptimo (Θ) que queremos obtener, el método de mínimos cuadrados iterativo se describe en los siguientes puntos:

1. la matriz A y el vector b se construyen de acuerdo con el valor del parámetro Θ_k y los puntos de los emparejamientos de los que partíamos, de acuerdo al sistema de ecuaciones 3.4
2. se resuelve el sistema de ecuaciones construido usando mínimos cuadrados
3. la solución obtenida $d\Theta$ (pequeño paso para converger al mínimo global) se añade al valor del parámetro estimado Θ_k
$$\Theta_{k+1} = \Theta_k + d\Theta \tag{3.7}$$
4. si la diferencia entre dos soluciones consecutivas es menor que un umbral definido por el usuario, el algoritmo ha convergido a la solución ($\Theta = \Theta_{k+1}$). Sino, volvemos al paso uno usando como estimación el nuevo parámetro calculado Θ_{k+1}.

3.2. Iterative Closest Point (ICP)

Como alternativa al método basado en puntos de interés descrito en la sección anterior, ahora se va a presentar el Iterative Closest Point. El ICP, como en el método anterior, calcula la matriz de transformación T que mapea una nube de puntos P en otra nube de puntos que llamaremos modelo M. En la Figura 1.5 se puede ver gráficamente el proceso que sigue el método basado en ICP. Además en la sección 1.4.2 se presenta una versión esquematizada de dicho método.

El Iterative Closest Point es un algoritmo que fue propuesto por Besl and McKay en los años 90. Desde que fue propuesto, muchas variedades del ICP han aparecido pero en este proyecto nos centraremos solo en la versión clásica descrita en [2].

Dadas dos nubes de puntos M y $P = \{p_0, ..., p_n\}$, el ICP calcula iterativamente la matriz de transformación T que minimiza el error entre puntos correspondientes de las nubes M y P

$$\underset{T}{\operatorname{argmin}} \left\{ \sum_{i=0}^{n} ||T \cdot p_i - m_i||^2 \right\} \tag{3.8}$$

donde m_i es el vecino más cercano del punto p_i en el modelo M.

La idea principal del ICP clásico (Algoritmo 3) se puede resumir en dos pasos:

1. construir el conjunto de correspondencias basándonos en la distancia más cercana de los puntos de P_i en M usando una estimación inicial T_0.
2. calcular la matriz de transformación T que minimiza la distancia entre las correspondencias (Ecuación 3.8)

Como cualquier método de descenso de gradiente, el ICP converge con una mayor seguridad a una buena solución cuanto mejor sea la estimación de la transformación inicial T_0. De otro modo, es probable que el ICP converga a mínimo local que de como resultado un mal registrado de nubes de puntos.

Algorithm 3 Algoritmo ICP

Require: Modelo M
Nube de puntos a registrar $P = \{p_0, ..., p_q\}$
Transformación inicial T_0
Ensure: Matriz de transformación T que alinea P con M
$T \leftarrow T_0$
while not *converga* **do**
for $i \leftarrow 1$ **to** N **do**
$m_i \leftarrow vecinoMasCercano(T \cdot p_i, M)$
if $||m_i - T \cdot p_i|| \leq dist_{max}$ **then**
$w_i \leftarrow 1$
else
$w_i \leftarrow 0$
end if
end for
$T \leftarrow \operatorname{argmin}_T \left\{ \sum_i w_i ||T \cdot p_i - m_i||^2 \right\}$
end while
return T

El ICP asume que las nubes de puntos a registrar no tienen porque estar completamente solapadas. Por lo tanto, algunos puntos entre las nubes de puntos no tendrán correspondencia. Por esta razón, un umbral ($dist_{max}$) se añade al ICP. De este modo, solo los puntos cuya distancia euclídea con su emparejamiento sea menor que el umbral $dist_{max}$ son considerados como correspondencias. La elección del umbral $dist_{max}$ representa una solución de compromiso entre convergencia y precisión:

- un valor pequeño puede resultar en mala convergencia porque el algoritmo no encuentra suficientes correspondencias para calcular la matriz de transformación T (Figura 3.3).

