Procesamiento de nubes de puntos por medio de la librería PCL

Point cloud processing by PCL library

Andrés Felipe Calvo Salcedo, Arley Bejarano Martínez, Edwin Andrés Quintero Salazar
Programa de Ingeniería Electrónica, Universidad Tecnológica de Pereira, Pereira, Colombia
afcalvo@utp.edu.co
abejarano@utp.edu.co
equintero@utp.edu.co

***Resumen*— En este artículo se presentan los resultados obtenidos al utilizar la librería *PCL* para el procesamiento de nubes de puntos provenientes de escaneos 3D de 3 objetos de muestra, mediante la aplicación de filtros *voxel* y *Statistical Gross Outlier Removal*. Al ejecutar el primer filtro sobre los imágenes tridimensionales de prueba, se obtuvo una reducción en el número de puntos del 95 %, lo cual facilita la realización de procesos posteriores dada la significativa reducción en el tamaño de los datos; mientras que para el segundo filtro, se logró la eliminación de datos atípicos, permitiendo una visualización clara y definida del objeto de estudio presente dentro de la escena 3D.**

***Palabras clave*— Filtros digitales, modelos 3D, nube de puntos, PCL, reconstrucción de superficies, voxel, vecindad.**

***Abstract*— This article presents the results obtained using this PCL library, for processing point clouds from 3D scans of three objects shown, by applying *voxel* and *Statistical Outlier Removal Gross* filters. When executing the first filter on the test three-dimensional images, is obtained a reduction in the number of points of 95%, which facilitates performing subsequent processes given the significant reduction in the size of the data. In the case of the second filter, the removal was achieved uncharacteristic data, allowing a clear, defined the object of this study within the 3D scene.**

***Key Word* — Digital filter, 3D models, point cloud, PCL, surface reconstruction, voxel, neighborhood.**

I. INTRODUCCIÓN

Cuando se adquiere la información de una escena por medio de técnicas 3D, como por ejemplo *lidar*, *triangulación activa*, *tiempo de vuelo*, *estereoscopia*, entre otras [1], se obtiene una gran cantidad de información, denominada *nube de puntos*, que debe ser interpretada y procesada para determinar ciertos detalles de la escena. Aunque esta operación se podría realizar con diferentes programas tales como *MatLab®*, *LabVIew®*, entre otros, esto conllevaría una alta carga computacional, además de que no se tendría un estándar o método definido. Es allí donde toma alta relevancia la herramienta conocida como *PCL* (Point Cloud Library), la cual consiste en una librería desarrollada para el tratamiento completo de nubes de puntos 3D, liberada bajo licencia *BSD*, que cuenta con varios métodos de alta eficiencia computacional dentro de los cuales es posible destacar filtrado, reconstrucción de superficies, determinación de puntos de interés. Además de lo anterior, esta librería cuenta con el apoyo de grandes compañías como *Google*, *Trimble*, *Toyota*, *Nvidia*, *Fotonic*, entre otras.

PCL surgió ante la necesidad de que los robots estén en la capacidad de percibir el mundo tal cual y como lo hacemos los seres humanos, es decir, que puedan determinar las diferentes características y detalles que el ojo humano puede observar. Esta librería se a potencializado debido al gran al gran avance y al bajo costo de los sensores de adquisición de imágenes 3D, en el que cabe destacar en sensor *Kinect®* de la compañía *Microsoft* el cual se basa en la tecnológica *PrimeSense* [2].

Generalmente, a la información obtenida de una escena mediante algún sistema de escaneo 3D se le denomina *nube de puntos*, y se puede usar para construir modelos digitales tridimensionales que se utilizan en una amplia variedad de aplicaciones, entre ellas la metrología, automovilismo, arqueología, arquitectura, videojuegos, etc.

Las nubes de puntos, al tratarse de conjuntos de información entregada por un sistema de medida, deben pasar por distintos procesos con el fin de mejorar la confiabilidad de los datos y obtener un correcto modelamiento tridimensional en una escena determinada. Para realizar esta tarea existen diferentes librerías y aplicaciones que facilitan el trabajo y ayudan a obtener resultados óptimos, entre los cuales se destacan *Autocad*, *3DReshaper*, *Blender*, *3D Studio Ma*, *OPENGL*, *PDAL* y *MATLAB*, pero muchas de estas aplicaciones no se pueden fusionar con un sistema embebido para realizar un sistema autómata o necesitan de costosas licencias de aplicación, dificultades que en gran medida son superadas cuando el procesamiento se realiza en una herramienta tan versátil como *PCL*.

