Posicionamiento de objetos 3D en una escena adquirida a través de un sensor

Edgar Fradejas Rodrigañez
edgar.fradejas@e-campus.uab.cat

Resumen– La propuesta de este trabajo se centra en el desarrollo de un algoritmo para la localización de objetos en una escena, obtenida mediante un sensor 3D de adquisición de nubes de puntos. El trabajo va desde la obtención de los datos (objeto y escena), hasta la implementación algorítmica de la solución y su enfoque práctico. Se hará un estudio de las posibles tecnologías de captación 3D y la selección del sensor a utilizar, y de las diferentes posibilidades algorítmicas, de filtrado y manipulación de gran volumen de información (nubes de puntos). El algoritmo obtendrá la matriz de transformación del objeto en la escena respecto a una posición de calibración definida. Se realizará un software de entorno gráfico que se encargará de la gestión de modelos y su detección en la escena mediante la aplicación del algoritmo desarrollado. Finalmente se asociará el trabajo realizado a necesidades reales existentes y sus usos en la industria.

Palabras clave– Nubes de puntos, sensores, 3D, localización de objetos, visión por computador.

Abstract– This project focuses on the development of an algorithm for detecting the location of objects in a scene, obtained using a 3D sensor for point cloud acquisition. The project steps include obtaining the data, the algorithmic implementation, and its practical demonstration. A study of the possible 3D capturing technologies was done in order to streamline the selection of the sensor to be used, and the ideation of different algorithmic possibilities, and also understanding the filtering and processing needs of a large volumes of information. The algorithm will search for the transformation matrix of the object in the scene with respect to a defined calibration position. A graphic environment will developed capable for controlling the sensor, manage the models, and detect the object in the scene using the algorithm. Finally, the study will be associated to the existing needs in the industry and how the presented solutions can be used effectively to satisfy those needs.

Keywords– Pointclouds, sensors, 3D, pose estimation, computer vision.

1 Introducción

La visión artificial o visión por computador es la disciplina que trata de adquirir, procesar y analizar imágenes, que obtiene información para su posterior almacenamiento, estudio y procesado. Por otro lado, la visión industrial consiste en aplicar los métodos de visión por computador en la industria con el fin de obtener información, automatizar e inspeccionar líneas de producción. La industria tienes muchos procesos en los cuales tiene cabida la visión por computador, desde la clasificación por colores de piezas, hasta el análisis de defectos en piezas milimétricas. Inicialmente se centraba en problemas 2D, pero con la mejora de sensores (hardware) y de algoritmos 3D (software), se está empezando a abarcar los problemas con este tipo de tecnología. Por lo tanto, una de las tendencias actuales en la Visión Industrial consiste en la extracción de información a través de sensores, procesado y análisis de escenas 3D. Dentro de los muchos problemas que se pueden abordar con visión por computador en el ámbito 3D el trabajo se centra en sensores que permiten obtener información en forma de nubes de puntos, como por ejemplo RealSense [1], Photoneo [2], entre otros, ya que van a permitir obtener información de textura, altura y posición de la escena. Se estudiarán algoritmos que van a permitir la loca-

• E-mail de contacto: edgar.fradejas@e-campus.uab.cat
• Mención realizada: Computación
• Trabajo tutorizado por: Felipe Lumbreras (CVC) y Coen Antens (CVC)
• Curso 2017/18

lización de objetos como, por ejemplo: Fast Point Feature Histograms [3], RANSAC [4],... Con la finalidad de obtener la posición de los objetos para su posterior recogida y manipulación con un brazo robótico (Bin Picking) [5].

1.1 Motivación

La tecnología evoluciona muy rápido, constantemente están surgiendo nuevos dispositivos, nuevos algoritmos,.... Gracias a esta constante evolución están surgiendo nuevas posibilidades en cuanto a aplicaciones relacionadas con la visión por computador. Hoy en día se tienen sensores capaces de reconstruir escenas de su alrededor en forma de nubes de puntos con mucha información sobre la escena (textura, localización, objetos, ...). Información que se tiene que tratar para obtener la información que interesa y es más relevante. Para ello se precisan algoritmos que permiten tratar y procesar toda esta información. La principal motivación de este trabajo, es la investigación y programación de algoritmos capaces de trabajar todos esos datos, con el fin de localizar objetos. Una posible aplicación de estos algoritmos, es la localización de objetos en entornos industriales con el fin de que un brazo robótico conozca su localización y le permita manipularlo. Este trabajo, es una buena oportunidad de introducirse en esta realidad, ya que va a permitir aplicar los conocimientos obtenidos durante el grado a una aplicación real en la industria. El trabajo se ha propuesto conjuntamente con la empresa donde trabajo (Vision Online SL), gracias a ellos se va a disponer de hardware necesario para su realización.

1.2 Objetivos

A continuación, se listan los objetivos propuestos para el desarrollo de este TFG. Están ordenados por orden cronológico, ya que son objetivos acumulativos, es decir no se puede realizar un objetivo concreto sin haber cumplido el objetivo previo. Por lo tanto, la idea es que una vez finalizados todos los objetivos, se debería de tener una base sólida de la tecnología trabajada y un software funcional.

1. Introducir los temas de sensores y reconstrucción 3D.
   (a) Estudiar los posibles sensores a utilizar en el proyecto.
   (b) Estudiar el estado del arte en temas de localización 3D, tratamiento de nubes de puntos y modelos 3D.

   La idea es obtener la máxima información y conocimiento sobre los posibles sensores disponibles y la tecnología que se está utilizando actualmente en la industria para temas de tratado de nubes de puntos y modelos 3D. Obteniendo una buena base para poder trabajar diferentes algoritmos.

2. Obtener nubes de puntos
   (a) Estudiar la tecnología a utilizar.
   (b) Definir los objetos a localizar y las escenas.
   (c) Obtener las nubes de puntos (objetos a localizar y escenas).
   (d) Evaluar y tratar las nubes de puntos obtenidas.

   Estos objetivos son la base del proyecto, ya que todo se basará en las nubes de puntos obtenidas, por lo que habrá que obtener bastante información con la que trabajar. Inicialmente se comenzará con objetos y escenas simples, para iniciar en el tema y finalmente se buscarán objetos y escenas más complicadas.

3. Crear Algoritmo de localización de objetos
   (a) Estudiar y analizar las diferentes librerías enfocadas a visión y tratar las nubes de puntos.
   (b) Estudiar y analizar las diferentes posibilidades algorítmicas.
   (c) Implementar algoritmo de localización.
   (d) Evaluar el algoritmo de localización.

   Aquí es donde se centra la lógica del software, donde se deberá de trabajar con mucha información (nube de puntos) y algoritmos para tratar de localizar objetos en escenas. Se trabajarán algoritmos como, por ejemplo: Fast Point Feature Histograms [3], RANSAC [4]. Se utilizarán librerías como PCL [7] y OpenCV [8] para trabajar los algoritmos.

4. Crear Aplicación Windows Presentation Foundation (WPF)
   (a) Estructurar y diseñar la aplicación.
   (b) Integrar la carga de nubes de puntos.

   Por último, se realizará una primera aproximación de aplicación de escritorio en WPF [9], la cual será la herramienta que permitirá utilizar los algoritmos desarrollados. La herramienta permitirá cargar las nubes de puntos para posteriormente ejecutar la localización de objetos y ver los resultados visibles por pantalla.