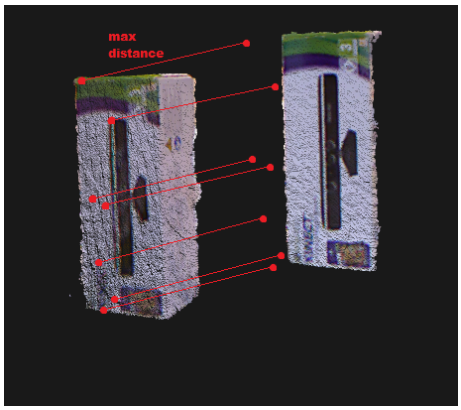

Figura 3.3: Esta figura muestra que si el objeto esta mas lejos que $dist_{max}$, el ICP no puede encontrar correspondencias. Por lo tanto, no puede calcular la matriz de transformacion T.

- valores grandes pueden dar como resultado correspondencias erróneas entre puntos que no estén solapados. De este modo, la transformación calculada no es la óptima (Figura 3.4).

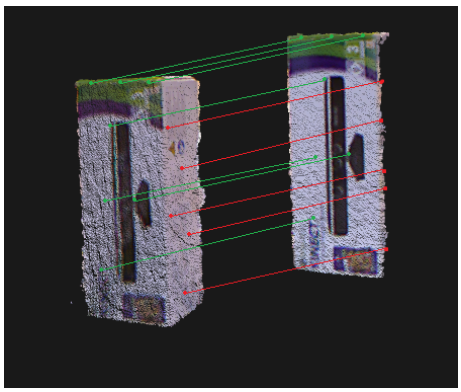

Figura 3.4: Las lineas verdes representan correspondencias correctas. Las lineas rojas describen correspondencias erróneas. Esta figura muestra que si $dist_{max}$ tiene un valor grande el algoritmo encuentra correspondencias entre zonas no solapadas.

La mejor elección de $dist_{max}$ para el proceso de registrado de nube de puntos es el menor valor que da como resultado un buen registrado. La elección depende de cuanto de buena es la estimación de la posición de la cámara. Cuanto mejor sea la estimación, más puede ser el umbral $dist_{max}$.

A la hora de registrar nubes de puntos usando el ICP hay que tener en mente algunos pequeños trucos que mejoran mucho el resultado final. Cuando se calcula la posición de la cámara, se comete un pequeño error. El error entre dos posiciones consecutivas de la cámara es pequeño pero estos errores se van acumulando poco a poco. Como resultado, el error acumulado entre la primera nube de puntos a registrar y la ultima puede ser considerable. Para evitar este fenómeno (Figura 3.5), en vez de transformar la nube de puntos P en el modelo M, lo que hacemos es transformar el modelo M en la nube de puntos P ($M = T^{-1} \cdot M + P$). Así, podemos asignar a $dist_{max}$ un valor mas pequeño.

Ademas, para obtener mejores resultados, el ICP se aplica a nubes de puntos a las que se les ha extraído el entorno en el que se encontraban. Como se ha explicado antes el ICP intenta minimizar la distancia euclídea de las correspondencias de las nubes de puntos a registrar. En comparación con el objeto, el entorno tiene una cantidad mucho más grande de puntos que el objeto. Por lo tanto, podría ocurrir que el entrono se ha alineado perfectamente pero el objeto no.

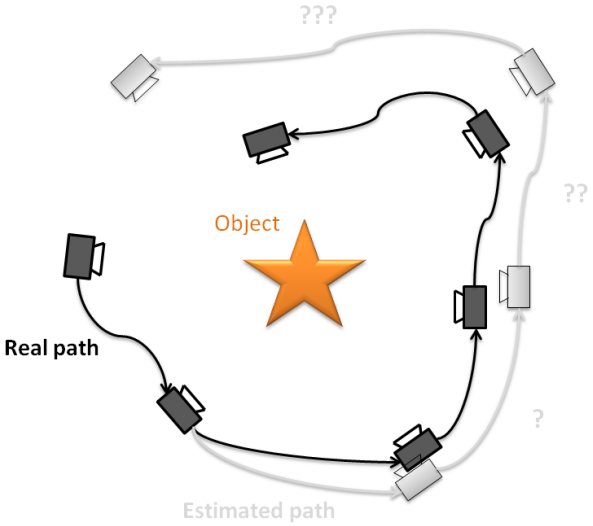

Figura 3.5: Esta figura describe el problema de los errores acumulados en el proceso de estimación de posición de la cámara.

Capítulo 4

Comparación de métodos

En el capítulo 3, se explican dos métodos diferentes para el registrado de nubes de puntos: método basado en puntos de interés y método basado en Iterative Closest Point. El objetivo de esta sección es el de hacer una comparación de ambos métodos desde una perspectiva practica teniendo en cuenta los siguientes puntos:

1. **Precisión.** Este es el punto más importante a la hora de comparar ambos métodos. Para hacer esta comparación de precisión, el modelo del objeto reconstruido por todos los métodos sera testado visualmente para ver las diferencias entre ellos.