Fecha de Recepción: 28 de Septiembre de 2012
Fecha de Aceptación: 30 de Diciembre de 2012

En el presente documento se propone la utilización de la librería PCL para el procesamiento (más concretamente filtrado) de diferentes nubes de puntos provenientes del escaneo 3D de algunos objetos, con el fin de establecer las ventajas y desventajas que presenta esta herramienta computacional.

II. CONTENIDO

A. Descripción de la librería *PCL*.

PCL es una librería de *C++* desarrollada para el tratamiento de nubes de puntos en N-dimensiones desarrollado por Willow Garage con el objetivo de realizar procesamiento con numerosas técnicas. Esta librería tiene algoritmos para aplicar filtros, estimación de funciones, reconstrucción de superficies, ajustes de modelos, segmentación, entre otros. Además se encuentra liberada bajo licencia *BSD*, es decir, que es libre para uso comercial e investigativo, y sus códigos fueron desarrollados en leguaje de programación *C++*. La financiación y el soporte de esta librería se presenta gracias a grandes empresas tales como, *Nvidia*, *Google*, *Toyota*, *Trimble*, *Urban Robotics*, *Honda Research Institute* y *Sandia Intelligent Systems and Robotics*. Por otra parte, *PCL* funciona sobre varias plataformas como son *Windows*, *Linux*, *MacOS*, y *Android*. Estas herramientas ofrecen el potencial necesario para procesar y reconstruir una escena tridimensional de una manera sencilla [3]**.**

División de PCL

Para facilitar el procesamiento de la gran cantidad de puntos que resultan al realizar un escaneo de una escena 3D, *PCL* se divide en bibliotecas modulares (librerías de código más pequeños), que pueden ser compilados de una forma independiente; esto con el fin de que pueda ser compilada en las plataformas de bajo rendimiento computacional [3].

Formato de la librería PCL

El formato utilizado por la librería *PCL*, se conoce como *PCD* (Point Cloud Data). Este formato nace ante la necesidad de capturar datos obtenidos a partir de sensores y representarlos como una nube de puntos 3D. Aunque existen muchos otros formatos para representar datos 3D como *CAPA*, *STL*, *OBJ*, *X3D*, etc., estos fueron diseñados en épocas diferentes y con diferentes propósitos, por lo cual no cuentan con la capacidad de manejo de datos que si posee *PCD*.

Archivo PCD

Como se mencionó anteriormente, *PCD* es sumamente útil para guardar nubes de puntos tridimensionales y utilizar las herramientas que ofrece la librería *PCL* con el fin de procesar los datos en coordenadas cartesianas. Los archivos cargados de esta forma cuentan con una cabecera, la cual especifica los diferentes parámetros que posee la nube de puntos [4]. En la Figura 1 se muestra la información contenida en la cabecera con cada uno de sus elementos, los cuales se describen a continuación:

```
# .PCD v.7 - Point Cloud Data file format
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 1614644
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 1614644
DATA ascii
221.7507 -0.182 2.059033e-09
221.7516 -0.1849982 -0.0002089203
221.7524 -0.1819931 -0.0004110629
221.753 -0.1579865 -0.0005352776
221.7537 -0.1579759 -0.0007136883
221.7548 -0.1709593 -0.0009654847
221.7556 -0.1709414 -0.001158542
221.7565 -0.1748184 -0.001383182
```

Figura 1. Cabecera del Formato PCD [5].

- *Versión de PCD*: Define la versión de la librería y el formato del archivo a procesar.
- *Fields:* Indica las dimensiones de la nube de puntos.
- *Size:* Especifica cada tamaño del campo.
- *Type:* Define los tipos de variable.
- *Count*: Indica cuantos elementos tiene cada dimensión.
- *Width:* Representa el ancho de la nube de puntos.
- *Height:* Contiene el alto de la nube de puntos (cuando este valor es igual a 1, se dice que se trata de una nube de puntos desordenada).
- *Viewpoint:* Especifica el punto de vista como una traslación (*tx ty tz*) más los cuaterniónes (*qw qx qy qz*). El valor por defecto es: *VIEWPOINT 0 0 0 1 0 0 0.*
- *Points:* Define la cantidad de puntos de la nube.
- *Data:* Establece el tipo de dato de la nube de puntos. Para la versión 0.7, se tienen dos formatos *ASCII* y *binary*.