1.3 Metodología

La metodología que se va a utilizar para la realización del proyecto, va a ser una metodología AGILE, que, a diferencia de la tradicional, permite trabajar el proyecto sobre iteraciones, minimizando el impacto de los posibles cambios en requisitos y objetivos. De esta forma se permite optimizar los tiempos de entrega, se mejora la calidad del software final, ya que durante las iteraciones se tienen muy presente los objetivos del proyecto y permite reorientar el proyecto en caso de introducir cambios/requisitos/nuevos objetivos. Durante el desarrollo se va a utilizar el método Kanban [10]. Generalmente se marcan los posibles estados para las tareas (normalmente: Por hacer, En proceso, Finalizada, todo y que puede haber tantos estados como queramos), y las tareas se identifican en tarjetas. Cada tarjeta se sitúa en el estado donde se encuentra su desarrollo, quedando de esta forma un panel con el estado de las tareas del proyecto. Se va a utilizar Trello [11] con cuatro paneles de estado: TO DO, DOING, TO UPGRADE, DONE.

Por otro lado, la gestión del trabajo a nivel de código y documentación se va a realizar con un software de control de versiones, Git. Permitiendo de esta forma, llevar un control del trabajo realizado a nivel de software, gestionar las versiones y organización de código. Siendo GitHub [12] el hosting donde se alojará el código y la documentación, en

un repositorio privado y GitKraken [13], el cliente con el cual se gestionará el repositorio.

Durante la realización del trabajo no se ha cambiado la metodología, se ha seguido trabajando sobre iteraciones en Kanban, utilizando el tablero de Trello para la asignación de tareas.

1.4 Planificación

El proyecto se va a dividir en varias fases. Cada fase estará dedicada al cumplimiento de un objetivo, y sus subobjetivos.

1.4.1 Fase de Introducción

En esta fase, se va a cumplir el primer objetivo: introducir los temas de sensores y reconstrucción 3D. Se va a hacer un estudio del hardware a utilizar para obtener las nubes de puntos a trabajar. También se va a hacer un estudio del estado actual en temas de localización 3D. Una vez finalizada esta fase, ya se debe tener una idea general de como enfocar el proyecto.

1.4.2 Fase de Obtención de Nubes de Puntos

En esta fase ya se tiene seleccionado el sensor con el que empezar a obtener nubes de puntos. Se hará un estudio de cómo funciona ese sensor a nivel software y hardware. Inicialmente se va a empezar a trabajar con el sensor RealSense. Este sensor permite, obtener escenas 3D gracias a su tecnología estéreo [14]. Va a permitir obtener información: profundidad de la escena, textura,... Información que posteriormente va a ser tratada para eliminar ruido, y trabajar en la parte de la escena que más interesa (donde esté localizado el objeto). Se van a trabajar algoritmos de segmentación en 3D [15], y algoritmos de clasificación [16]. Se definirán unos objetos y escenas iniciales, con los que empezar a trabajar tanto con el sensor como a nivel de software y tratamiento.

1.4.3 Fase de Localización

En esta fase es donde se hará un estudio del software y algoritmos de visión por computador. Surface Matching [17] es un tipo de algoritmo para localizar keypoints en común de un modelo a otro modelo. Este algoritmo se divide en dos partes principales:

1. Extracción de descriptores: En este proceso se buscan puntos relevantes en un modelo, puntos que lo identifiquen.

2. Matching: En este proceso se hace una búsqueda de correspondencia entre los descriptores del modelo 1 a los descriptores del modelo 2, obteniendo de esta forma con que forma se corresponden (traslación, rotación, coordenadas, ...).

Para ello, se va a utilizar la librería PCL (Point Cloud Library) [7]. Se utilizarán y evaluarán diferentes algoritmos de búsquedas de descriptores para poder utilizar el matching de los puntos del modelo, con los puntos de la escena. Se empezará utilizando Fast Point Feature Histogram [3] para calcular descriptores, y se utilizará RANSAC [4] para poder hacer el matching entre los descriptores de las escenas con los descriptores del modelo del objeto. Se implementará el algoritmo de localización de objetos en escenas 3D, y se diseñaran varias soluciones, haciendo un estudio algorítmico de ellas, para determinar cuál es la que funciona mejor para este proyecto.

2 Estado del Arte

En esta fase se ha iniciado el estudio de las diferentes técnicas que se utilizan actualmente para el escaneo 3D y el estudio de los sensores disponibles para la realización del trabajo.

2.1 Tecnologías de Obtención de Nubes de Puntos

Existen diferentes tecnologías o métodos para la obtención de nubes de puntos. Cada tecnología tiene sus ventajas y desventajas, por lo que es importante conocer las diferentes técnicas que se están utilizando para saber como enfocar las soluciones. [20]

1. Triangulación Laser (Laser triangulation)

   La triangulación láser se basa en la trayectoria del láser al impactar sobre el objeto. Como se puede apreciar en la Fig.1, se mantiene la cámara y el láser fijos, y se mueve el objeto, que al ser impactado por el láser la trayectoria de la línea del láser se ve modificada. Por lo tanto, a partir de la triangulación trigonométrica, con la distancia entre láser-sensor y sensor-objeto, se puede obtener el ángulo de proyección relacionado con la distancia entre el objeto y el sensor.

   Como ventajas, es una técnica que puede dar mucha resolución y precisión en la nube, sin embargo, al basarse en un láser, los objetos transparentes o muy brillantes se hacen difíciles de escanear.

2. Luz Estructurada (Structured Light)

   La luz estructurada utiliza la misma idea de triangulación trigonométrica, con la diferencia de que el objeto debe de estar estático y en vez de utilizarse una proyección láser de una línea, se hace una proyección de patrones (generalmente binarios).

   Como ventaja, la luz estructurada es una técnica rápida y que permite obtener bastante resolución. Como inconveniente, al utilizar proyección de patrones es muy sensible a la luz, por lo que no se pueden obtener buenos resultados en exteriores ni en objetos transparentes.

3. Tiempo de Vuelo (Time of flight)

   Los escáner basados en pulso láser, más conocido por tiempo de vuelo, se basan en el tiempo que tarda una emisión de infrarrojo en ir y volver a un objeto. Conociendo la velocidad constante de la luz ($3 \times 10^8 m/s$), se puede obtener la distancia exacta de un objeto calculando el tiempo que tarda el infrarrojo desde que es emitido hasta que vuelve.

   La gran ventaja del tiempo de vuelo, es que permite escanear entornos y objetos de gran tamaño, sin embargo, es de ejecución más lenta.

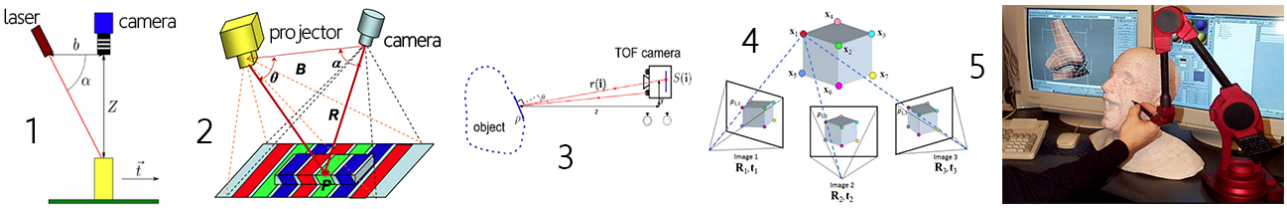

Fig. 1: Técnicas de reconstrucción 3D.
(1) Triangulación Laser [22]. (2) Luz Estructurada [23]. (3) Tiempo de Vuelo [24]. (4) Structure from motion [25]. (5) Escaneo por contacto [26].