2. **Velocidad.** Aunque la velocidad en el proceso no es un punto clave, la eficiencia computacional es una característica que los métodos deberían cumplir y que puede ayudar a decidir cual de los métodos es más adecuado para un determinado objeto.

La comparación de los modelos 3D reconstruidos se realizará para dos objetos diferentes que nos permitirán enumerar los puntos fuertes y débiles de ambos métodos. Cada uno de los dos métodos comparados, tendrá a su vez dos casos de estudio. En el estudio del método basado en puntos de interés un caso se estudia para puntos de interés SURF y otro para puntos de interés ORB. En el método basado en el ICP, el estudio se realiza para nubes cuya resolución se ha disminuido y para nubes cuya resolución no se ha modificado.

4.1. Test 1

Para el primer test, hemos escogido una caja. El modelo en 3D final de la caja para los cuatro casos de estudio, se ha reconstruido utilizando 55 nubes de puntos tomadas desde diferentes puntos de vista. El objeto a reconstruir se encontraba sobre una mesa con ningún objeto más en ella.

Los resultados obtenidos para los cuatros casos de estudio se muestran en la Figura 4.1. En esta figura podemos observar que los casos a), c) y d) convergen a una solución similar y de precisión bastante aceptable. En el único caso en el que el resultado no es tan bueno es en el caso b) esto es debido a que para su reconstrucción fueron utilizadas nubes de puntos con una resolución más baja. El objetivo de usar nubes de puntos con una resolución más baja era el de obtener un modelo en 3D en un menor tiempo y comprobar si obteníamos los mismos resultados que usando nubes de puntos sin disminuir su resolución.

En este punto, centrándonos en la precisión solo un modelo ha sido descartado. Por lo tanto, para decidir cual es el mejor método para este objeto tendremos que analizar la velocidad de reconstrucción del modelo. En la Figura 4.2 se muestra los diferentes tiempos de reconstrucción del modelo. En esta figura podemos ver claramente que el método basado en ICP es mucho más lento que el método basado en puntos de interés. Por lo tanto, podemos afirmar que para objetos con texturas la mejor opción es el método basado en puntos de interés porque es tan preciso como el método basado en ICP pero sin embargo es varias veces más rápido.

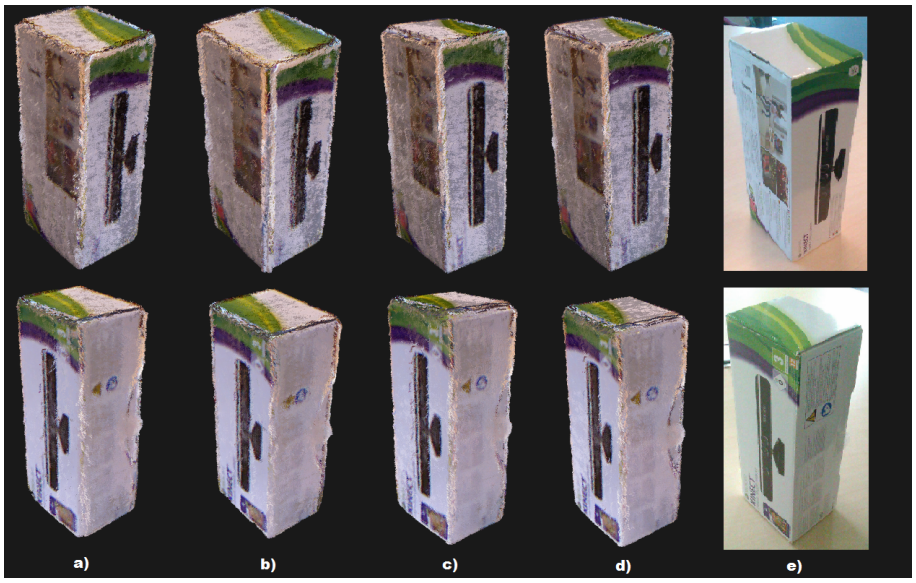

Figura 4.1: a) Resultado usando ICP con nubes de puntos sin disminuir su resolución. b) Resultado usando ICP con nubes de puntos con disminución de resolución. c) Resultado usando el método basado en puntos de interés SURF d) Resultado usando el método basado en puntos de interés ORB. e) Objeto real.

