_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MEJORAS ESCÁNER 3D DE OBJETOS

Manuel Luna Aleixos

Tutor: Antonio Albiol Colomer

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 4 de julio de 2016

Resumen

El objeto de estudio de este proyecto consiste en obtener un modelo 3D completo de un objeto a partir de modelos parciales obtenidos desde distintos puntos de vista, utilizando para ello una cámara web y un proyector.

Para conseguir el modelo 3D completo se necesita estimar una matriz de rotación-translación relativa a cada nube de puntos.

Previamente, será necesario obtener los puntos característicos, así como las componentes normales, además de otras características necesarias para poder realizar los emparejamientos entre las nubes de puntos.

Una vez obtenidos los emparejamientos, habrá que descartar aquellos que sean erróneos o puedan dar lugar a confusión. Estos errores se deben a ruido o a una mala estimación a la hora de realizar los emparejamientos.

Finalmente, una vez se hayan seleccionado los emparejamientos considerados como óptimos, se procederá a calcular las matrices de rotación-translación y aplicarlas a las distintas nubes de puntos para obtener el objeto completo en 3D.

Resum

L'objectiu d'aquest projecte és obtindre un model en 3D complet d'un objecte a partir de models parcials obtinguts des de distints punts de vista, emprant una càmara web i un projector.

Per aconseguir el model 3D complet cal estimar una matriu de rotació-traslació relativa a cada núvol de punts.

Previament caldrà obtindre els punts característics, així com les components normals, a més a més d'altres característiques necessàries per poder realitzar aparellaments entre els núvols de punts.

Una vegada obtinguts els aparellaments, s'haurà de descartar aquells que siguen erronis o que puguen donar lloc a confusió. Aquests errors es deuen a confusions o a una mala estimació a l'hora de realitzar els aparellaments.

Finalment, una vegada s'han seleccionat els aparellaments considerats com a òptims, es procedirà a calcular les matrius de rotació-traslació i a aplicar-les als distints núvols de punts per a obtindre l'objecte complet en 3D.

Abstract

The aim of this project is to achieve a whole 3D model of an object from partial models obtained from several points of view, using for it a web camera and a projector.

In order to obtain the complete 3D model is necessary to estimate a rotation-translation matrix relative to each point cloud.

Previously, it will be necessary to obtain the keypoints, as well as the normal components and other necessary features in order to match the point clouds.

After match the point clouds, it will be necessary discard the matches which are wrong or can cause confusion. These mistakes are due to noise or a bad estimation when the matches were made.

Finally, after select the best matches, it proceeds to calculate the rotation-translation matrix and apply them to the several point clouds in order to achieve the whole object in 3D.

Índice

Capítulo 1. Introducción

La realización de este proyecto está basado en la continuación del proyecto realizado por Anna Arias i Duart y Raúl Payá Llorens en el curso 2014-2015, donde se logró obtener una nube de puntos a partir de la obtención de imágenes en 2D empleando una cámara web y un proyector.

1.1 Trabajo Previo

La primera parte del proyecto consiste en calibrar tanto la cámara como el proyector a utilizar. Lo que se pretende es conseguir un modelo de la geometría de la cámara así como un modelo de distorsión de la lente. Estos dos modelos serán los llamados parámetros intrínsecos de la cámara. También se obtendrán los parámetros intrínsecos del proyector, entendiendo el proyector como el sistema inverso de la cámara, ya que la luz viaja en el sentido contrario al de la cámara.

Una vez obtenidos los parámetros intrínsecos tanto del proyector como de la cámara se deberá calcular los parámetros extrínsecos, de esta manera se podrá saber cuál es la matriz de rotación y el vector de translación que indican cómo están rotados y trasladados los ejes del proyector respecto de la cámara.

La segunda parte se centra en la obtención una colección de imágenes usando luz estructurada. Para ello se ilumina el objeto a escanear mediante la luz que proyecta el escáner y son captadas por la cámara. En base a la posición de estas franjas se estiman las coordenadas en 3D de los puntos. Estas coordenadas en 3D se consiguen obteniendo la ecuación del rayo de luz de un píxel empleando una imagen obtenida por la cámara y cuando este rayo intersecta con el plano del proyector al cuál el píxel pertenece. Este proceso se repite para cada píxel de la imagen.

Figura 1 – Colección de imágenes usando luz estructurada aplicando código gray.

1.2 Estructura de la memoria

El objetivo de este trabajo es conseguir obtener un objeto completo en 3D, ya que hasta ahora sólo se podía obtener una parte del objeto a escanear.

Antes de comenzar con la descripción detalla del proyecto se va a realizar una breve explicación del proceso seguido para la elaboración de este trabajo, así el lector podrá tener una idea general del desarrollo empleado para obtener el resultado final.

Realizar un modelo completo en 3D con varias nubes de puntos es lo que se conoce como

registro (registration). Este registro consiste en encontrar las posiciones relativas de distintas vistas en una coordenada global, de forma que las distintas nubes de puntos puedan intersectarse y sobrepongan sus puntos perfectamente. Cada nube de puntos tiene lo que se conoce como puntos característicos (keypoints). Un keypoint es aquel punto que es claramente distintivo de sus vecinos. Sus características, que son invariantes a rotaciones y translaciones, permiten que éste se repita en distintas nubes de puntos, convirtiéndose así en un punto de unión entre dos nubes distintas y, por tanto, haciéndose posible la conformación de un objeto en 3D.

Toda la elaboración de este trabajo ha sido realizado utilizando la librería PCL (Point Cloud Library).

Figura 2 – Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 3 – Resultado de la unión de las nubes de puntos de la *Figura 2*

A continuación, se describirá con detalle cada apartado.

- En primer lugar, se deben identificar los keypoints que mejor representen el conjunto de los datos para cada nube de puntos.

- Una vez obtenido los keypoints, se estimarán las características de los descriptores.

- A continuación, para cada XYZ de la nube de puntos y empleando los descriptores se estimaran las correspondencias, que están basadas en las similaridades que se obtienen de los descriptores.

- Obviamente, se obtendrán errores que no interesan, ya que todas las correspondencias no son buenas. Por tanto, habrá que eliminar todas las correspondencias que contribuyan a una estimación errónea del proceso realizado.

- Finalmente, habrá que estimar una rotación y translación de una nube de puntos para poder sobreponerla sobre la otra nube de puntos.

Para una mejor comprensión de los pasos computacionales descritos anteriormente se puede visualizar el diagrama mostrado en la *Figura 4*.

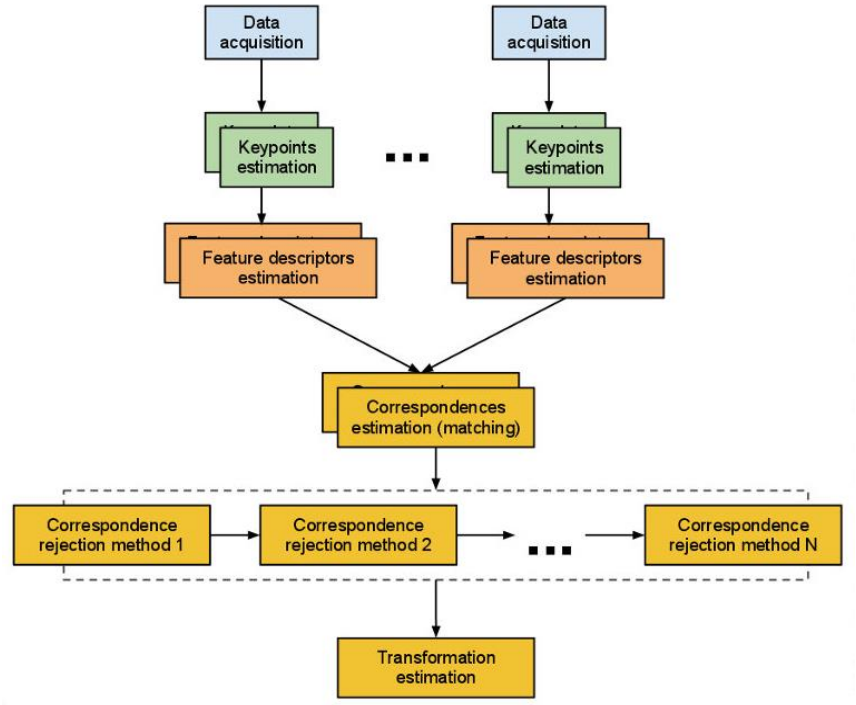


Figura 4 – Diagrama de flujo del desarrollo del proyecto

Capítulo 2. Desarrollo del proyecto

Este capítulo se divide en diversos apartados. La realización consecutiva de todos ellos se conoce como ICP (Iterative Closest Point). ICP sirve para minimizar las diferencias entre dos nubes de puntos, de forma que se pueda realizar una reconstrucción de superficies en 2D o 3D. Este procedimiento se puede ejecutar una única vez o varias veces, dependiendo del resultado final que se quiera obtener.

2.1 Keypoints

Este módulo consiste en encontrar puntos característicos estables, distintivos y que puedan ser identificados aplicando bien los criterios de detección en una nube de puntos. Además, estos puntos deben tener la propiedad de repetitividad, con lo que permiten identificarlos en diferentes nubes de puntos. De esta forma, pueden ser emparejados correctamente y a la hora de realizar la superposición de las nubes de puntos éstos cuadran perfectamente y la composición del resultado de la unión de las dos nubes es óptima.

Obviamente, el número de keypoints es mucho menor que el total de puntos que componen una nube de puntos. Cuando se consigue estimar una buena selección de keypoints en una nube de puntos en combinación con los descriptores de cada keypoint se consigue una representación "compacta" de la nube de puntos. Esto es así debido a que con pocos puntos estamos indicando que es lo más importante o característico de la nube de puntos.

En la librería PCL se consideran característicos los puntos que son repetibles y distintivos.