4. Structure from motion (SfM)

   Structure from motion, se basa en la obtención de 3D a partir de imágenes del mismo objeto desde diferentes puntos de vista. Una vez obtenidas las imágenes, se procesan con algoritmos de visión por computador y métodos geométricos (extracción de características invariantes a escala y rotación, por ejemplo, SIFT).

   La ventaja principal de SfM, es la rápida adquisición y la precisión que proporciona, sin embargo, como inconveniente se puede destacar el tiempo de cálculo de los algoritmos empleados.

5. Escaneo por contacto (Contact based 3D scanning)

   Este método se diferencia de los demás, debido a que se basa en el contacto. El objeto a escanear debe de estar fijo, y manualmente se van tocando los puntos sobre el objeto. Se suele utilizar algún brazo robótico, para obtener más precisión. [21]

   La mayor ventaja es que este método permite escanear objetos transparentes, pero es un proceso lento y manual, difícil de automatizar.

2.2 Sensores

Durante el desarrollo del trabajo se ha tenido a acceso a diferentes sensores que utilizan alguna tecnología de las mencionadas en el punto anterior, con el fin de obtener nubes de puntos de escenas y objetos.

1. Cámara 3D Intel RealSense R200
   Cámara desarrollada por Intel, formada por un proyector de láser infrarrojo y tres lentes: RGB, infrarrojos izquierda e infrarrojos derecha. [1]

2. Kinect
   Dispositivo desarrollado por Microsoft, para interactuar con videoconsolas utilizando gestos y comandos de voz. Actualmente cuenta con dos versiones, Kinect v1 y Kinect v2. La primera versión utiliza la tecnología de método de luz estructurada, la segunda versión se basa en la tecnología de tiempo de vuelo (ToF).

3. Photoneo PhoXi
   Escáner fabricado por Photoneo. Utiliza la tecnología de luz estructurada. Proyecta patrones láser sobre el objeto para codificar cada punto de la escena de forma única. [2] Photoneo dispone de varias versiones del sensor dependiendo del tamaño (XS, S, M, L, XL).

3 Desarrollo

3.1 Obtención de Nubes de Puntos

Inicialmente se ha empezado a trabajar con el sensor RealSense R200, sobre una escena con tres objetos preestablecidos. Los objetos de la escena son tres objetos de gran volumen para una primera aproximación de algoritmos. Los objetos son: un destornillador, una bola de plástico y un rollo de cinta. Para trabajar con el sensor se ha utilizado el SDK de RealSense proporcionado por Intel, en un proyecto de Visual Studio 2015 que permite trabajar con el sensor, obteniendo imágenes RGB y de mapa de profundidad. También permite hacer una percepción de la escena, obteniendo así las nubes de puntos.

Fig. 2: Setup de obtención de nube de puntos con Photoneo PhoXi L, junto a robot UR.

Finalmente se ha decidido utilizar el sensor Photoneo PhoXi L a máxima resolución (3.2 millones de puntos), que nos ofrece más información de la escena, y con menos ruido. Se ha estado trabajando con tazas y con el tapón de un bolígrafo gigante.

3.2 Procesado

Para el procesado de las nubes de puntos se ha empezado a trabajar con una nube de una mesa, que proporciona como ejemplo la web de PCL [7]. Posteriormente una vez hecha la programación, se ha pasado a probar con las nubes de puntos obtenidas con los sensores. De esta forma, se trabaja sobre una escena ideal, densa y con mucha información para luego pasar a la realidad: nubes de menos calidad. Se ha trabajado con la librería Point Cloud Library (PCL) [7], instalada y configurada sobre Visual Studio 2015.

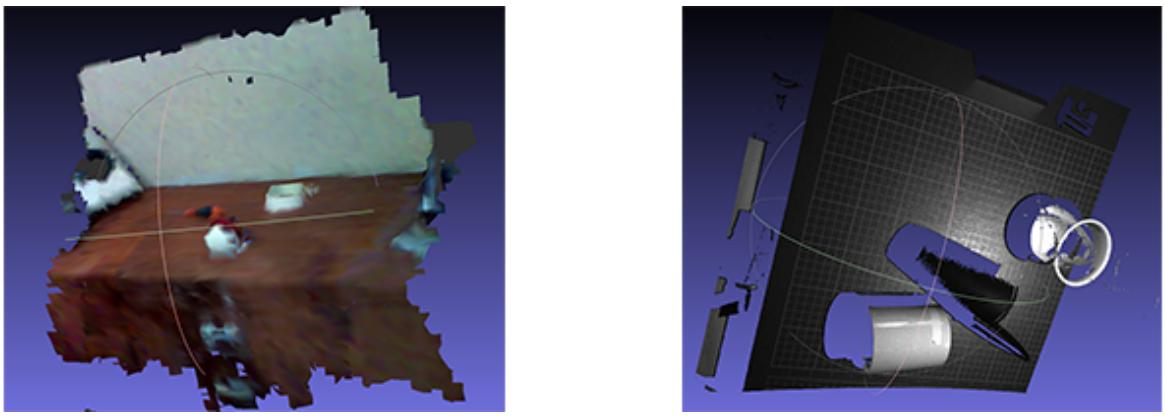

Fig. 3: (Izquierda) Escena con textura capturada con Realsense. (Derecha) Escena con textura capturada con Photoneo PhoXi L.

3.2.1 Downsampling

Con esta técnica, se ha reducido la densidad de la nube de puntos. Generalmente una buena nube de puntos, tiene mucha información, es decir, millones de puntos. Las nubes con tanta densidad son difíciles de tratar, ya que el tiempo de cómputo para los algoritmos sobre la nube entera se dispara tanto en tiempo como en memoria. Para solventar el problema, se utiliza la técnica de Downsampling, que consiste en aplicar filtros a la nube para quedarnos con los puntos más representativos.
Se ha utilizado un filtro VoxelGrid. Este filtro, se basa en una red formada por Voxels (Fig.4). La nube se cubre de la estructura, y por cada voxel se calcula el centroide utilizando el concepto de vecindad (neighbor), siendo el centroide el punto más representativo. El tamaño del voxel se puede definir en altura, anchura y profundidad, por lo tanto, se puede decidir si hacer un downsampling ligero o más exhaustivo.

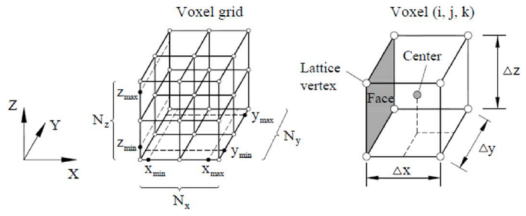

Fig. 4: Downsampling, Voxelgrid [28].

De esta forma estamos reduciendo la densidad de la nube considerablemente (dependiendo del tamaño definido por voxel), sin perder mucha información ya que nos quedamos con la información más representativa.

3.2.2 Eliminar outliers

Los sensores y las técnicas de obtención de nubes de puntos no son perfectas, siempre existe ruido, información errónea que no da información real sobre la escena. Este ruido es recomendable eliminarlo, ya que influye en los resultados de los algoritmos, al trabajar sobre información que realmente no es de la escena. Para ello se ha utilizado el filtro Statistical Gross Outlier Removal (Fig.4). Este filtro recibe como parámetros el número de vecinos definido y la desviación estándar.