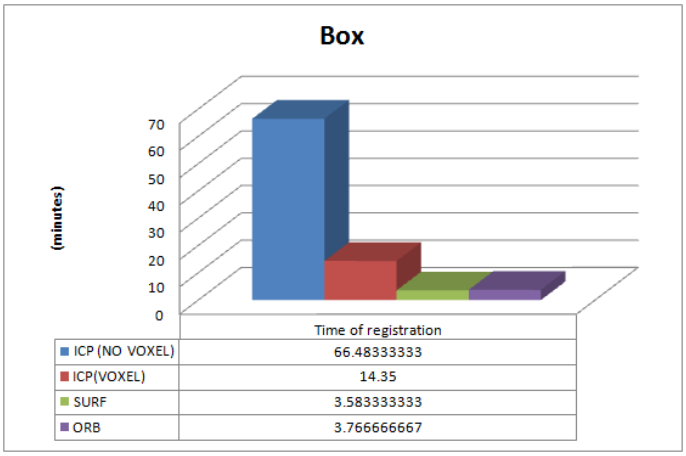

Figura 4.2: Esta figura muestra gráficamente la diferencia de tiempo requerido para para reconstruir un modelo para cada uno de los métodos y casos de estudio.

4.2. Test 2

Para el segundo test, un libro fue elegido. El modelo en 3D final del libro se ha reconstruido utilizando 55 nubes de puntos tomadas desde diferentes puntos de vista. El objeto a reconstruir se encontraba sobre una mesa con ningún objeto más en ella.

En este caso, el libro no tenía suficientes puntos de interés como para ejecutar el método basado en puntos de interés. Por esa razón solo el método basado en ICP fue estudiado. Los resultados obtenidos para el método basado en el ICP se muestran en la Figura 4.3. Como ocurrió en el test 1, el ICP aplicado a nubes de puntos sin disminución de resolución obtiene resultados mucho más precisos. Pero, como se ve en la Figura 4.4, el ICP aplicado a nubes de puntos con una resolución disminuida es varias veces más rápido. Por lo tanto, la decisión de que ICP debería ser utilizado depende de que es más importante: precisión o rapidez. En nuestro caso como lo que buscamos es precisión en el modelo, seleccionaríamos el ICP que utiliza nubes de puntos sin disminución de resolución.

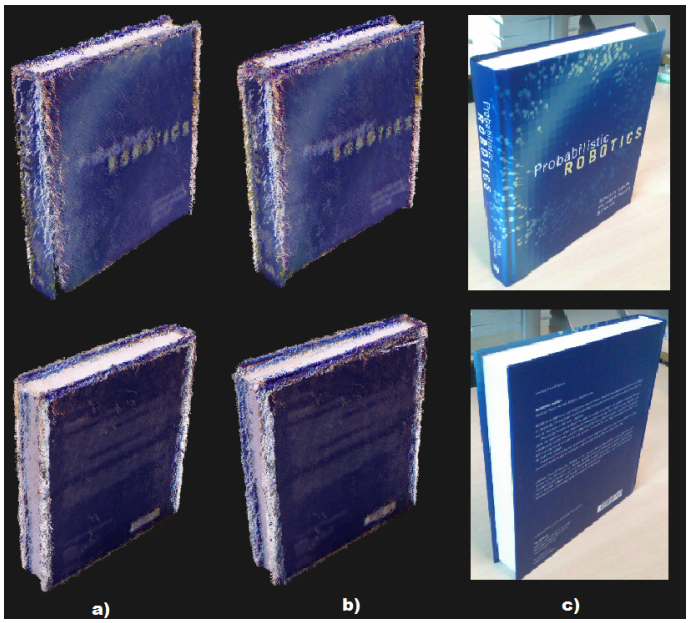

Figura 4.3: a) Resultado usando ICP con nubes de puntos sin disminuir su resolución. b) Resultado usando ICP con nubes de puntos con disminución de resolución. c) Objeto real.