- Puntos Repetibles (Repeatable): Son puntos que se obtienen con respecto a pequeñas variaciones sobre un punto de vista, ruido, etc. Si se realizara una obtención de varias nubes de puntos con respecto a una misma vista con muy poca variación en cuanto al ángulo de rotación, habrían varios puntos que se repetirían en las distintas nubes de puntos obtenidas. Son también conocidos como puntos locales.

- Puntos Distintivos (Distinctiveness): Son puntos adecuados para una efectiva descripción y emparejamiento. Estos puntos deben ser altamente caracterizables y definidos. Son puntos globalmente definidos.

Para la obtención de los keypoints se ha usado de la librería pcl_keypoints las clases UniformSampling e ISSKeypoint3D.

Hay que ser conscientes de la cantidad de puntos que componen una nube de puntos, por lo que hay muchos puntos que nos interesa despreciarlos cuanto antes para intentar aligerar el cálculo computacional que se va a realizar más tarde.

Esto mismo es lo que realiza UniformSampling, que emplea un método conocido como reducción de la resolución, realizado vía *voxelization* (se describe como un pixel en 3D, que es básicamente un cubo). Éste método consiste en dividir en múltiples regiones en forma de cubo con la resolución deseada. Se comienza con un cubo que encapsula a todos los puntos, después, se subdivide por un factor de 2 donde se incrementa la resolución del cubo debido a que cada vez el tamaño del cubo es más pequeño. De este modo, se puede dar más resolución a ciertas zonas.

Para un mejor entendimiento de este proceso se puede visualizar la figura siguiente obtenida de la página PCL.

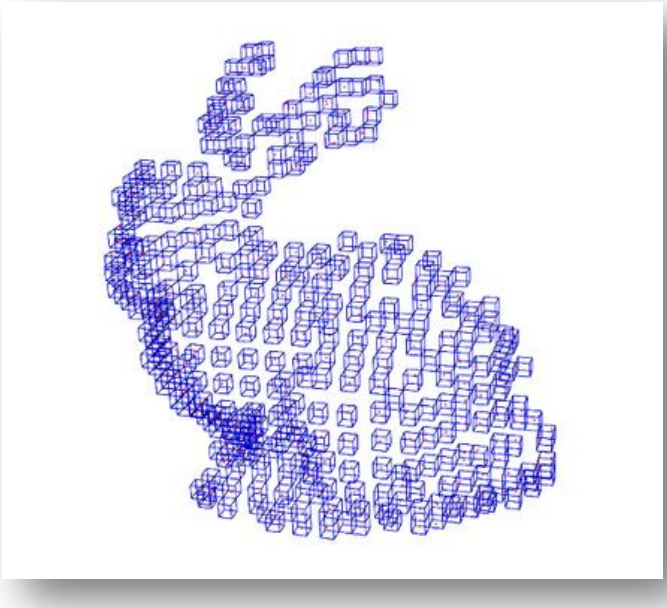


Figura 5 – Visualización del proceso de voxelization

Una vez descartados los puntos que no se van a considerar como puntos característicos

pasamos a utilizar un modelo de descriptor de formas 3D llamado Firma de Forma Intrínseca (ISS, Intrinsic Shape Signature). El método ISS consiste en un sistema de referencia intrínseco que permite un registro rápido de la colocación del objeto, del cual se extraen los keypoints. Utiliza la magnitud del valor propio más pequeño (de esta manera se incluyen sólo puntos con grandes variaciones a lo largo de cada dirección principal) y la relación entre dos valores propios sucesivos (así se consigue eliminar los puntos que tienen una propagación similar a lo largo de direcciones principales).

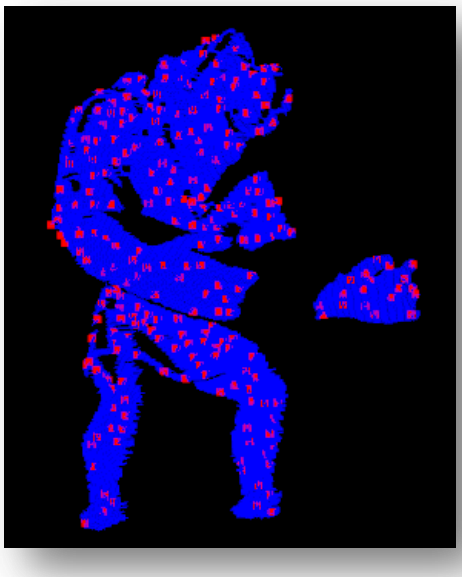


Figura 6 – Visualización de los puntos característicos sobre una nube de puntos

2.2 Características de los descriptores

Los descriptores sirven para obtener las características de los keypoints. Por ello, al realizar las correspondencias se tendrá que tener en cuenta los descriptores para emparejar los puntos que tengan características parecidas, suponiendo por tanto que son el mismo keypoint. En definitiva, los descriptores realmente describen como es la geometría del objeto alrededor del punto que están analizando.

2.2.1 Componentes Normales

Para poder obtener los descriptores, hay que calcular previamente las componentes normales de todos los puntos. Como es bien sabido, la componente normal de un plano es un vector unitario que es perpendicular a este plano. En tres dimensiones, la componente normal de un

punto *P* es un vector perpendicular a la tangente del plano de la superficie de *P*. Esto nos sirve para poder añadir una característica a tener en cuenta por nuestro descriptor, aunque no sea muy descriptiva nos puede servir para discriminar las correspondencias entre dos puntos. Además, las componentes normales pueden darnos información sobre la curvatura de una superficie en algunos puntos, por tanto, sí que puede ser realmente útil su utilización a la hora de estimar los descriptores. Así pues, muchas de las clases para obtener los descriptores permiten incorporar las nubes de puntos de las componentes normales.

Figura 7 – Obtención de las componentes normales para un objeto

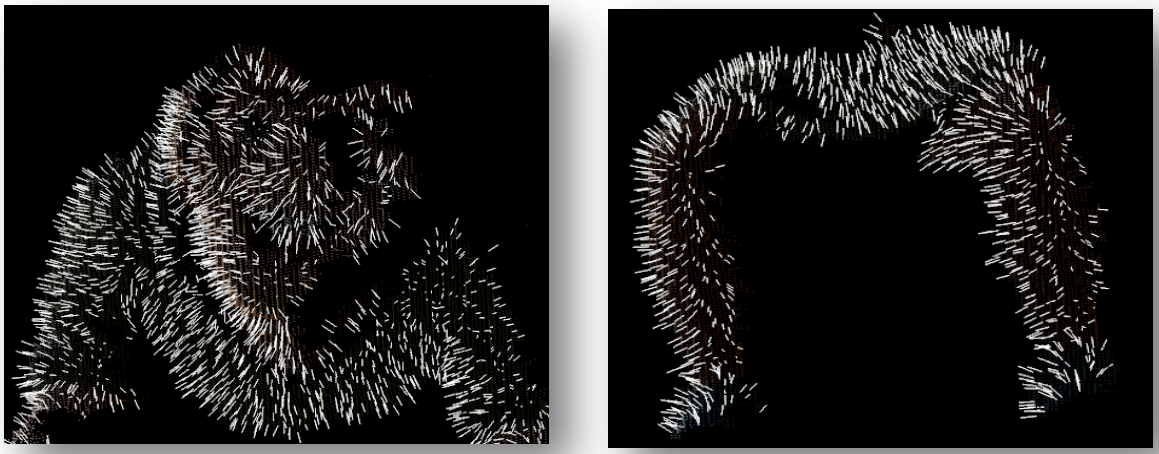


Figura 8 – Zoom de la zona superior (imagen izquierda) y de la zona inferior (imagen derecha) de la *Figura 7*

La librería PCL ofrece la posibilidad de estimar las componentes normales de cada punto. El método empleado para ello ha sido el método nearest neighbor. Éste método estima la componente normal de cada punto, considerando para ello los k puntos vecinos de la nube de puntos más próximos a uno dado. Así pues, dado k vecinos, es posible estimar la componente normal de un punto calculando la matriz de covarianza de las coordenadas XYZ.

2.2.2 Local Reference Frame

Otra característica a tener en cuenta en los descriptores son los marcos de referencia local (Local Reference Frame). Los marcos de referencia local son un sistema de coordenadas o marco de referencia que sólo trabaja sobre una pequeña región restringida del espacio o espacio-tiempo. Esta característica es realmente importante, ya que se utilizará tanto para la obtención de los descriptores como para la obtención de las matrices de rotación y translación de las nubes de puntos.

El objetivo de los marcos de referencia local es conseguir hacer que un punto sea invariante frente a las rotaciones y translaciones, además de ser robusto frente al ruido y al ocultamiento.

Se definen múltiples marcos de referencias locales y, por tanto, múltiples descripciones para el mismo punto. Esto tiene como contra que el proceso se hace menos eficiente, ya que existen más descriptores que deben ser computados y emparejados. Sin embargo, la mejora del resultado aumenta significativamente dado que aunque los keypoints, pertenecientes a distintas nubes de puntos, no tengan ningún sistema de referencia sobre el que situarse en el espacio, el Local Reference Frame permite que sean referenciados en un eje de coordenadas local. Por tanto, las características de los puntos y el emparejamiento se hace mucho más robusto.

2.2.3 Descriptor SHOT

Como se ha comentado anteriormente, los descriptores describen como es la geometría del objeto alrededor del punto que están analizando, sin importar el ruido, la resolución o las transformaciones. Para obtener el descriptor se ha empleado la clase SHOTEstimation de la librería PCL. Esta clase estima la orientación de la firma del histograma (SHOT) para una nube de puntos dada, conteniendo como mínimo para su cálculo los puntos que contiene y sus componentes normales. Además, usa paralelamente el estándar OpenMP (Open Multi-Processing), que es una interfaz de programación de aplicaciones (API) que soporta multiplataformas de programación multiacceso de memoria compartida.