Herramientas de PCL

Como se puede observar en la Figura 2, *PCL* se encuentra dividido en módulos para facilitar su compilación, las cuales son:

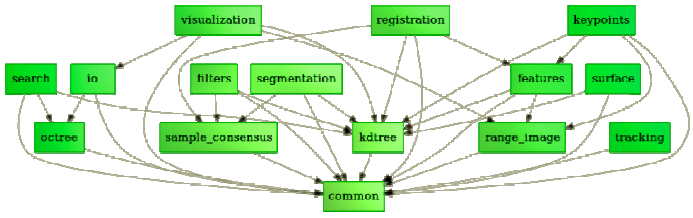

Figura 2. Organización biblioteca PCL [5].

- *Filtros (libpclfilters)*

Debido a los errores generados en la medida, es necesario utilizar herramientas que permitan eliminar datos atípicos con el fin de obtener mayor confiabilidad en los procesos a desarrollar con la información obtenida. Para esta labor se utilizan los filtros digitales. *PCL* posee métodos de filtrado digital como por ejemplo eliminadores de datos atípicos, filtro tipo *voxel*, eliminador con condición, suavizado, índices de extracción y proyecciones [6].

- *Características (libpclfeatures)*

PCL permite estimar parámetros de funciones que determinan si un conjunto de datos representa una función geométrica en el espacio. Esta biblioteca contiene funciones de las estructuras de datos y métodos para la estimación de la función en tres dimensiones a partir de los datos de una nube de puntos. La biblioteca también posibilita la extracción de características tridimensionales (como las tradicionales en las superficies y curvaturas), estimación de bordes, descripciones *PFH* (*Point Feature Histograms*) y *FPFH* (*Fast Point Feature Histograms*), rotación de imágenes, entre otras [6].

- *Puntos de Interés (libpclkeypoints)*

Esta biblioteca permite determinar los puntos clave en una escena tridimensional creada con una nube de puntos. Normalmente el número de puntos de interés en una nube es menor al total de datos, permitiendo optimizar el procesamiento de los datos [6].

- *Registro (libpclregistration)*

La librería de registro permite la construcción de un modelo tridimensional por medio de las correspondencias en *n* vistas del objeto evaluado. La combinación de *n* conjuntos de datos (nubes de puntos) en un modelo global consistente, generalmente se realiza mediante una técnica llamada registro, la cual consiste en hallar la correspondencia entre cada nube puntos y realizar una transformación para minimizar el error de alineación (tal como se puede observar en la Figura 3), y así visualizar todas las nubes de puntos como un conjunto global que posee el modelo con toda sus vistas.

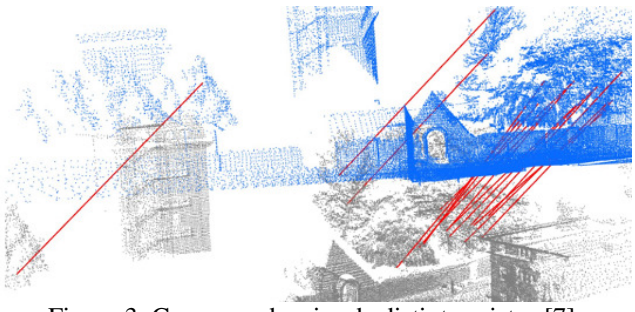

Figura 3. Correspondencias de distintas vistas [7].

- *KdTree (libpclKdtree)*

Esta biblioteca consiste en una estructura tipo árbol que almacena un conjunto de puntos *k-dimensional* con el fin de realizar búsquedas eficientes del vecino más cercano. Esta herramienta se puede utilizar para encontrar las correspondencias entre grupos de puntos para los métodos de características geométricas, o en descriptores para definir la zona local alrededor de un punto o puntos.

- *Octree (libpcloctree)*

La biblioteca *octree* proporciona métodos eficaces para la creación de una estructura de datos tipo árbol jerárquico, y subdividir nubes de puntos en conjuntos para realizar el procesamiento de forma eficiente. La aplicación ofrece rutinas eficientes de búsqueda de vecinos, tales como "*Búsqueda de vecinos Voxel*", "*El vecino más cercano a buscar determinado por un radio K*", entre otros. Estos algoritmos tienen la gran ventaja de que se ajustan automáticamente a la dimensión del conjunto de puntos.