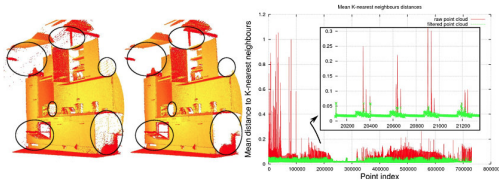

Fig. 5: Statistical Gross Outlier Remova filter [29].

El filtro calcula la media y genera una campana de Gauss, con la estadística de a cuanta distancia de un punto puede encontrarse un vecino real, por lo tanto, los puntos fuera de este rango son considerados outliers y eliminados de la escena.

3.2.3 Plane Segmentation

Todas las nubes de puntos que se han obtenido tienen una superficie plana donde se sitúan los objetos (generalmente una mesa). Es por eso, que, para reducir la densidad de las nubes de puntos y segmentar mejor los objetos, lo ideal es quitar esa superficie. Para ello se ha realizado una segmentación del plano.
Los planos vienen definidos por la ecuación del plano:

$$Ax + By + Cj + D = 0 \quad (1)$$

Encontrando la ecuación del plano que define la mesa, se puede segmentar todos los puntos que queden por debajo, eliminando así el posible ruido y la superficie donde se encuentran los objetos.

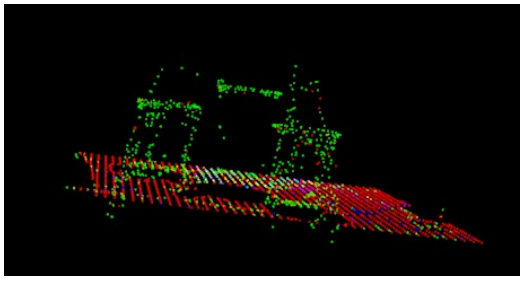

Fig. 6: Plane Segmentation, plano (color rojo), objetos (color verde).

Para encontrar el plano se ha utilizado RANSAC [5] con diferentes iteraciones y umbral de distancia, dependiendo

de la nube de puntos. Se ha establecido un porcentaje máximo a filtrar, para evitar encontrar algún objeto que tenga un plano bien definido. RANSAC permite encontrar un modelo matemático (en este caso la ecuación del plano), y nos devuelve los coeficientes, con los cuales se van a encontrar los puntos que corresponden al plano.

3.2.4 Filtrado en las dimensiones

Una vez se ha obtenido el plano, se ha hecho un filtrado a lo largo de las tres dimensiones. Como se sabe el plano, se puede afirmar que todo lo que queda a la altura y por debajo, no es interesante para la localización.
Se ha realizado un filtrado obteniendo la información que queda por encima del plano, reduciendo así la densidad de la nube, ya que la gran parte de la información estaba en la superficie donde se encuentran los objetos. Se puede hacer un filtrado a lo largo de las tres dimensiones X, Y, Z, pero hay que tener en cuenta el estado de la nube de puntos, se puede dar el caso en que nuestra nube no esté corregida y debido al ángulo del sensor respecto a la proyección, las coordenadas no estén referenciadas al origen del sistema de coordenadas del sensor, por lo que se puede tener una cierta inclinación.

3.2.5 Corrección del Ángulo del Sensor

El sensor Photoneo PhoXi L trabaja con la tecnología de luz estructurada, en la que se proyectan patrones sobre la escena. El sensor tiene separado el proyector y la cámara, lo que genera una cierta inclinación en la nube debido al ángulo de proyección. Se ha corregido esta inclinación utilizando la matriz de corrección del sensor sobre las nubes a tratar ya que al corregir la inclinación la segmentación del plano ha mejorado mucho y ha permitido aplicar el filtrado en las dimensiones.

3.3 3D pose estimation

Este punto es el más crítico del trabajo. Hasta ahora se ha trabajado para filtrar la nube de puntos lo máximo posible para intentar dejar únicamente los objetos en la escena. En esta parte, se ha empezado con la localización de los objetos en la escena. Definimos localización como encontrar un modelo de un objeto en una escena, obteniendo así la matriz de transformada que nos indica la rotación, la traslación y la escala a la que se encuentra el objeto en la escena respecto al modelo.
El algoritmo trabaja con dos nubes de puntos: la escena y el modelo del objeto a buscar en la escena. A la nube de puntos de la escena, se le aplican todas las técnicas mencionadas anteriormente para intentar reducir al máximo el espacio de posibilidades donde buscar el modelo. Al modelo se le reduce la densidad de la nube con las mismas proporcionas que se ha reducido la escena. Tanto a la escena como al modelo, se les aplica un algoritmo para calcular las normales. Las normales son un vector perpendicular al punto que nos indican la orientación de este.
En las nubes ya, con las normales calculadas, se inicia la búsqueda de keypoints utilizando Fast Point Feature Histogram [4]. Los keypoints son puntos muy representativos en la escena. Únicamente tienen asociada la posición donde se encuentran por lo que necesitamos de descriptores para almacenar más información. Un descriptor, es un vector de características de un keypoint, es decir, da información del punto. Son invariantes, por mucho que la nube rote, se traslade o se reduzca la escala, esos descriptores seguirán estando. Por lo tanto, si se encuentran keypoints en el modelo y en la escena, se puede intentar hacer una correspondencia entre ellos utilizando descriptores.

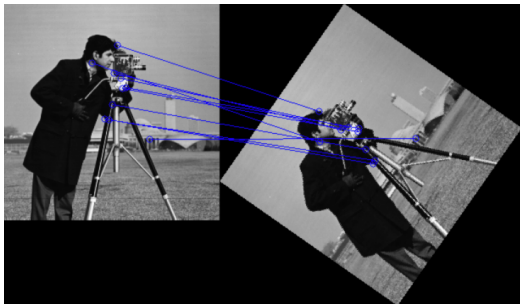

Fig. 7: Ejemplo de keypoints en una imagen [31].

Con los keypoints localizados, se aplica el algoritmo RANSAC para intentar hacer la correspondencia entre los descriptores del modelo en la escena. Dependiendo de los parámetros del algoritmo, como por ejemplo número de iteraciones, distancia aceptada y densidad de la nube, este proceso puede ser lento. Se aplica hasta encontrar una correspondencia entre keypoints, o finalmente hasta que se termine el algoritmo y no haya podido converger, es decir, no se ha podido lograr la correspondencia.

3.4 Resultados

3.4.1 Primeros Resultados

Inicialmente se ha empezado a trabajar con una nube de ejemplo proporcionada por la web de PCL [32]. A la nube se le ha aplicado la eliminación de outliers a la nube sin tratar, y a la misma nube tratada con reducción de densidad (Fig.9). La nube inicial es muy densa, tiene 460400 puntos, por lo que ha sido conveniente reducir la densidad, se ha utilizado inicialmente 50 vecinos y una desviación estándar de 1.

Se ha hecho la comparación de tiempo de ejecución eliminando outliers sobre la nube de ejemplo. Se comparan tiempos entre la nube sin tratar, y la nube tratada reduciendo la densidad.

TABLA 1: ELIMINANDO OUTLIERS SOBRE LA NUBE DE EJEMPLO.

	Puntos	Vecinos	σ	Tiempo (s)
Inicial	460400	50	1	153,428
Reducida	41049	50	1	14,077

TABLA 2: ELIMINANDO OUTLIERS SOBRE LA NUBE DE REALSENSE.