El uso de histogramas es lo que hace que el efecto de filtrado logre la robustez necesaria frente al ruido. Después de haber definido un LRF (Local Reference Frame) único y robusto, es posible mejorar el poder de discriminación del descriptor mediante la introducción de la información geométrica relacionada con la ubicación de los puntos dentro del soporte, imitando así una "firma". Esto se realiza calculando primeramente un conjunto de histogramas locales sobre los volúmenes 3D definidos por una rejilla 3D y, a continuación, agrupando todos los histogramas locales para formar el descriptor real.

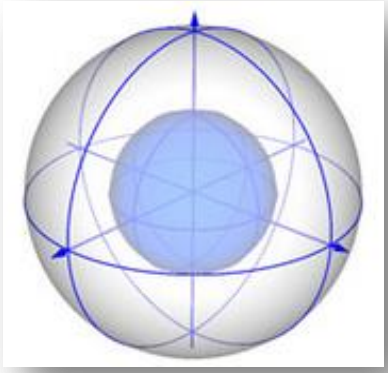


Figura 9 – Estructura de computación para SHOT

SHOT realiza una "firma" para cada keypoint, de forma que en el momento de realizar el emparejamiento entre dos nubes de puntos, va comparando las firmas de cada punto característico, y emparejando las que son más parecidas. Así pues, es importante aportar toda la información posible sobre los keypoint para que las firmas entre ellos sean lo más diferentes posibles, de forma que al emparejar dos nubes de puntos no haya confusión.

Como estructura de la firma se usa una rejilla esférica isotrópica que abarca particiones a lo largo del radio, el acimut y los ejes de elevación.

En particular, los experimentos indican que 32 es el número adecuado de contenedores espaciales para dividir el volumen de la rejilla esférica isotrópica, siendo el resultado de 8 divisiones del acimut, 2 divisiones de elevación y 2 divisiones radiales. Sin embargo, en la *Figura 9* se observa que el resultado es de 4 divisiones de acimut, 2 divisiones de elevación y 2 divisiones radiales para hacer la visualización más sencilla.

Para lograr la robustez a las variaciones de la densidad de puntos, se normaliza el descriptor a 1.

2.3 Correspondencias

Las correspondencias están formadas por los índices de emparejamientos realizados a través de la búsqueda de los puntos más cercanos entre dos nubes de puntos y de la distancia que los separa. Como condición se ha tenido en cuenta que la distancia sea estrictamente inferior a 0.25, sabiendo que las distancias para el descriptor empleado varían desde 0 hasta 1.

Todo el proceso que se va a explicar a continuación trata sobre la búsqueda del punto más cercano, que se emplea para encontrar las correspondencias entre dos nubes de puntos.

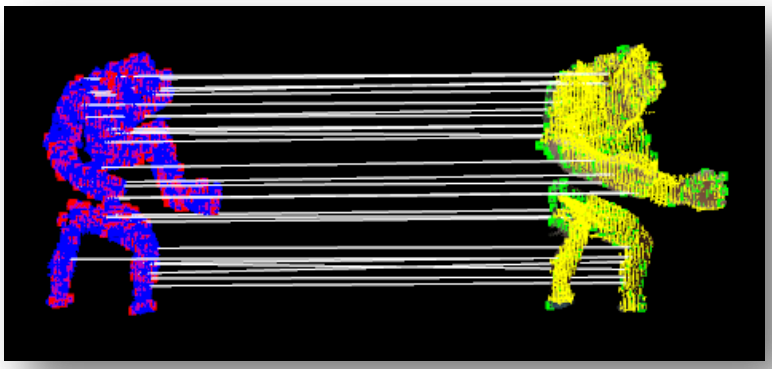


Figura 10 – Correspondencias obtenidas entre dos nubes de puntos

2.3.1 Kd-tree

K – dimensional tree también llamado kd-tree (árbol de k dimensiones) es una estructura de datos de particionado del espacio que organiza los puntos en un espacio euclídeo de k dimensiones. Realiza una estructura de datos muy útil que nos permite realizar búsquedas en un espacio multidimensional, así pues, la búsqueda del vecino más próximo sobre un punto conocido P se realiza de manera rápida y sencilla.

En definitiva, Kd-tree es una generalización de la búsqueda en árbol binaria en k dimensiones del espacio.

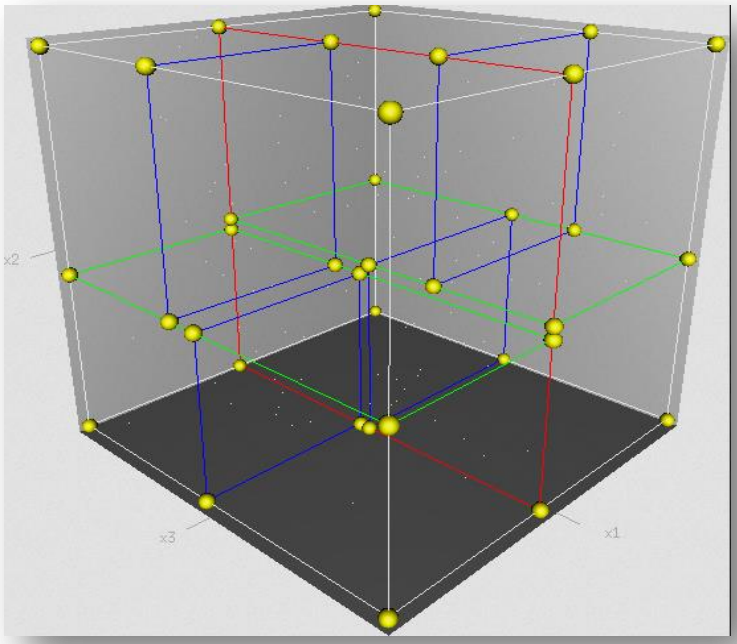


Figura 11 – Estructura de división de kdtree para 3D

Conforme se va descendiendo en el árbol el funcionamiento consiste en emplear ciclos a través de los ejes para seleccionar los planos. De esta manera, si por ejemplo la raíz tiene el plano alineado con el eje *x*, sus descendientes tendrían los planos alineados con el eje *y*, y a su vez, los descendientes de éste estarían alineados con el eje *z*.

Para saber cuál es el elemento raíz en cada descendencia el plano de corte será la mediana de los puntos que componen el kd-tree.

Debido a que en 3 dimensiones es complicado explicar la búsqueda del vecino más próximo, la explicación se realizará sobre 2 dimensiones. En el programa se ha empleado la clase KdTreeFLANN, ésta emplea la estructura kd-tree utilizando FLANN (Fast Library for Approximate Nearest Neighbor).

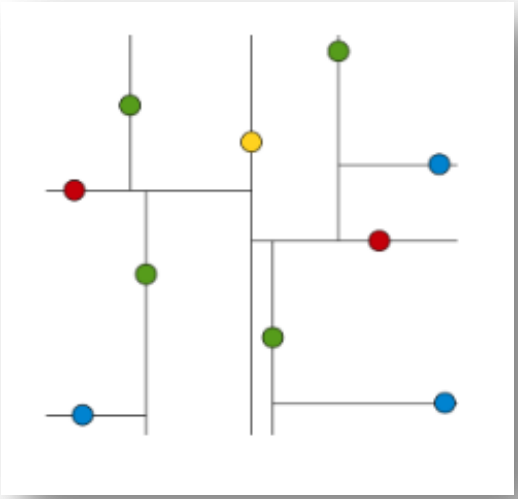


Figura 12 – Estructura kdtree

En la *Figura 12* se observa la estructura en árbol para una serie de puntos. El punto amarillo corresponde al nodo raíz. Éste nodo divide el plano x en dos.

Los puntos rojos son los descendientes del nodo raíz. Siguiendo la descendencia del árbol, se observa que los puntos verdes son descendientes de los puntos rojos. Finalmente, los últimos puntos del árbol son los puntos azules.

El procedimiento para encontrar el vecino más próximo a partir de un punto dado es:

1.- Se comienza por el nodo raíz, el algoritmo va descendiendo el árbol recursivamente.

2.- Una vez el algoritmo alcanza el nodo descendiente, éste se guarda como el “mejor nodo”.

3.- El algoritmo desenvuelve la recursividad del árbol, realizando los siguientes pasos:

3.1.- Si el nodo actual está más cerca del punto dado que el mejor nodo, entonces el nodo actual se convierte en el mejor nodo.

3.2.- El algoritmo comprueba si puede haber algún otro punto en los distintos planos de división que están más cerca del punto de búsqueda a partir del mejor nodo. Esto se hace por la intersección de división del hiperplano con una hiperesfera alrededor del punto de búsqueda que tiene como radio la distancia al mejor nodo. Desde los hiperplanos, se

implementan todas las coordenadas de los ejes en una simple comparación para comprobar si la distancia entre las divisiones de coordenadas al punto de búsqueda son menores que la distancia existente al mejor nodo.

3.2.1.- Si la hiperesfera cruza el plano significa que pueden haber puntos cercanos en el otro lado del plano, de forma y manera que el algoritmo debe seguir descendiendo a la otra rama del árbol en busca de puntos más estrechos, siguiendo el mismo proceso recursivo en toda la búsqueda.

3.2.2.- Si la hiperesfera no intersecta el plano de división, entonces el algoritmo continúa subiendo por el árbol, y la rama entera en el otro lado del nodo es eliminada. Sin embargo, el algoritmo debería estimar el otro lado del hiperplano para determinar si realmente existe o no puntos más cercanos, siguiendo el proceso de recursividad. Ya que existe la posibilidad de que aunque no se corte el plano de división podría haber un punto más cercano al otro lado del punto de división.

4.- Cuando el algoritmo acaba este proceso en el nodo raíz, entonces la búsqueda ha sido completada.

En las siguientes figuras de este apartado se puede visualizar la búsqueda del vecino más cercano dado un punto P (punto de color morado).