- *Segmentación (libpclsegmentation)*

Esta biblioteca contiene algoritmos especialmente desarrollados para la segmentación de nubes de puntos. Estos algoritmos son los más adecuados para el procesamiento de datos, los cuales se componen de un número de regiones espacialmente aisladas. Een tales casos, el agrupamiento se utiliza a menudo para romper la nube en sus partes constituyentes y luego procesar cada grupo de forma independiente. Estos algoritmos se realizan a través de métodos de consenso y son útiles para una variedad de modelos paramétricos (planos, cilíndricos, esferas, líneas, etc.) [6].

- *Sample_consensus (libpclsample_consensus)*

La biblioteca *sample_consensus* contiene herramientas para estimación de modelos específicos en una nube de puntos, tales como líneas, planos, cilindros y esferas. Esta librería utiliza métodos de ajuste y estimación, como por ejemplo *RANSAC*, y modelos como planos, cilindros, etc.

- *Reconstrucción de Superficies (libpclsurface)*

Esta biblioteca se utiliza para mejorar la visualización de un modelo tridimensional procesando los puntos de una nube para obtener una representación en malla o una superficie alisada como se puede observar en la Figura 4. La complejidad de la estimación de la superficie se puede ajustar, y normales se puede estimar en la misma etapa si es necesario [6].

- *Imágenes de Rango (libpcl_rangeimage)*

Esta biblioteca permite transformar una nube de puntos en una imagen *de rango*, la cual consiste en una representación en 3D común. Este tipo de imágenes se generan con frecuencia en

sistemas estéreo o dispositivos de tiempo de vuelo. Con el conocimiento de los parámetros intrínsecos de la cámara por medio de su calibración, una imagen de rango se puede transformar en una nube de puntos para su posterior procesamiento en *PCL* [6].

Figura 4. Modelo 3D de conejo con superficie en mallas [8].

- *IO (libpclio)*

Consiste en una biblioteca que permite operaciones de lectura y escritura de archivos *PCD* (*Point Cloud Data*). También cuenta con una interfaz denominada *OpenNI Grabber Framewor*, la cual permite acceder y controlar tanto el *Kinect* (*Microsoft*) como el *Xtion-PRo* (*ASUS*).

- *Visualización (libpclvisualization)*

Esta librería permite la rápida visualización de algoritmos que operan sobre nubes de puntos 3D. Cuenta con métodos de procesamiento configuración de propiedades visuales como colores, tamaños de punto y opacidad. También cuenta con algoritmos para dibujar formas básicas en la pantalla y provee un módulo de visualización de histogramas [6].

B. Filtros en PCL.

A continuación se describen los algoritmos de filtrado contenidos en *PCL*, y que fueron aplicados a las imágenes tridimensionales tomadas mediante un escáner láser de los objetos de prueba.

Filtro Statistical Gross Outlier Removal.

Statistical Gross Outlier Removal es un filtro de la librería de *PCL* que tiene como función eliminar valores atípicos en una nube puntos basándose en análisis estadístico y en los conceptos de vencidad para entornos tridimensionales. La herramienta define la cantidad de puntos a evaluar y la desviación estándar deseada; posteriormente se calcula la media, y de esta forma se obtiene una campana de *gauss* con la que es posible determinar los límites de distancia al cual un vecino puede encontrarse. Esta campana se puede observar en la Figura 5 [9].

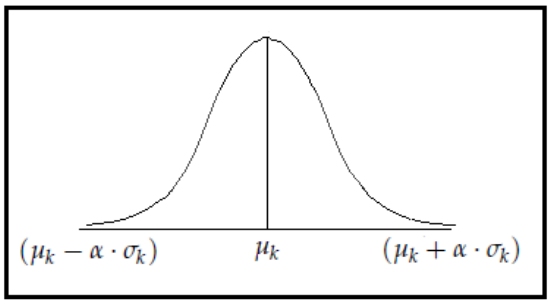

Figura 5. Curva Gaussiana Filtro Estadístico [9]

Donde:

- μ_k= Es el promedio de las distancia de la cantidad de puntos a evaluar.
- $\alpha \cdot \sigma_k$= Es la desviación máxima de la media a la que un dato puede estar.