	Puntos	Vecinos	σ	Tiempo (s)
Inicial	6782	50	1	2,118
Reducida	5363	50	1	1,577

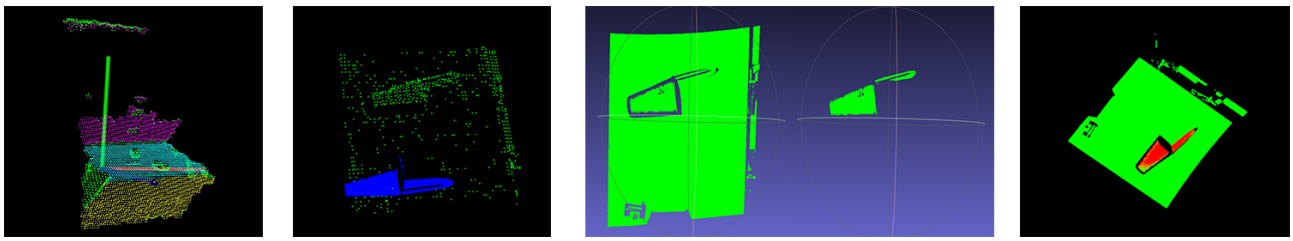

Fig. 8: (Izquierda) Segmentación de planos en la nube de puntos de RealSense. (Centro-Izquierda) Localización fallida en nube de Photoneo, escena tratada (color verde), modelo (color azul). (Centro-Derecha) Segmentación del plano añadiendo una tolerancia de 15 mm. En la parte izquierda de la imagen, escena original, en la parte derecha escena después de la segmentación del plano. (Derecha) Localización del modelo sobre la escena segmentada y visualizado sobre la escena original. En verde la escena original y en rojo la localización. (Ver en versión digital para apreciar bien las diferencias).

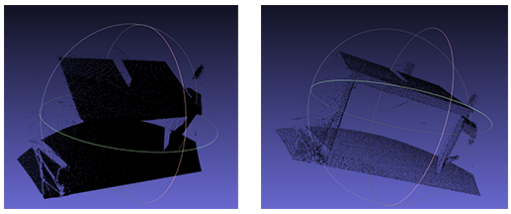

Fig. 9: (Izquierda) Nube de puntos inicial. (Derecha) Nube de puntos al aplicar Downsampling.

Se puede observar como en la nube de ejemplo, se reduce el tiempo un 90% y en la nube adquirida por el sensor RealSense (Fig. 3), un 72%.

3.4.2 Resultados Avanzados

Se ha continuado trabajando con la nube de puntos inicial obtenida con RealSense (Fig. 3). Como se puede apreciar, en este caso hay tres planos bien definidos, quedando de color verde la nube segmentada (Fig. 8). Una vez segmentada la nube se ha procedido a obtener el modelo de un objeto para su localización, pero con RealSense ha sido muy difícil ya que la calidad de las nubes de puntos es muy mala, con mucho ruido por lo que al intentar hacer la localización no se han obtenido resultados. Por lo tanto, se han empezado a hacer pruebas con el sensor Photoneo PhoXi L. Con Photoneo se han capturado varias escenas, donde se encuentran tazas y el tapón de un bolígrafo gigante. Las escenas son de gran resolución y tienen 3.200.000 puntos, por lo que para trabajar con ellas se ha tenido que hacer varios filtros, y aun así los algoritmos se demoran. Photoneo no permite la reconstrucción 3D de un objeto, por lo que se ha tenido que capturar el objeto en una escena y mediante el programa Meshlab [30] se ha segmentado de la escena. Para trabajar con las nubes se ha eliminado la textura, se les ha aplicado downsampling, segmentación del plano (ejemplo Fig. 6) (ya que la gran mayoría de puntos están en la zona de la mesa) y eliminación de outliers.

Una vez configurados los parámetros del algoritmo de localización que se han demostrado idóneos para estas nubes a base de pruebas, se han empezado a localizar objetos de forma errónea, como se puede apreciar en la Fig. 8. Una vez eliminados los outliers, hecho un downsampling y segmentado el plano, observamos cómo no se ha segmentado del todo la mesa, entonces el algoritmo detecta alguna zona de la mesa como la parte plana del tapón de bolígrafo y hace la correspondencia. Se ha detectado una inclinación en la nube lo que hace que la segmentación no funcione correctamente. En el sensor hay una distancia física entre la cámara y el proyector de patrones, esta separación produce una cierta inclinación en la nube resultante, para solucionarla se ha aplicado la corrección del ángulo a la escena mediante la matriz de corrección de Photoneo PhoXi L (Fig. 10).

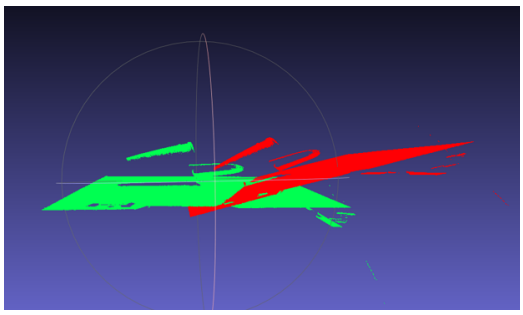

Fig. 10: Corrección del ángulo de la escena. Escena inicial (color rojo), escena corregida (color verde).

Se ha mejorado la segmentación del plano, para permitir incluir una tolerancia, ya que nunca se va a encontrar un plano perfectamente definido. Se puede ver como al añadir una tolerancia de 15mm al algoritmo de segmentación conseguimos eliminar el plano en su totalidad (Fig. 8). Ejecutando el algoritmo de localización sobre la nueva escena, se ha podido localizar el objeto correctamente (Fig. 8). Se ha hecho la correspondencia de 3006 puntos entre la escena y el objeto. Como el objeto después del tratado tiene 3006 puntos, se puede decir que se ha localizado el objeto perfectamente (con una pequeña tolerancia).

TABLA 3: LOCALIZACIÓN DEL TAPÓN (PUNTOS)

Escena	Modelo	Sampling	Inliers	Tiempo (s)
3107	3006	2x2x2	3006/3006	87,179

TABLA 4: LOCALIZACIÓN DE UNA TAZA (PUNTOS)

Escena	Modelo	Sampling	Inliers	Tiempo (s)
9139	3272	2x2x2	3272/3272	22,048

Se ha realizado el mismo proceso para más escenas. En la Fig. 11 se puede ver todo el proceso de localización de una taza en la escena.

Como se puede observar en la Fig. 12, se ha realizado un estudio para evaluar la importancia de la densidad de

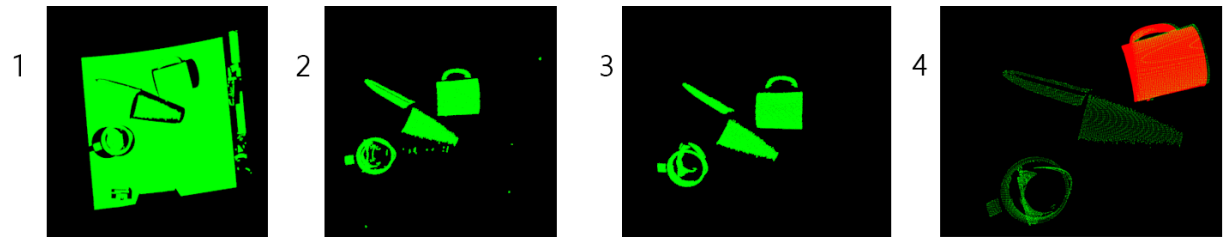

Fig. 11: Localización de una taza en una escena. De izquierda a derecha se puede visualizar todo el proceso de localización. (1) Inicialmente se carga la nube de la escena y se le corrige la inclinación provocada por el sensor. Se procede a segmentar el plano (2) y se eliminan outliers (3). Paralelamente se realiza el mismo proceso (pero sin segmentación del plano) para el modelo, en este caso la taza. Finalmente se ejecuta el algoritmo de matching entre los descriptores del objeto y de la escena (4). En color verde la escena tratada y de color rojo la localización del modelo. (Ver en versión digital para apreciar bien las diferencias).

la nube en el tiempo que demora la localización. Se puede observar que a mayor downsampling (menor densidad), menor es el tiempo que se demora la localización. Por lo tanto, interesa reducir la nube al máximo donde la mayoría de puntos sean keypoints.