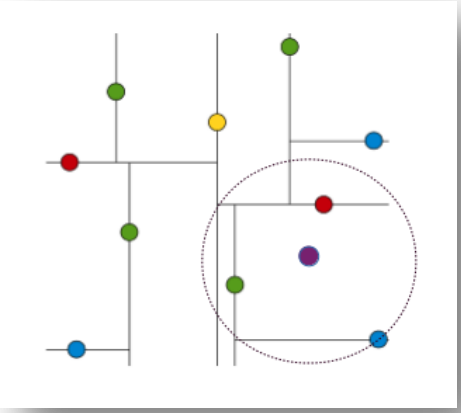


Figura 13 – Procedimiento 2 para encontrar el vecino más próximo

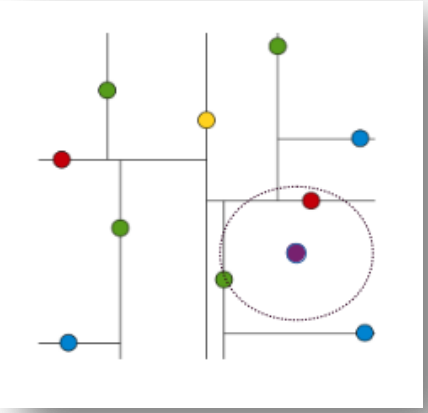


Figura 14 – Procedimiento 3.1 para encontrar el vecino más próximo

Si se visualiza la *Figura 14* se observa como el punto rojo está más cerca del punto morado y además, la circunferencia está conteniendo a otros planos de intersección. Por tanto, hay que aplicar los otros procedimientos pertenecientes al punto 3.2.

Antes de llegar al nodo raíz para finalizar el procedimiento, se evalúa la distancia al nodo de color rojo. Siendo éste el nodo más próximo al punto dado. Así pues, éste se guardará como mejor nodo y se continuará con el procedimiento de los puntos 3.2, puesto que en la *Figura 15* se visualiza como la circunferencia a partir del punto dado (punto de color morado) está intersectando a los otros planos de división.

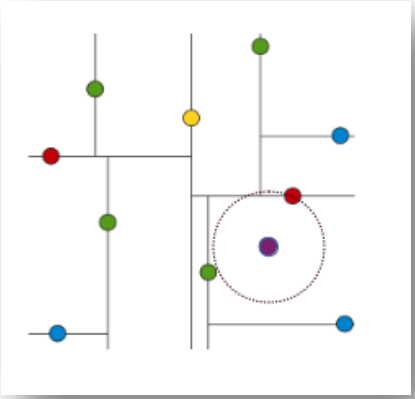


Figura 15 – Selección del nodo actual (color rojo) como nodo más cercano

2.4 RANSAC

Llegados a este punto, ya se han obtenido las correspondencias entre dos nubes de puntos, sólo faltaría estimar las matrices de rotación y translación. Sin embargo, hay que eliminar las correspondencias no deseadas, producidas por ruido o confusión. Así pues, para ello empleamos el método RANSAC.

Random sample consensus (RANSAC) es un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contienen valores atípicos. Es un algoritmo no determinista en el sentido de que produce un resultado razonable sólo con una cierta probabilidad. Los datos consisten en los llamados “inliers”. Los inliers son datos cuya distribución se explica por un conjunto de parámetros del modelo, aunque pueden estar sujetos a datos que no encajan en el modelo como pueden ser el ruido y los valores atípicos.

Así pues, RANSAC logra su objetivo de la siguiente manera.

1.- Selecciona un subconjunto aleatorio de los datos originales. Este subconjunto pasa a llamarse “inliers hipotéticos”.

2.- Se monta un modelo basándose en el conjunto de inliers hipotéticos.

3.- Todos los demás datos se prueban contra el modelo ajustado. Se consideran como parte del conjunto del consenso aquellos puntos que se ajustan al modelo estimado, de acuerdo con alguna función de pérdida de modelos específicos.

4.- El modelo estimado es razonablemente bueno si se han clasificado suficientes puntos como parte del conjunto de consenso.

5.- Finalmente, el modelo puede mejorarse si se vuelve a estimar todos los miembros del

conjunto de consenso.

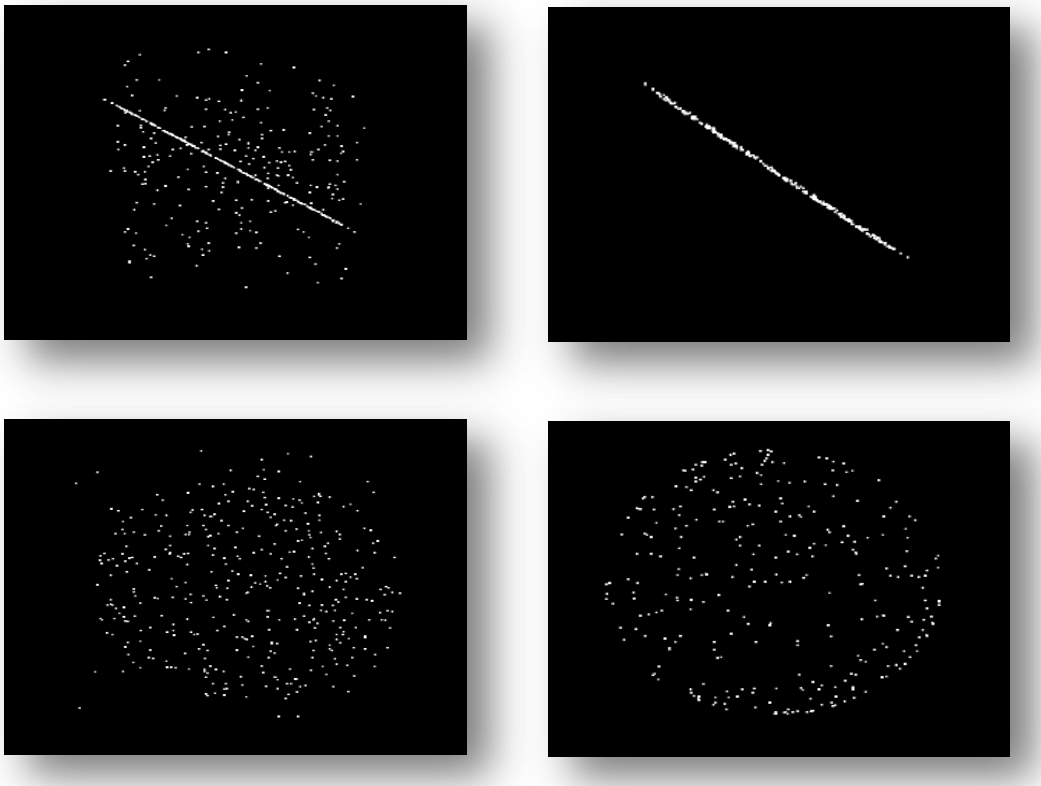


Figura 16 – A la izquierda se visualizan nubes de puntos sin aplicar el método RANSAC. A la derecha las mismas nubes de puntos aplicando el método RANSAC

También tenemos que tener en cuenta que RANSAC no funciona perfectamente. Para comenzar, no hay un tiempo máximo para calcular los parámetros. De esta forma, puede que cuando el número de iteraciones haya finalizado el resultado puede que no sea el óptimo y puede incluso que no se ajuste a los datos. Además, RANSAC no siempre es capaz de encontrar la configuración óptima aun analizando los modelos moderadamente contaminados. Por lo general, tiene un rendimiento pobre cuando el número de inliers es inferior al 50%.

2.5 Hough 3D Grouping

Éste último apartado consiste en la obtención de las matrices de rotación y traslación para una nube de puntos con el objetivo de desplazarla con respecto a la nube con la cual se va a emparejar y así formar una única nube.

Estas matrices se obtienen introduciendo como datos las correspondencias estimadas como óptimas después de aplicar RANSAC, los kepoints de las nubes de puntos y los Local Reference Frame.

Este método es mucho más complejo que los anteriores. Para la clase empleada se utiliza una técnica conocida como la transformada de Hough. Esta técnica fue concebida para realizar la detección de figuras en imágenes en 2D, aunque más tarde se amplió para trabajar con formas arbitrarias o dimensiones superiores. El método funciona con un procedimiento de votación, con los votos siendo arrojados a los candidatos que se encuentran para ser el más adecuado. Si hay suficientes votos para una determinada posición en el espacio, entonces se considera que la forma se encuentra ahí y se recuperan sus parámetros.

El procedimiento de voto se lleva a cabo en un espacio de parámetros, lo que se tendrían 6 dimensiones en el caso en el que sea necesario una pose completa a estimar (rotación más translación). Obviamente, esto supondría un coste computacional muy elevado, de forma que el método implementado en PCL sólo utiliza un espacio de 3D y emplea los LRF (Local Reference Frames) para tener en cuenta las tres dimensiones restantes.

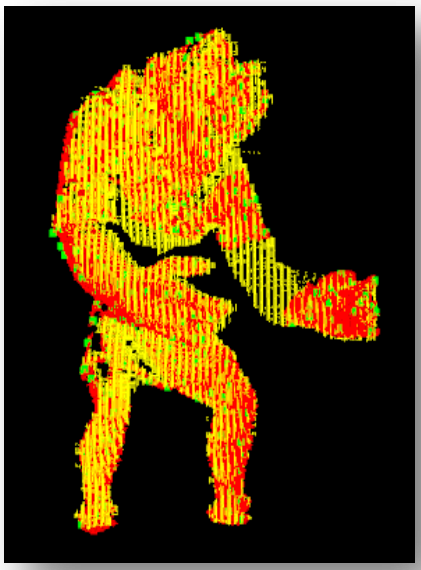


Figura 17 – Resultado final al realizar el emparejamiento entre dos nubes de puntos

En la *Figura 17* se puede visualizar la unión entre dos nubes de puntos una vez se le ha aplicado a una de ellas la matriz de rotación y de translación. El color amarillo corresponde con una nube de puntos (a la que se le ha aplicado la matriz de rotación y translación), y el color rojo a otra.