Filtro VoxelGrid

Este filtro realiza un submuestreo de la nube de puntos por medio de una red tipo *voxel* [10], es decir, toma una cantidad de puntos y calcula su *centroide*, basado en el concepto de vecindad que se explicó anteriormente. Con estos vecinos y su centroide hace que se reduzca la cantidad de puntos, y además permite obtener un acabado en la imagen, como se puede observar en la Figura 6, esto con el fin de facilitar las siguientes etapas de procesamientos (reconstrucción de superficies y segmentación) [11].

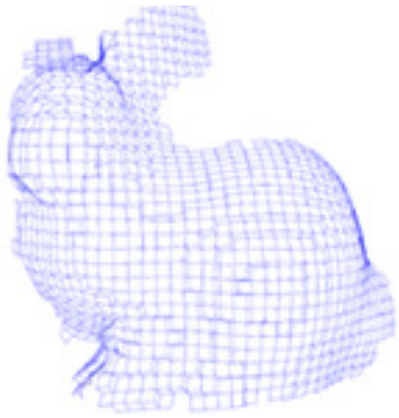

Figura 6. Modelo 3D de Conejo tipo Voxel para el ejemplo de la Figura 4 [8].

III. RESULTADOS

Los resultados que se presentan a continuación corresponden al procesamiento de nubes de puntos obtenidas de la base de datos

generada en el proyecto "*Escáner 3D para el control de calidad de piezas metalúrgicas*" realizado por los autores [9], a las cuales se les realizó un proceso de mejoramiento en la imagen por medio de herramientas de filtrado de la librería *PCL*.

En la Figura 7 se observa el escaneo de un transformador monofásico. Esta imagen presenta un exceso de información la cual conlleva un alto costo computacional y a una cantidad significativa de memoria para su almacenamiento.

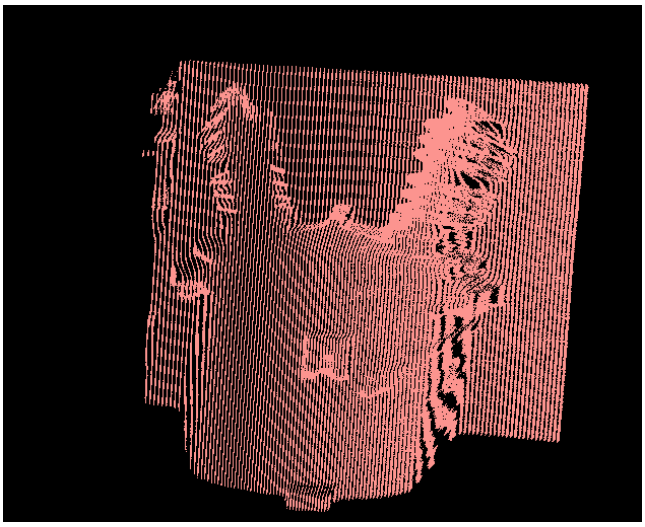

Figura 7. Modelo 3D de un transformador monofásico sin filtrar [9].

Para eliminar esta cantidad de puntos sin perder la información relevante que permite distinguir el objeto, es necesario aplicar un filtro *voxel* con el fin de reducir la cantidad de puntos en el conjunto de datos. El resultado de la aplicación de este filtro a la imagen tridimensional de la Figura 7 se presenta en la Figura 8.

En la Figura 8 se puede apreciar una disminución en el número de puntos sin perder el detalle del objeto evaluado, lo cual facilita enormemente el procesamiento posterior que se requiera aplicar sobre la imagen, ya que la cantidad de puntos disminuye considerablemente. Cabe destacar que la imagen original cuenta con una 1'325.898 puntos, mientras que la imagen filtrada solo posee 65.307 puntos, con lo cual se evidencia la efectividad del filtro *Voxel*.

Ahora bien, si lo que se desea es eliminar datos atípicos, es necesario utilizar un filtro estadístico como el *Statistical Gross Outlier Removal*, el cual, por medio de una curva *gaussiana* determina los puntos que deben ser eliminados. Para verificar la efectividad de este algoritmo incluido en PCL, se utilizó la imagen correspondiente al escaneo tridimensional de un cilindro de gas de 40 libras, la cual se presenta en la Figura 9. Por su parte, en la Figura 10 se puede apreciar el resultado obtenido al aplicar el filtro *Statistical Gross Outlier Removal* sobre la nube de puntos de la Figura 9.