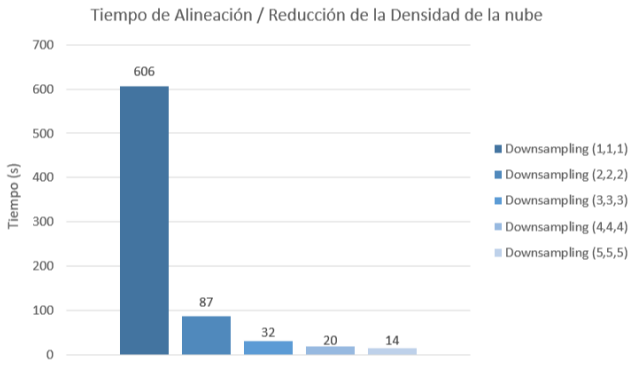

Fig. 12: Análisis de tiempo del algoritmo de localización con diferentes tipos de downsampling.

3.5 Aplicación Windows Presentation Foundation

Se ha desarrollado una sencilla interfaz gráfica utilizando WPF para visualizar los resultados obtenidos. Inicialmente se ha hecho un Mockup con un aproximación del diseño deseado (Fig. 23). Se ha programado en C# utilizando el framework .NET. El estado actual de la aplicación permite cargar el path de dos nubes de puntos, nube de la escena inicial y la nube del modelo. Finalmente, la llamada al programa de localización pasándole, le las dos nubes para iniciar el algoritmo (Fig. 13).

4 Conclusiones

En esta fase del proyecto, se pueden sacar dos tipos de conclusiones sobre el trabajo realizado.

4.1 Sensores y Nubes de Puntos

El tipo de sensor influye mucho en la calidad de la nube de puntos. Cada sensor utiliza una tecnología diferente, por lo que hay que hacer un estudio de qué sensor es más adecuado en cada situación. En este caso se ha visto que RealSense es un sensor más enfocado al ocio y no se puede comparar con Photoneo PhoXi L a nivel de calidad de nube de puntos.

4.2 Algoritmos

En general, los algoritmos de correspondencia suelen ser lentos, no se pueden aplicar a un gran volumen de información porque si no se demoran mucho tiempo en procesar (más de 10 minutos). Es necesario utilizar el conocimiento de la escena (segmentación de planos, downsampling, filtrado, ...) para aprovechar al máximo estos algoritmos. Por lo tanto, hay dos opciones que deben de aplicarse para optimizar al máximo el tiempo de ejecución de los algoritmos de correspondencia:

1. Software: Aplicar el conocimiento de la escena para segmentar únicamente la información que interesa y ejecutar el algoritmo sobre esa información reducida.

2. Hardware: Es recomendable utilizar GPUs para el procesado, ya que para nubes densas el algoritmo se demora mucho en CPU e incluso, en casos extremos, se obtienen excepciones por falta de memoria.

A pesar de que se han podido localizar varios objetos, se debe de mejorar la localización por varios motivos:

1. Tiempo: La localización se demora demasiado tiempo para una aplicación. No debería de tardar más de 5 segundos.

2. Generalización de los parámetros: Se debe de intentar generalizar los parámetros del algoritmo, ya que cada escena/objeto tiene una configuración de parámetros que da mejores resultados.

5 Siguientes Pasos

Se han pensado posibles mejoras con el fin de mejorar la localización.

1. 2D: Dado que las nubes de puntos tienen textura, se podría intentar pasar de un espacio 3D a una representación 2D de la escena, con tal de hacer una primera localización del área de búsqueda del objeto. Una vez localizada en 2D el área donde está situado el objeto, volver a la representación en 3D y ejecutar el algoritmo unicamente sobre la zona detectada. Esto se puede

hacer siempre y cuando se mantenga la nube organizada ya que, en el caso contrario, no se podría proyectar a un espacio 2D correctamente.

2. GPU: Una mejora necesaria es el hecho de pasar la ejecución a una GPU. Se ha visto como el algoritmo se demora bastante e incluso da errores de memoria en CPU. La mejora del hardware ayudaría mucho a poder hacer más iteraciones de búsqueda sin perjudicar el tiempo del proceso.

Una vez el algoritmo funcione perfectamente y en un tiempo razonable para una aplicación (unos 5 segundos), se podría pasar a trabajar con un robot. Para ello se debería de poner en el mismo sistema de coordenadas, las coordenadas del robot y las coordenadas del sensor, ya que cuando se localice un objeto el robot debe de ir a buscarlo. Finalmente se mejorará la aplicación WPF para permitir parámetros de configuración y calibración del sensor.

6 AGRADECIMIENTOS

A mi tutor, Felipe Lumbreras y a Coen Antens (CVC) por todo el apoyo demostrado desde mi estancia de prácticas en el Centro de Visión por Computador, hasta el día de hoy y por todo el conocimiento que han compartido conmigo.

A todo el equipo de Vision Online SL, por la oportunidad, apoyo y todo el conocimiento que me han ofrecido.

A mi familia, por apoyarme y aguantarme en los días de café, mal humor y estrés.

REFERENCIAS

[1] Intel. (2018). Tecnología Intel® RealSense™. [online] Available at: https://www.intel.es/content/www/es/es/architecture-and-technology/realsense-overview.html [Accessed 28 Feb. 2018].

[2] Photoneo.com. (2018). Photoneo PhoXi L – Focused on 3D. [online] Available at: http://www.photoneo.com/product-detail/phoxi-l/ [Accessed 28 Feb. 2018].

[3] Bogdan, R., Blodow, N.,y Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D Registration. Robotics and Automation, 2009. ICRA '09. IEEE International Conference on

[4] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications ACM, 1

[5] Ingenieros. (2018). Robótica Guiada por Visión: Bin Picking, una solución para los procesos productivos. [online] Available at: http://www.ingenieros.es/noticias/ver/robotica-guiada-por-vision-bin-picking-una-solucion-para-los-procesos-productivos/5951 [Accessed 28 Feb. 2018].

[6] Mathforum.org. (2018). Transformation Matrix. [online] Available at: http://mathforum.org/mathimages/index.php/Transformation_Matrix [Accessed 4 Mar. 2018].

[7] Documentation - Point Cloud Library (PCL). [online] Available at: http://pointclouds.org/documentation/ [Accessed 21 Feb. 2018].

[8] Opencv.org. (2018). OpenCV library. [online] Available at: https://opencv.org [Accessed 21 Feb. 2018].

[9] Docs.microsoft.com. (2018). Introducción (WPF). [online] Available at: https://docs.microsoft.com/es-es/dotnet/framework/wpf/getting-started/ [Accessed 4 Mar. 2018].

[10] Kanban - El orientador ágil. [online] Available at: https://es.atlassian.com/agile/kanban [Accessed 28 Feb. 2018].