Capítulo 3. Resultados

Los resultados que se obtienen de la aplicación del desarrollo del proyecto (capítulo 2) son los emparejamientos entre las nubes de puntos, que han sido estimadas de las vistas originales. Dichos resultados han sido los siguientes:

Figura 18 – Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 19 - Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 20 – Resultado de la unión de las nubes de puntos de la *Figura 18*

Figura 21 - Resultado de la unión de las nubes de puntos de la *Figura 19*

Figura 22 - Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 23 - Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 24 - Resultado de la unión de las nubes de puntos de la *Figura 22*

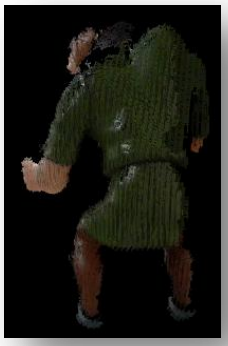


Figura 25 - Resultado de la unión de las nubes de puntos de la *Figura 23*

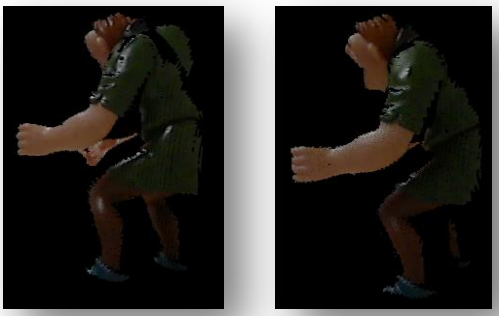


Figura 26 - Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 27 - Nubes de puntos obtenidas a partir de imágenes en 2D

Figura 28 - Resultado de la unión de las nubes de puntos de la *Figura 26*

Figura 29 - Resultado de la unión de las nubes de puntos de la *Figura 27*

Dichos resultados han sido el fruto de aplicar el proceso a distintas nubes de puntos obtenidas de imágenes RGB (nubes de puntos originales). Sin embargo, si se quiere obtener el objeto en 3D, juntando todas las vistas anteriores, es necesario realizar el mismo procedimiento pero aplicándolo de distinta manera. Por ello, a continuación se detallan tres procedimientos utilizados en el presente proyecto.

3.1 Primer procedimiento

La base de este procedimiento es considerar los resultados obtenidos de los emparejamientos entre las nubes de puntos originales como punto de partida y aplicarles de nuevo el desarrollo del proyecto, repitiendo el proceso con las nubes de puntos obtenidas hasta lograr una única nube.

En la *Figura 30* podemos observar que las nubes de puntos 1, 2, 3, 4, 5, 6 y 7 corresponden a las nubes de puntos originales. Las flechas indican el emparejamiento entre estas nubes de puntos, siendo el cloud 1-2 la nube de puntos fruto de la unión entre las nubes 1 y 2, la cual es ahora el punto de partida. El procedimiento consiste en repetir los emparejamientos sucesivamente obteniendo cada vez nuevas nubes de puntos (cloud 1-3, cloud 2-4, cloud 3-5,…) hasta quedarnos con una sola nube (cloud 1-7) que representará el objeto completo en 3 dimensiones.

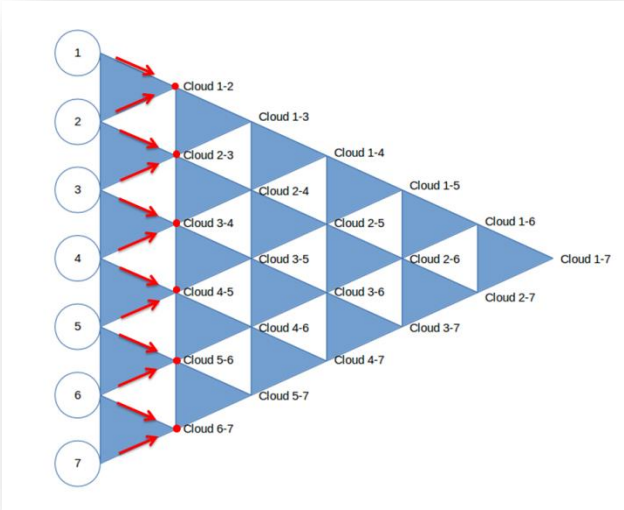


Figura 30 – Diagrama de flujo para la obtención de un objeto completo en 3D.

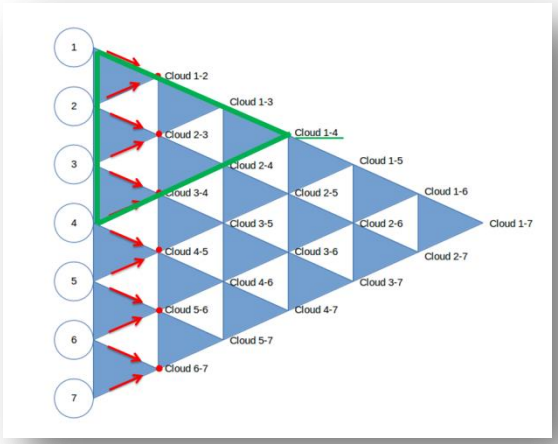


Figura 31 – Diagrama de flujo para la obtención de cloud 1-4 (triángulo verde)

El resultado obtenido de la aplicación de este procedimiento ha sido producto del emparejamiento de cuatro nubes de puntos, que correspondería a la nube cloud 1-4 en la *Figura 31*.

En el apartado de conclusiones se explicará por qué el resultado obtenido no ha sido un modelo completo.

Los resultados de aplicar este procedimiento han sido:

Figura 32 – Resultado obtenido para el primer procedimiento.

3.2 Segundo procedimiento

En este caso, como en el anterior, se parte de los resultados del emparejamiento de las nubes de puntos originales. Sin embargo, lo que ahora resulta útil son las matrices de rotación y traslación. Éstas se almacenan una vez se observa que el resultado del emparejamiento entre las nubes originales es óptimo. Por ejemplo, si observamos la *Figura 33* cuando el emparejamiento entre la nube de puntos original 1 y la nube de puntos original 2 sea óptima se obtendrá cloud 1-2 y, por tanto, se guardará la matriz de rotación-traslación 1.

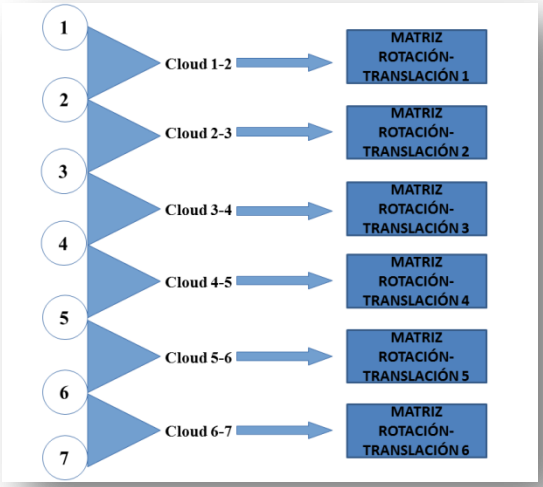


Figura 33 – Obtención de las matrices de rotación-translación una vez considerado el resultado del emparejamiento entre dos nubes de puntos como óptimo

Se ha de tener en cuenta que en este procedimiento el desarrollo del proyecto (capítulo 2) se aplica de manera iterativa, ya que es muy complicado encontrar en la primera ejecución un resultado robusto y fiable. Cabe destacar, que se ha de tener en cuenta el número de veces que se repite el proceso, hasta un máximo de 10 veces, ya que se deberán almacenar las matrices de rotación y traslación tantas veces como haya sido aplicado.

Una vez obtenidas todas las matrices, se aplicarán sucesivamente a cada nube que se vaya obteniendo.

De este modo, se comenzará por la cloud 1-2, aplicándole la matriz de rotación-traslación 2 y obteniendo así la cloud 1-2_rot, (rotada) situada en el mismo eje de coordenadas que la nube de puntos 2 original. A continuación, a esta nube de puntos rotada (cloud 1-2_rot) se une a la nube de puntos original 3 para obtener la cloud 1-3, que estará situada sobre esta misma nube original. Como se ha explicado, este proceso se repitería sucesivamente con todas las nubes de puntos (cloud 1-

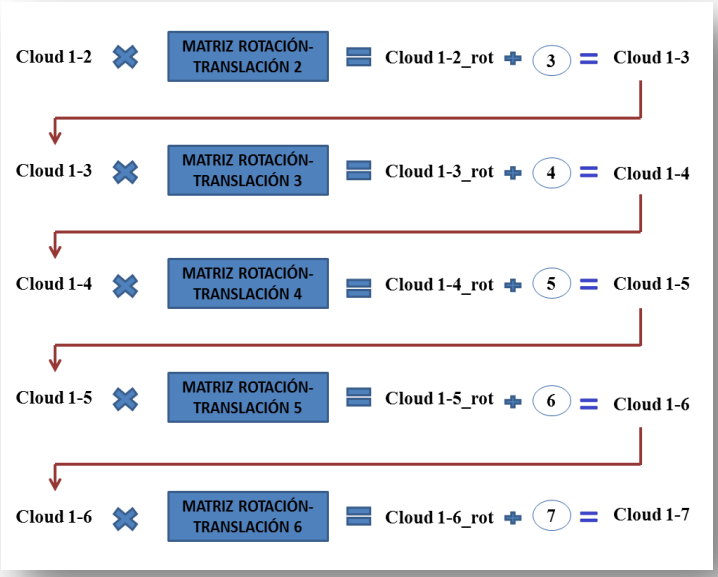


Figura 34 – Aplicación de las matrices de rotación a las distintas nubes de puntos

3, cloud 1-4, etc.) hasta lograr la nube de puntos cloud 1-7, lo que supondría tener el objeto en tres dimensiones completo.