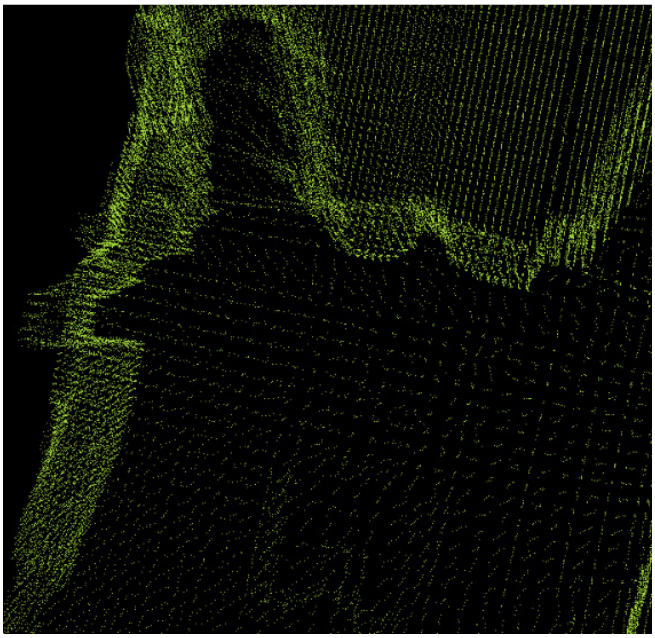

Figura 8. Resultado del filtro *voxel* sobre la imagen tridimensional de la Figura 7 [9].

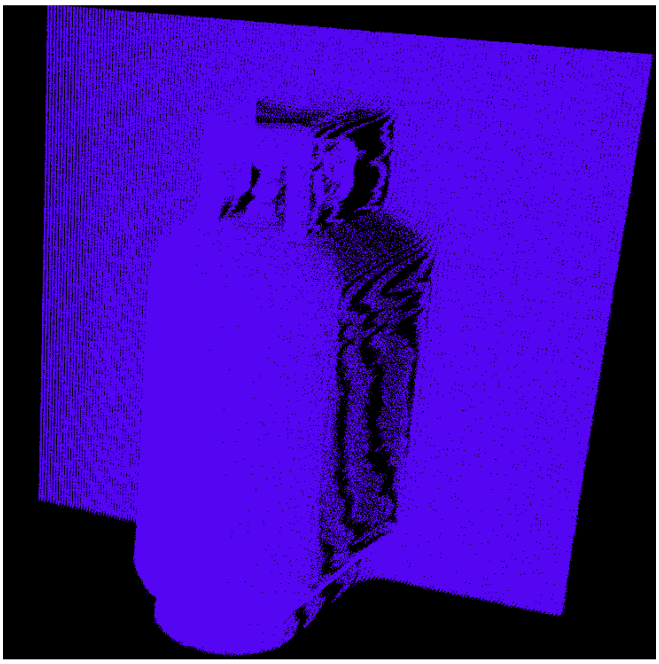

Figura 9. Escaneo tridimensional de un cilindro de 40 libras [9].

En la Figura 10 se puede apreciar que una vez adelantado el procesamiento es más notoria la curvatura del cilindro de gas, ya que se eliminaron los puntos atípicos y/o ruido presente en la escena generado por la medida del sensor. Este tipo de filtro es de gran ayuda, ya que los dispositivos que capturan este tipo de imágenes entregan escenas muy ruidosas donde no se pueden distinguir ciertas características del modelo.

El proceso de eliminación de datos atípicos se puede repetir varias veces con el fin de obtener un filtrado de orden mayor mejorando datos obtenidos por el dispositivo de captura.

En la Figura 11 se observa el resultado de un filtrado de segundo orden en otro cilindro de gas, permitiendo obtener una excelente visualización del objeto.