[11] Trello.com - (2018). Trello. [online] Available at: https://trello.com [Accessed 28 Feb. 2018].

[12] GitHub. (2018). Build software better, together. [online] Available at: https://github.com [Accessed 28 Mar. 2018].

[13] GitKraken.com. (2018). Git GUI for Windows, Mac & Linux — GitKraken. [online] Available at: https://www.gitkraken.com [Accessed 28 Feb. 2018].

[14] Intel® RealSense™. (2018). Stereo - Intel® RealSense™. [online] Available at: https://realsense.intel.com/stereo/ [Accessed 10 Mar. 2018].

[15] Nguyen, A., (2013). Fischler and R.C. Bolles. 3D Point Cloud Segmentation: A survey.

[16] Grilli, E., Menna, F. ,Remondino, F. (2017). A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS

[17] surface_matching. Surface Matching — OpenCV 3.0.0-dev documentation. [online] Available at: https://docs.opencv.org/3.0-beta/modules/surface_matching/doc/surface_matching.html [Accessed 21 Feb. 2018].

[18] H Bay, T Tuytelaars, L Van Gool (2006). SURF: Speeded Up Robust Features, 2006. European conference on computer visión.

[19] Opencv. (2018). Introduction to SIFT [online] Available at: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html [Accessed 10 Mar. 2018].

[20] Aniwaa. (2018). 3D scanning technologies - Aniwaa.com. [online] Available at: https://www.aniwaa.com/3d-scanning-technologies-and-the-3d-scanning-process/ [Accessed 10 Apr. 2018].

[21] Buildprotos.com. (2018). Contact Based 3D Scanning — Build Protos. [online] Available at: http://buildprotos.com/tag/contact-based-3d-scanning/ [Accessed 14 Apr. 2018].

[22] Anon, (2018). [online] Available at: https://www.researchgate.net/figure/General-Configuration-of-a-Laser-Triangulation-System_fig1_283108894 [Accessed 14 Apr. 2018].

[23] [online] Available at: https://www.osapublishing.org/aop/fulltext.cfm?uri=aop-3-2-128&id=211561 [Accessed 14 Apr. 2018].

[24] [online] Available at: https://www.researchgate.net/publication/37683544_Real-time_scattering_compensation_for_time-of-flight_camera [Accessed 14 Apr. 2018].

[25] [online] Available at: https://www.researchgate.net/figure/Structure-from-Motion-SfM-process-is-illustrated-The-structure-in-the_fig2_269327935 [Accessed 14 Apr. 2018].

[26] Forums.reprap.org. (2018). 3D Scanning versus 3D digitizing. [online] Available at: http://forums.reprap.org/read.php?138,68792 [Accessed 14 Apr. 2018].

[27] Slideshare.net. (2018). Introduction to Kinect - Update v 1.8. [online] Available at: https://www.slideshare.net/MatteoValoriani/introduction-to-kinect-update-v-18 [Accessed 15 Apr. 2018].

[28] [online] Available at: https://www.researchgate.net/figure/Voxel-grid-spanning-a-volume-in-a-3D-space-bounded-by-x-min-x-max-y-min-y-max_fig3_259053167 [Accessed 15 Apr. 2018].

[29] [online] Available at: http://pointclouds.org/documentation/tutorials/statistical_outlier.php [Accessed 15 Apr. 2018].

[30] Meshlab.net. (2018). MeshLab. [online] Available at: http://www.meshlab.net [Accessed 20 May 2018].

[31] python, m. (2018). module' object has no attribute 'drawMatches' opencv python. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/20259025/module-object-has-no-attribute-drawmatches-opencv-python/26227854#26227854 [Accessed 21 May 2018].

[32] Raw.github.com. (2018). [online] Available at: https://raw.github.com/PointCloudLibrary/data/master/tutorials/table_scene _lms400.pcd [Accessed 21 May. 2018].

7 ANEXO

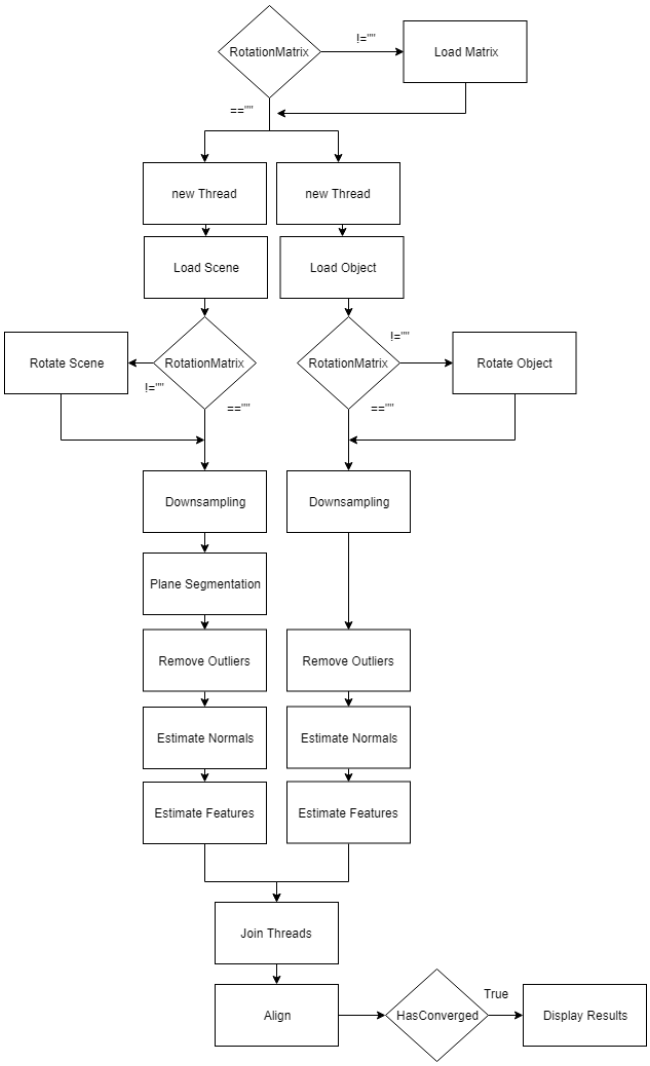

Fig. 13: Diagrama de flujo del algoritmo de localización.

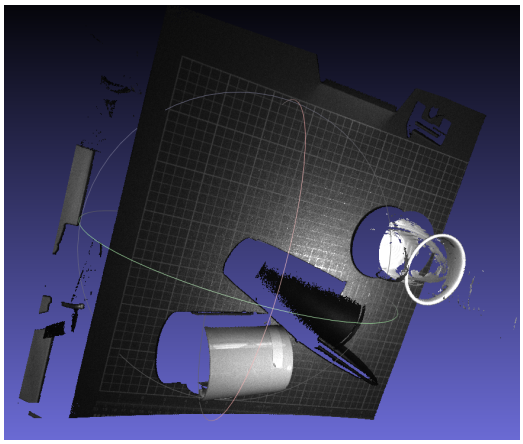

Fig. 14: Escena capturada con Photoneo PhoXi L, dos tazas y tapón de bolígrafo.