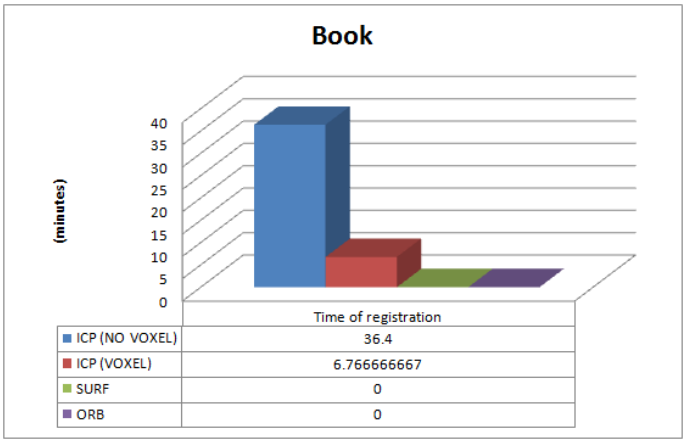

Figura 4.4: Esta figura muestra gráficamente la diferencia de tiempo requerido para para reconstruir un modelo con el método basado en ICP.

Capítulo 5

Conclusiones y recomendaciones

El objetivo de esta memoria era el de estudiar la mejor manera de crear modelos en 3D de objetos. Para eso, en esta memoria se ha descrito paso a paso el proceso que una nube de puntos P sigue para ser registrada con un modelo M. Además, se ha dado una alternativa al conocido algoritmo Iterative Closest Point para el registrado de nubes de puntos. El método alternativo es un método basado en puntos de interés. En este último capítulo, se van a desarrollar las conclusiones obtenidas tras el estudio e investigación de ambos métodos. Además de estas conclusiones, también se darán algunas recomendaciones que podrían mejorar el proceso de registrado de nubes de puntos para ambos métodos.

5.1. Conclusiones

El incremento de la capacidad computacional y de la posibilidad de encontrar sensores de bajo coste han impulsado el desarrollo de nuevas aplicaciones como la de crear modelos en 3D. Sin embargo, estos sensores no son perfectos. Para el desarrollo de este proyecto el sensor elegido fue la kinect, que lamentablemente no es una excepción. La kinect tiene problemas con el mapeo de la información de profundidad y la información RGB, es decir, hay un pequeño desfase entre ellas que hace que la información RGB sea errónea en algunas zonas de las nubes puntos como los bordes. Esto hace que, los modelos reconstruidos no sean tan precisos como nos gustaría. Además, la kinect también tiene problemas detectando la profundidad en zonas donde hay una variación grande de profundidad. Esto provoca igualmente puntos erróneos que afectan al resultado final del modelo.

Otro problema a destacar debido al uso de la kinect es que esta no se puede usar para obtener modelos de objetos translucidos o reflectantes. Esto es debido a que la kinect no puede estimar la profundidad en estos tipos de objetos, limitando el rango de objetos que podemos reconstruir a los no reflectantes y/o no translucidos.

Partiendo de la idea de que la kinect no es perfecta y de que no podemos obtener modelos en 3D perfectos, vamos a dar una visión de los dos métodos estudiados en esta memoria. La conclusión principal viendo los resultados obtenidos es que los métodos estudiados no son sustitutivos uno del otro, es decir no hay uno que funcione mejor para todos los objetos. En la sección de resultados hemos visto que cuando el objeto a reconstruir tiene textura, el método basado en puntos de interés obtiene mejores resultados, ya que, el resultado final es igual o más preciso que en el caso del ICP y además el proceso es varias veces más rápido. Sin embargo, para objetos que no tienen textura, el método basado en puntos de interés no se puede aplicar por lo que el método basado en el ICP es la única solución aunque el proceso de reconstrucción sea varias veces más lento.

Como hemos explicado los métodos estudiados en esta memoria no son sustitutivos unos de otros, ya que el método basado en puntos de interés no puede ser aplicado a todos los objetos. Sin embargo, este método puede ser usado incluso cuando la estimación inicial de la cámara que toma las nubes de puntos no es precisa (5cm-15cm), mientras que el ICP solo converge cuando la estimación de la posición de la cámara es muy precisa. Además, aunque no se ha estudiado el caso, usando el método basado en puntos de interés, podríamos reconstruir los modelos sin estimación de posición de la cámara si el desplazamiento

de la cámara fuera lento y la tasa de nubes de puntos por segundo fuera suficientemente alta. Realizar esto con el ICP podría ser también posible pero sería totalmente inviable en términos de tiempo de ejecución.

5.2. Recomendaciones

Aunque los resultado obtenidos de este proyecto parecen prometedores, en esta sección vamos a plantear algunas acciones que en una futura investigación podrían ayudar a mejorar la precisión y/o el rendimiento de ambos métodos. Los puntos que se podrían mejorar son los siguiente:

- **Versiones más eficientes del ICP.** En el ámbito de este proyecto solo nos hemos centrado en el estudio del Iterative Closest Point clásico que como se ha visto aunque consigue resultados aceptables, es tremendamente ineficiente en términos de tiempo de ejecución. Por eso se podrían utilizar alguna de las variantes existentes del ICP. Algunas de estas variantes son: ICP point-to-plane o el Generalizaed-ICP [15]. A priori, el Generalized-ICP parece una variante prometedora que podría mejorar el tiempo de ejecución y la precisión del modelo final.