Los resultados de aplicar este procedimiento han sido:

Figura 35 – Vistas de un objeto en 3D aplicando el segundo procedimiento

Figura 36 - Vistas de un objeto en 3D aplicando el segundo procedimiento

En la *Figura 36* se observa que en las distintas vistas del objeto hay trozos donde se refleja la luz en el momento de escanearlo.

3.3 Tercer procedimiento

Este procedimiento sigue el mismo proceso que el anterior, pero sin seleccionar todas las nubes de puntos como en el proceso de reconstrucción de la *Figura 34*. De este modo, evitamos un sobreexceso de puntos en el objeto final.

En la *Figura 37* se puede observar este procedimiento detalladamente.

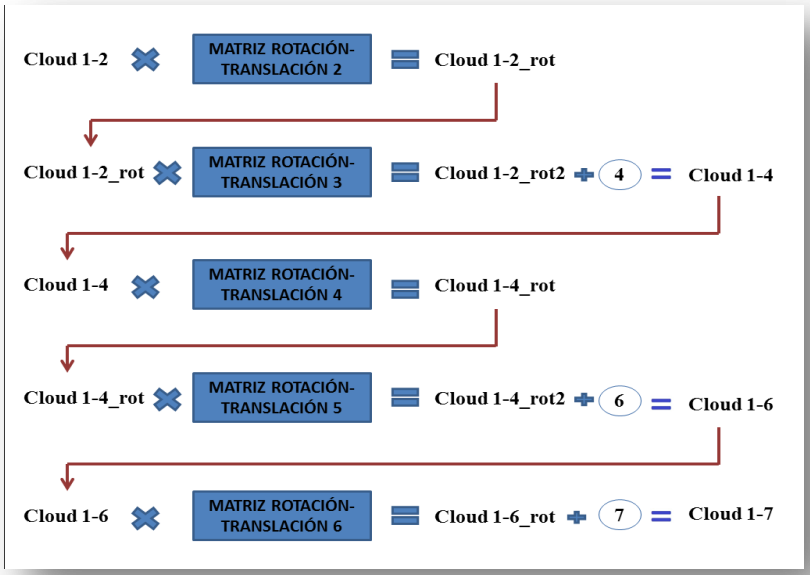


Figura 37 - Aplicación de las matrices de rotación a las distintas nubes de puntos

Los resultados de aplicar este procedimiento han sido:

Figura 38 - Vistas de un objeto en 3D aplicando el tercer procedimiento

Figura 39 - Vistas de un objeto en 3D aplicando el tercer procedimiento

Capítulo 4. Conclusiones

La obtención de resultados en este proyecto ha sido compleja. En primer lugar, debido a la falta de familiarización con imágenes en tres dimensiones. En segundo lugar, puesto que para cada emparejamiento entre dos nubes de puntos se ha de tener en cuenta los valores de los parámetros indicados en el anexo (capítulo 6). Por otro lado, debido a que la estimación de un buen resultado es subjetiva, por lo que aun obteniendo resultados parecidos entre dos nubes de puntos para un mismo emparejamiento, es muy complicado comprobar numéricamente cuál de los dos resultados es el óptimo. Por último, cabe destacar que no todos los puntos de una imagen en 2D pueden ser trasladados a un modelo en 3D, cuestión que provoca que la resolución disminuya.

A continuación se comentan los resultados obtenidos del proyecto, expuestos en el capítulo 3.

Respecto al primer procedimiento, los resultados sólo han sido obtenidos mediante la unión de 4 nubes de puntos. Esto es debido a que si nos fijamos en la *Figura 33* las nubes de puntos originales se van repitiendo con cada emparejamiento realizado, excepto las nubes de los extremos. Por tanto, para la nube de puntos cloud 1-4, las nubes de puntos originales 2 y 3 se repiten varias veces, por lo que obtenemos un sobreexceso de puntos cada vez que realizamos un nuevo emparejamiento. En consecuencia, para cada nuevo emparejamiento el cálculo computacional aumenta significativamente, provocando que la realización del emparejamiento sea tan costosa que no resulta viable computacionalmente. Por este motivo, sólo se ha realizado para un objeto.

En cuanto al segundo procedimiento, se observa cómo el resultado para la *Figura 36* es mejor que los obtenidos en la *Figura 35*, debido a que en la *Figura 36* los detalles no están tan definidos. Por ejemplo, el ojo de la *Figura 36* es mucho más grande que el de la *Figura 35*, por lo que cuando se realizan los emparejamientos los detalles pierden resolución en ambos. De esta

forma, se puede diferenciar el ojo en la *Figura 36* mientras que en la *Figura 35* resulta difícil diferenciar ya no el ojo, sino la propia cara.

Por otro lado, cabe destacar que a medida que se están realizando los emparejamientos, se comenten errores, por lo que vamos arrastrando éstos en cada ejecución del procedimiento. Como se puede observar en la *Figura 35*, el error acumulado es bastante visible tanto en los laterales de las imágenes como en la parte inferior. Mientras que en la *Figura 36* se puede observar el error en las manos y piernas de las figuras.

Como se ha explicado en el capítulo 2, la obtención de las matrices de rotación-translación se calculan teniendo como datos las correspondencias encontradas, de forma que con que una correspondencia no sea correcta, se determinará una matriz de rotación-translación que trasladará una nube de puntos erróneamente con respecto a la otra. En la siguiente figura se observa como las correspondencias estimadas entre dos nubes de puntos parecen correctas, sin embargo, el resultado es erróneo, debido a que hay varias correspondencias que no se estiman correctamente.

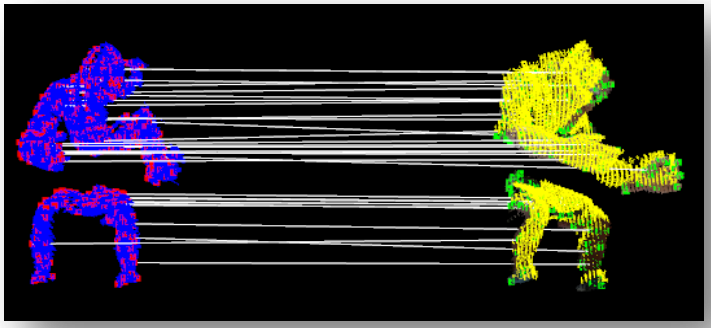


Figura 40 – Correspondencias estimadas entre 2 nubes de puntos

Figura 41 – Resultado erróneo a partir de la *Figura 40*

Por último, respecto al tercer procedimiento, los resultados son muy parecidos a los obtenidos en el procedimiento anterior. La única ventaja es que la nube de puntos disminuye en

gran cantidad su tamaño de puntos, pues no estamos añadiendo todas las nubes de puntos para la obtención del resultado final. Al realizar esto, la percepción de que el error ha disminuido puede mejorar, como se puede observar si se comparan las piernas de la *Figura 39* y la *Figura 36*. Aunque hay que tener en cuenta que el error sigue siendo el mismo.

Para finalizar este apartado, es importante conocer la dificultad de encontrar un modelo óptimo para el emparejamiento de dos superficies planas, como por ejemplo el de una caja, o para la parte plana de la *Figura 35* o *Figura 38*.

Los puntos característicos obtenidos para una nube de puntos a partir de un objeto cuya superficie es plana son muy semejantes entre sí, puesto que la mayoría de estos puntos tendrán las mismas componentes normales, al igual que las características de los Local Reference Frame y, por lo tanto, la firma de los descriptores será prácticamente igual. Así pues, las correspondencias estimadas serán completamente erróneas, como se puede visualizar en la siguiente figura.

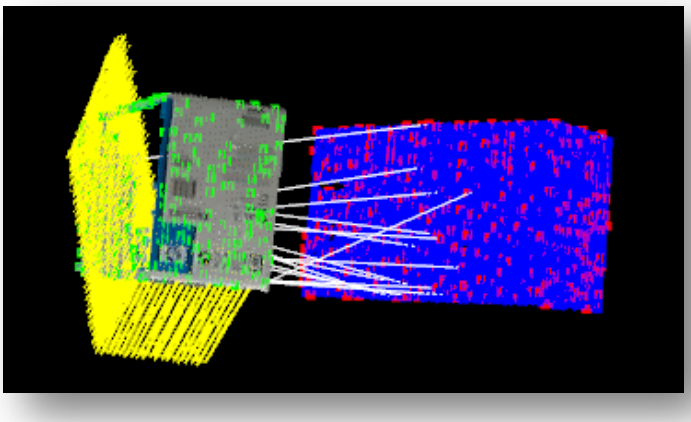


Figura 42 – Correspondencias erróneas para el emparejamiento de una caja

La nube de puntos en azul y la gris corresponden a dos nubes de puntos distintas, las cuales se quieren emparejar. La nube de puntos amarilla hace referencia a la aplicación de la matriz de rotación-translación para la nube de puntos azul, de forma que pueda unirse correctamente sobre la nube de puntos de color gris. Como se puede observar, las correspondencias son erróneas, y la nube de puntos trasladada (nube de puntos de color amarillo) no coincide sobre la nube de puntos a emparejar (nube de puntos de color gris).

Figura 43 – Resultado a partir de las correspondencias de la *Figura 42*

4.1 Trabajo futuro

En este último apartado se va a tratar de proponer posibles continuaciones y mejoras de cara al desarrollo futuro del escáner de objetos en 3D.

Para comenzar, una mejora muy importante sería la de encontrar los valores adecuados de manera automática para los parámetros comentados en el anexo (capítulo 6). Esta tarea es muy compleja, puesto que como se ha comentado anteriormente, los valores varían para los emparejamientos entre dos nubes de puntos.