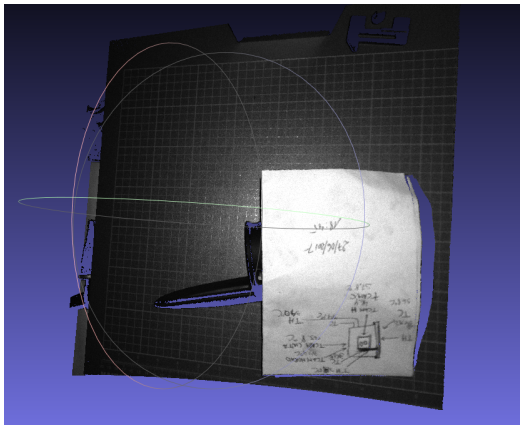

Fig. 15: Escena capturada con Photoneo PhoXi L, tapón de bolígrafo con oclusiones.

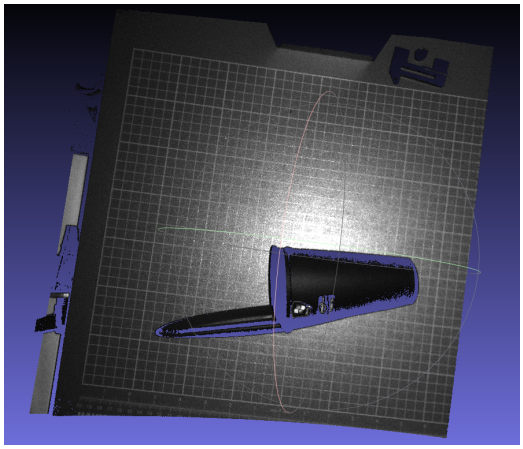

Fig. 16: Escena capturada con Photoneo PhoXi L, tapón de bolígrafo.

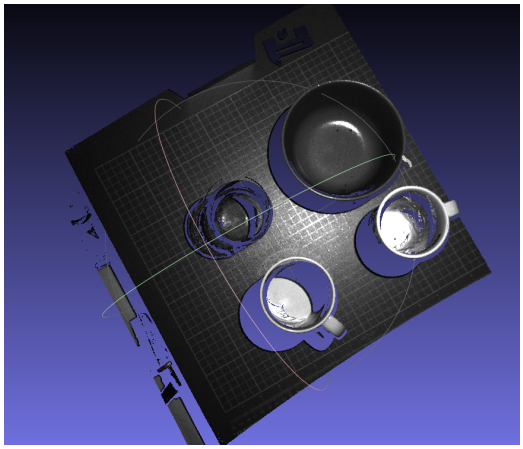

Fig. 17: Escena capturada con Photoneo PhoXi L.

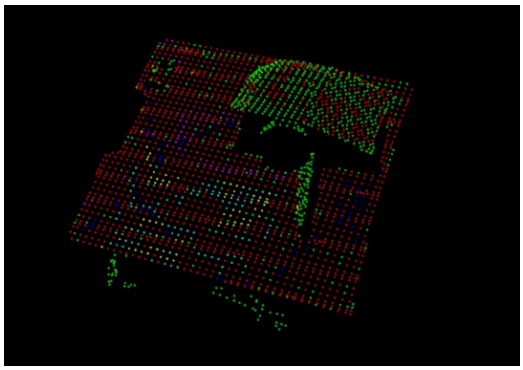

Fig. 18: Segmentación del plano de la Fig. 15. En color rojo el plano a segmentar.

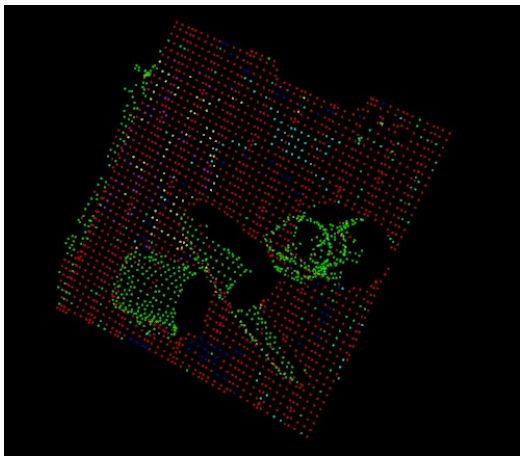

Fig. 19: Segmentación del plano de la Fig. 14. En color rojo el plano a segmentar.

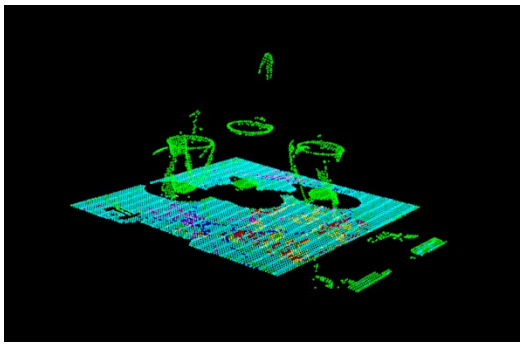

Fig. 20: Segmentación del plano de una escena adquirida con Photoneo. En color azul el plano a segmentar.

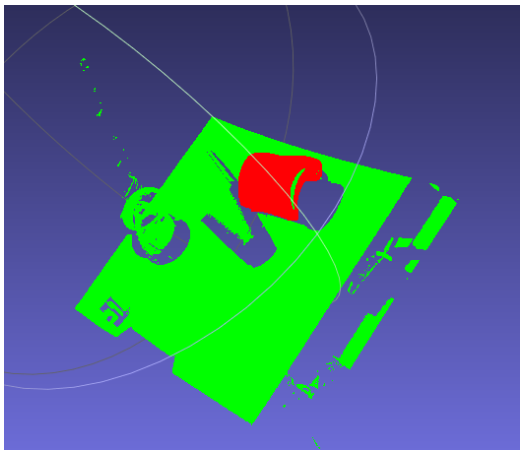

Fig. 21: Localización de una taza en una escena adquirida con Photoneo PhoXi L.

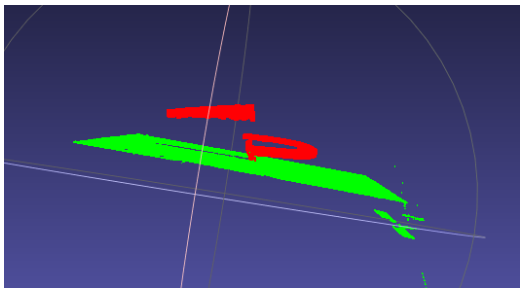

Fig. 22: Localización de un tapón de bolígrafo bic gigante en una escena adquirida con Photoneo PhoXi L.

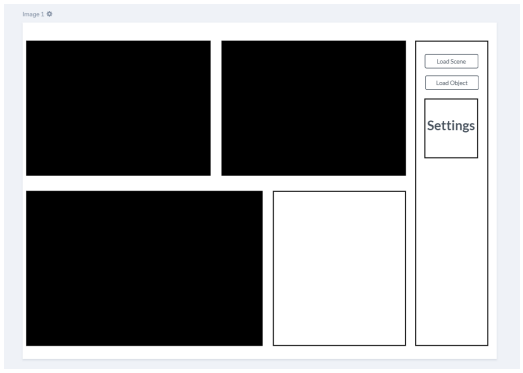

Fig. 23: Mockup de la aplicación en WPF.

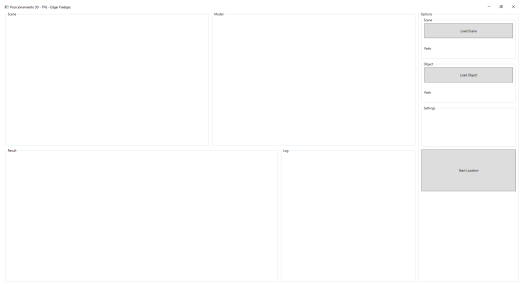

Fig. 24: Aplicación en WPF.