- **Loop closure.** Los resultados que hemos obtenido con el método basado en puntos de interés son satisfactorios, pero estos también pueden ser mejorados. En el proceso de registrado de dos nubes de puntos se comete un pequeño error. Nuestro propósito es registrar iterativamente un número de nubes de puntos con un tamaño considerable. Este hecho provoca que el pequeño error cometido a la hora de registrar dos nubes de puntos se vaya acumulando a lo largo del proceso iterativo de reconstrucción del modelo. Para mejorar el resultado final se podría aplicar la técnica del loop closure. La idea básica de esta técnica es detectar cuando hemos rodeado por completo el objeto a reconstruir. Al detectar que hemos rodeado el objeto, se podrían actualizar las transformaciones calculadas para corregir el error acumulado en el proceso iterativo de registrado de las nubes de puntos.

- **Intentar minimizar las imprecisiones en los bordes.** Uno de las razones por la que la precisión del modelo reconstruido no es tan buena como nos gustaría se debe a la imprecisión de la kinect en los bordes. Estas imprecisiones hacen que en los bordes de los objetos aparezcan zonas adyacentes a los bordes que deberían estar a una profundidad diferente. Esto hace que nuestros modelos tengan valores RGB erróneos en estas áreas. Intentar reducir el efecto de este problema resultaría en modelos mucho más precisos. Una idea a estudiar podría ser la de jugar con los contornos de la imagen RGB y con la imagen de profundidad para detectar las zonas problemáticas y eliminarlas.

Bibliografía

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, *Surf: Speeded up robust features*, Computer Vision–ECCV 2006, Springer, 2006, pp. 404–417.

[2] Paul J Besl and Neil D McKay, *Method for registration of 3-d shapes*, Robotics-DL tentative, International Society for Optics and Photonics, 1992, pp. 586–606.

[3] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, *Brief: binary robust independent elementary features*, Computer Vision–ECCV 2010, Springer, 2010, pp. 778–792.

[4] Franklin C Crow, *Summed-area tables for texture mapping*, ACM SIGGRAPH Computer Graphics, vol. 18, ACM, 1984, pp. 207–212.

[5] Konstantinos G Derpanis, *Integral image-based representations*, Department of Computer Science and Engineering York University Paper **1** (2007), no. 2, 1–6.

[6] 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** (1981), no. 6, 381–395.

[7] Armin Gruen and Devrim Akca, *Least squares 3d surface and curve matching*, ISPRS Journal of Photogrammetry and Remote Sensing **59** (2005), no. 3, 151–174.

[8] John C Hart, George K Francis, and Louis H Kauffman, *Visualizing quaternion rotation*, ACM Transactions on Graphics (TOG) **13** (1994), no. 3, 256–276.

[9] Georg Klein and David Murray, *Parallel tracking and mapping for small ar workspaces*, Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, IEEE, 2007, pp. 225–234.

[10] Alexander Neubeck and Luc Van Gool, *Efficient non-maximum suppression*, Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, vol. 3, IEEE, pp. 850–855.

[11] Andreas Nüchter, Jan Elseberg, Peter Schneider, and Dietrich Paulus, *Study of parameterizations for the rigid body transformations of the scan registration problem*, Computer Vision and Image Understanding **114** (2010), no. 8, 963–980.

[12] Edward Rosten and Tom Drummond, *Machine learning for high-speed corner detection*, Computer Vision–ECCV 2006, Springer, 2006, pp. 430–443.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, *Orb: an efficient alternative to sift or surf*, Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2564–2571.

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

[15] A Segal, Dirk Haehnel, and Sebastian Thrun, *Generalized-icp*, Proc. of Robotics: Science and Systems (RSS), vol. 25, 2009, pp. 26–27.

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

[17] Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Haussermann, Rob Janssen, JMM Montiel, Alexander Perzylo, et al., *Roboearth*, Robotics & Automation Magazine, IEEE **18** (2011), no. 2, 69–82.

[18] John R Williams and Kevin Amaratunga, *Introduction to wavelets in engineering*, International journal for numerical methods in engineering **37** (1994), no. 14, 2365–2388.