Por otra parte, se podría mejorar el código para la obtención de los resultados para cualquier tipo de objeto a escanear. Por ejemplo, si se quiere escanear un objeto como una caja, cuya superficie es plana, se tendría que implementar una optimización en cuanto a la obtención de los keypoints para conseguir obtener puntos que realmente puedan considerarse característicos. Además de intentar mejorar el código, se podría elaborar el mismo procedimiento explicado en el presente trabajo, pero adaptando el procedimiento para un entorno en 2D. Obviamente, en 2D no se pueden estimar las matrices de rotación-translación, por lo que se tendrían que almacenar los keypoints, los descriptores y las correspondencias estimadas entre distintas imágenes, de forma y manera que adaptando más tarde estos parámetros a un entorno de 3D se puedan estimar dichas matrices y formar un modelo completo en 3D.

A continuación se pueden visualizar las correspondencias obtenidas entre distintas imágenes en 2D.

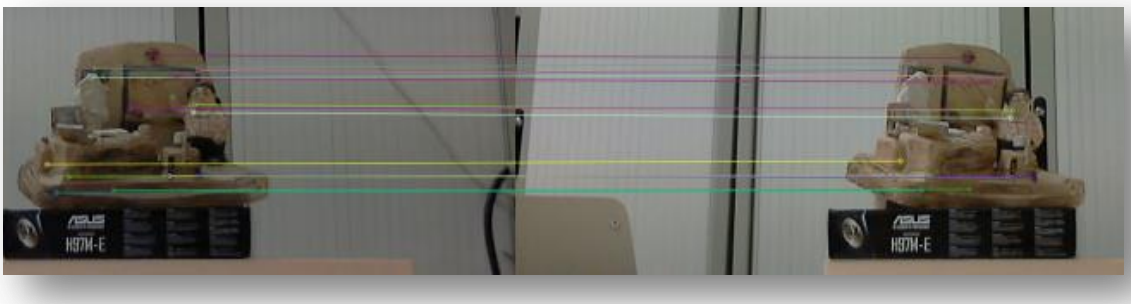


Figura 44 – Correspondencias estimadas en 2D

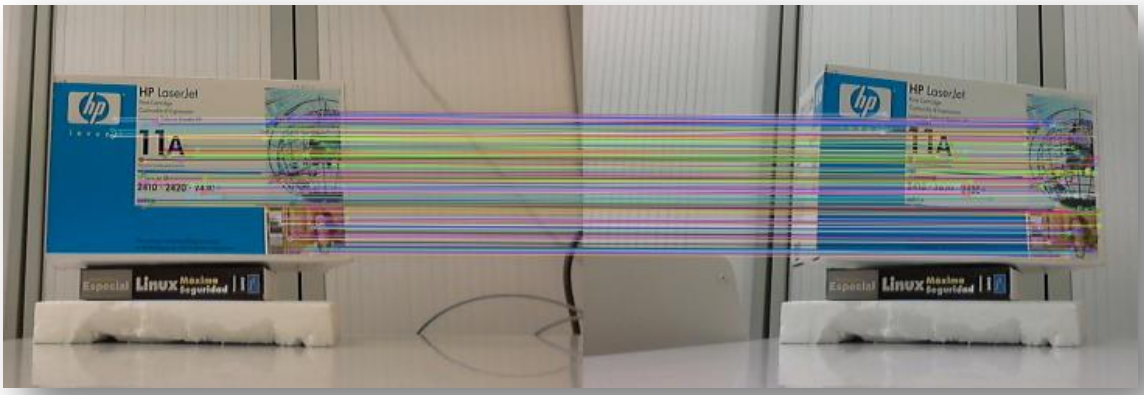


Figura 45 – Correspondencias estimadas en 2D

Como se observa en la *Figura 45*, las correspondencias estimadas son consideradas óptimas. Por tanto, si adaptamos el modelo en 2D, como se ha comentado anteriormente, a un modelo en 3D, la obtención del emparejamiento entre dos nubes de puntos cuya superficie es plana sería posible.

Para concluir este apartado, una última propuesta para trabajo a realizar en el futuro sería implementar un proceso, por ejemplo, mediante la triangulación de puntos, para originar la superficie del objeto y crear una vista más realista en 3D.

Capítulo 5. Bibliografía

[1] Arias i Duart, Anna. Realización de un escáner 3D usando un proyector y una cámara. Calibración. Julio 2015.

[2] Payá Llorens, Raúl. Realización de un escáner 3D. Medidas. Julio 2015.

[3] Hartley, R; Zisserman, A: "Multiple View Geometry in Computer Vision" Cambridge University Press. Second Edition, pp 280 – 380. 2003.

[4] Tombari, F.; Di Stefano, L. "Hough Voting for 3D Object Recognition under Occlusion and Clutter". IPSJ Transactions on Computer Vision and Applications, vol. 4, Marzo 2012.

[5] Bradski, G.; Kaebler, A. "Learning OpenCV. Computer Vision with the OpenCV Library"

O'Reilly Media, Inc. First Edition. pp 193 - 206; 265 - 300; 370 - 454 .September 2008.

[6] Falcao, G.; Hurtos, N and Massich, J. "Plane-based calibration of a projector-camera system" VIBOT master 9.1 (2008): 1-12.

[7] Lanman, D.; Taubin, G. "Build your own 3D scanner: 3D photography for begginers" SIGGARPH 2009 Course Note, Wednesday, August 5 ,2009.

[8] Moreno, D.; Taubin, G. "Simple, accurate, and robust projector-camera calibration." 3D Imaging. Modeling, Processing, Visualization and Transmission (3DIMPTV) , 2012 Second International Conference on. IEEE, 2012.

[9] Tombari, F.; Di Stefano, L. Object recognition in 3D scenes with occlusions and clutter by Hough voting. 2010, Fourth Pacific-Rim Symposium on Image and Video Technology

[10] Petrelli, A.; Di Stefano, L. "On the repeatability of the local reference frame for partial shape matching", 13th International Conference on Computer Vision (ICCV), 2011

[11] Yu Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition" Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference, pp.689-696, Sept. 27 2009-Oct. 4 2009

[12] Tombari,F.; Salti, S. and Di Stefano, L . A Combined Texture-Shape Descriptor For Enhanced 3D Feature Matching. In Proceedings of the 18th International Conference on Image Processing (ICIP), Brussels, Belgium, September 11-14, 2011.

[13] Point Cloud Library, "PCL: Uniform Sampling. Class Template Reference", http://docs.pointclouds.org/1.7.1/classpcl_1_1_uniform_sampling.html#details. [Online]

[14] Point Cloud Library, "PCL: BOARD Local Reference Frame Estimation", http://docs.pointclouds.org/1.7.1/classpcl_1_1_b_o_a_r_d_local_reference_frame_estimation.html#details. [Online]

[15] Point Cloud Library, "PCL: Normal Estimation OMP. Class Template Reference", http://docs.pointclouds.org/1.7.1/classpcl_1_1_normal_estimation_o_m_p.html#details. [Online]

[16] Point Cloud Library, "PCL: Correspondence Rejector Sample Consensus. Class Template Reference", http://docs.pointclouds.org/1.7.1/classpcl_1_1registration_1_1_correspondence_rejector_sample_consensus.html. [Online]

[17] Point Cloud Library, "PCL: Hough 3D Grouping. Class Template Reference", http://docs.pointclouds.org/trunk/classpcl_1_1_hough3_d_grouping.html. [Online]

[18] Point Cloud Library, "PCL: ISS Keypoint 3D. Class Template Reference", http://docs.pointclouds.org/trunk/classpcl_1_1_i_s_s_keypoint3_d.html. [Online]

[19] Point Cloud Library, "PCL: SHOT Estimation OMP. Class Template Reference", http://docs.pointclouds.org/trunk/classpcl_1_1_s_h_o_t_estimation_o_m_p.html. [Online]

[20] Point Cloud Library, "PCL: KdTree. Class Template Reference", http://docs.pointclouds.org/trunk/classpcl_1_1search_1_1_kd_tree.html#details. [Online]

Capítulo 6. Anexos

En el presente capítulo se van a detallar los parámetros y las clases empleadas para los apartados del capítulo 2 que son necesarios conocer si se quiere realizar un escaneo de un objeto completo en 3D.

2.1 Keypoints

Para la obtención de los keypoints se utiliza la clase UniformSampling y ISSKeypoint3D de la librería PCL.

- UniformSampling:
  - setRadiusSearch: Tamaño del radio de una cuadrícula en 3D.
- ISS:
  - setSalientRadius: Establece el radio de la zona esférica para calcular la matriz de dispersión de los puntos que se sitúen en su interior.
  - setNonMaxRadius: Ajusta el radio para la eliminación de los puntos que se encuentren fuera del radio estimado para los máximos locales.

Se recomienda no modificar los valores asignados a setSalientRadius y setNonMaxRadius, puesto que dependen del valor asignado a setRadiusSearch de UniformSampling.

2.2.1 Componentes Normales

A la hora de obtener las componentes normales se ha hecho uso de la clase NormalEstimationOMP proporcionada por PCL.

- setKSearch: Calcula la componente normal a partir de la obtención de los k vecinos más próximos.

2.2.2 Local Reference Frame

Los Local Reference Frame han sido estimados a partir de la clase BOARDLocalRefeneceFrameEstimation.

- setRadiusSearch: Tamaño del radio de una cuadrícula en 3D.

2.2.3 Descriptor SHOT

El nombre de la clase empleada para el cálculo de los descriptores ha sido SHOTEstimationOMP.

- setRadiusSearch: Tamaño del radio de una cuadrícula en 3D.

2.4 RANSAC

Llegados al punto de eliminar las correspondencias no consideradas como óptimas se ha empleado la clase CorrespondenceRejectorSampleConsensus.

- setInlierThreshold: Ajusta la máxima distancia entre los puntos emparejados. Los puntos cuyas correspondencias estimadas tengan una distancia menor que el umbral introducido se considerarán inliers.
- setMaximumIterations: Ajusta el máximo número de iteraciones a realizar.

2.5 Hough 3D Grouping

Por último, la clase empleada para obtener las matrices de rotación-translación ha sido Hough3DGrouping.

- setHoughBinSize: Ajusta el tamaño del contenedor en el espacio estimado por Hough. Este parámetro debe tener un valor similar al parámetro setRadiusSearch del apartado de Local Reference Frame.
- setHoughThreshold: Establece el número mínimo de votos en el espacio de Hough necesarios para inferir la presencia de una nube de puntos en otra. Se aconseja que el valor de éste parámetro sea 5.

Un aspecto importante a tener en cuenta es que los valores a introducir en los parámetros se consideran en milímetros.

Como se ha comentado a lo largo del documento, los valores de estos parámetros son independientes para cada emparejamiento entre dos nubes de puntos. Así pues, a continuación se va a proceder a explicar cómo variar los parámetros dependiendo del resultado obtenido para así poder modificar los parámetros de forma que se pueda agilizar el escaneo del objeto en cuestión.

Antes de realizar estos procedimientos, hay que estimar una buena obtención de los puntos característicos, puesto que si éstos no son estimados correctamente, los siguientes procedimientos no tendrán validez.

1.- Las correspondencias obtenidas antes de aplicar RANSAC son muy bajas.

Se debe modificar los parámetros pertenecientes a la obtención de los descriptores. Hay que tener mucho cuidado a la variación de estos parámetros, debido a que si el radio introducido es muy pequeño para el apartado de Local Reference Frame y para el descriptor SHOT, muchos puntos característicos que estén sobre una superficie parecida, con vecinos con características semejantes, obtendrán una firma en el descriptor prácticamente idéntica, por lo que al realizar las correspondencias se habrán realizado emparejamientos erróneos. Lo mismo sucede si aumentamos mucho el valor del radio introducido. Puesto que los puntos característicos que tengan una distancia de separación entre ellos no muy elevada, estarán estimando muchos las mismas características a su alrededor y por lo tanto sus firmas serán semejantes, así pues se obtendrán también un error a la hora de realizar los emparejamientos.

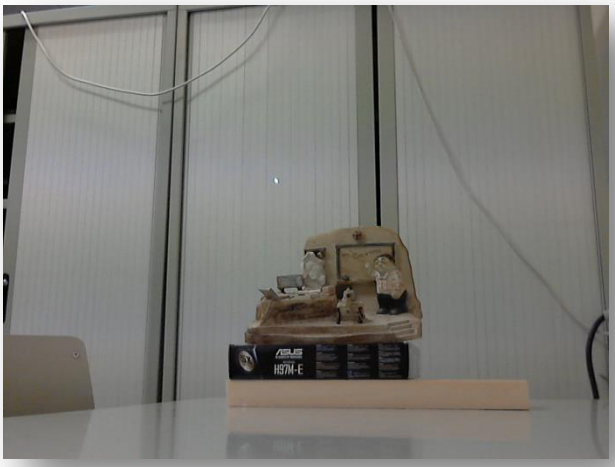
2.- Las correspondencias obtenidas al aplicar RANSAC son muy bajas.

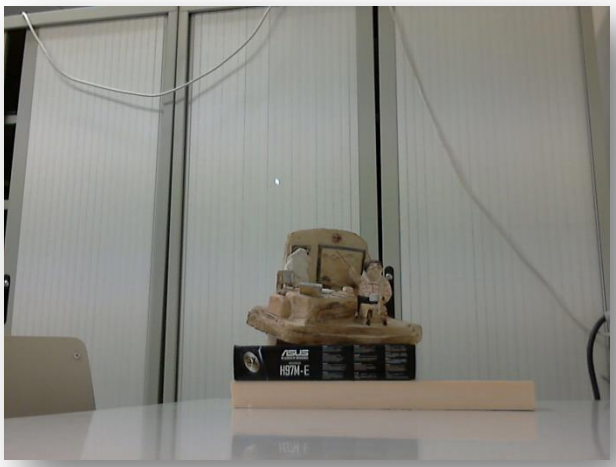
Lo primero es aumentar el valor del umbral aplicado a setInlierThreshold, ya que cuanto más pequeño es el valor del umbral, más estricto es RANSAC a la hora de descartar correspondencias, por lo que nuestro umbral aplicado puede ser demasiado estricto.

3.- Se han estimado varias matrices de rotación-translación para un emparejamiento.

El programa puede ser ejecutado las veces que se considere necesarias hasta estimar que se haya logrado el resultado esperado, sin embargo, en cada ejecución se debe de encontrar una sola matriz de rotación–translación. En el caso de que se hayan encontrado varias matrices de rotación-translación hay que reducir el setRadiusSearch perteneciente a Local Reference Frame.

Por último, se va a adjuntar las imágenes originales empleadas para la obtención de las vistas en 3D que han permitiendo obtener los resultados de las *Figuras 35, 36, 38 y 39.*





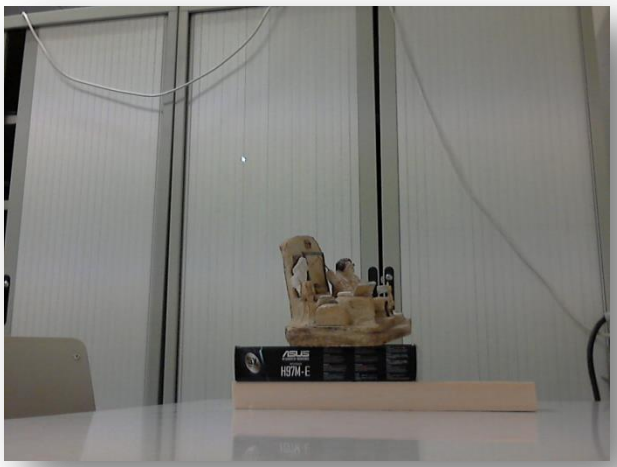



Figura 46 - Vistas de una escultura realizadas para la obtención de la misma en 3D

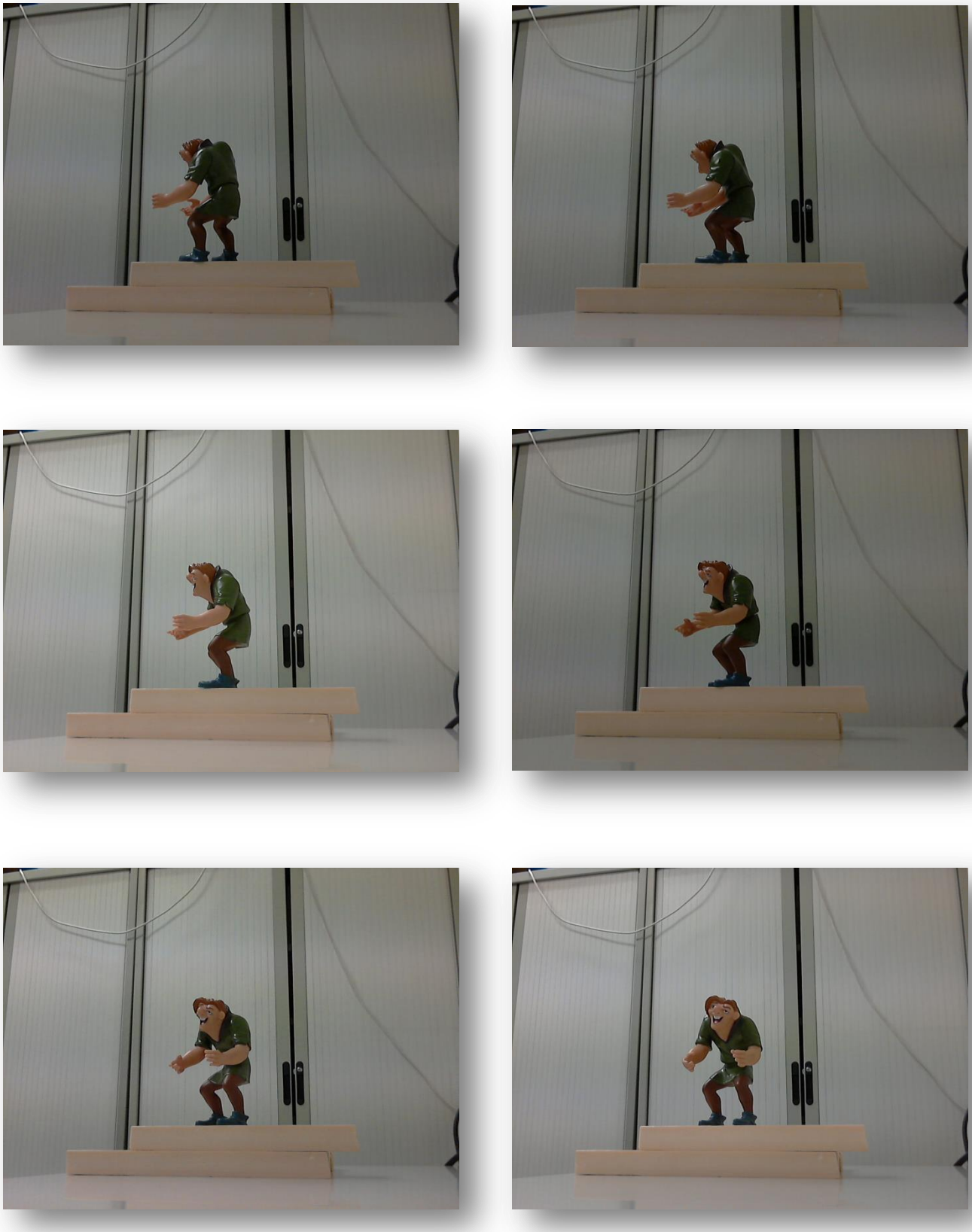


Figura 47 - Vistas de un muñeco realizadas para la obtención del mismo en 3D