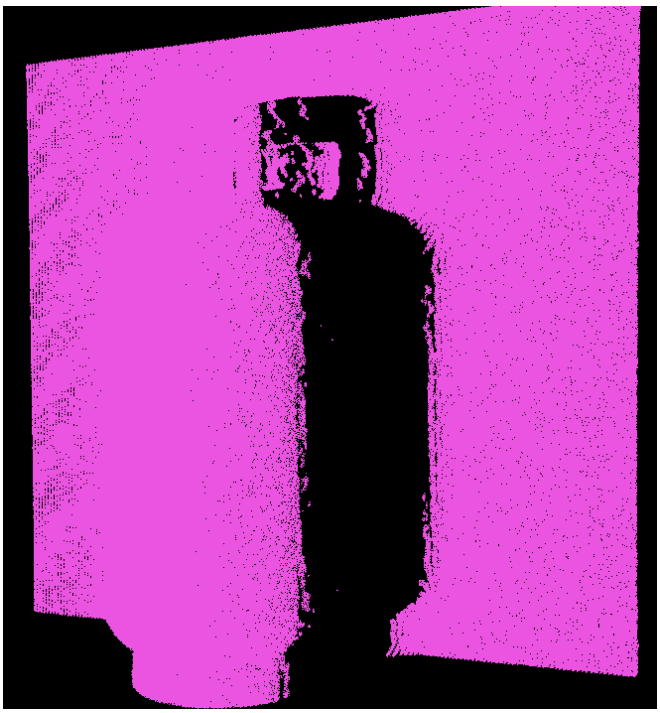

Figura 10. Resultado de aplicar el filtro *Statistical Gross Outlier Removal* sobre la imagen tridimensional de la Figura 9 [9].

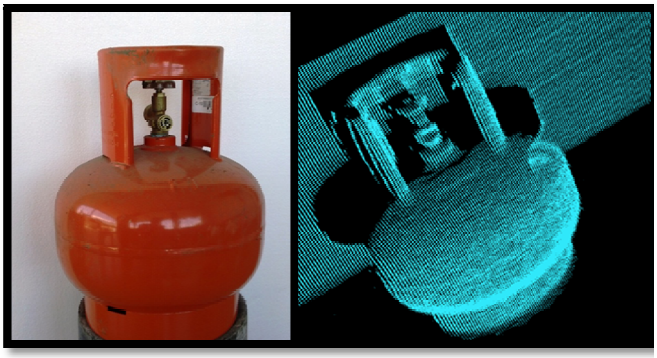

Figura 11. Filtrado de segundo orden para un modelo 3D de un cilindro de 20 libras [9].

IV. CONCLUSIONES

PCL es una excelente herramienta que cuenta con diferentes métodos para el tratamiento de nube de puntos, y al poseer una forma modular, puede ser implementado en plataformas de bajo rendimiento; además de tratarse de una biblioteca liberada bajo licencia *BSD*, lo cual conlleva un gran crecimiento en los algoritmos desarrollados. Otro aspecto es que cuenta con el respaldo de grandes compañías del área de la robótica y la visión artificial, situación que garantiza su evolución y perfeccionamiento.

Las herramientas de filtrado utilizadas en el presente proyecto demuestran las grandes ventajas que proporciona el uso de este tipo de herramientas, ya que definiendo uno o dos parámetros los métodos implementados dan como resultado una mejora bastante significativa de la imagen, tanto en la visualización como en el almacenamiento de la misma.

PCL, al tratarse de una librería y no de un software de aplicación específico, permite ser fusionada con sistemas embebidos para desarrollar aplicaciones de visión 3D en tiempo real, tales como vehículos autónomos y sistemas de metrología. Esto explica porque grandes compañías y centros académicos y de investigación del mundo centran el procesamiento de las nubes de puntos procedentes de sistemas robóticos o de visión artificial sobre *PCL*.

REFERENCIAS

[1] GARCÍA, J. L. L. Y TARONGERS, J. M. B. Teoría y práctica del escaneado láser terrestre. Informe técnico, 3D RiskMapping. 2008.

[2] RADU BOGDAN RUSU, 3D is here: Point Cloud Library (PCL), Technische Universitaet Muenchen, Alemania, Octubre de 2009.

[3] WILLOW GARAGE. What is PCL?. [En línea]. Disponible en: <http:// pointclouds.org/about.html> [Consulta 20 de Marzo de 2012].

[4] WILLOW GARAGE. The PCD (Point Cloud Data) file format. [En línea]. Disponible en: <http://pointclouds.org/documentation/tutorials/pcdfileformat.php>. [Consulta 30 de marzo de 2012].

[5] WILLOW GARAGE. What is PCL?. [En línea]. Disponible en: <http:// pointclouds.org/about.html> [Consulta 20 de Marzo de 2012].

[6] WILLOW GARAGE. PCL documentation and tutorials. [En línea]. Disponible en: <http://pointclouds.org/documentation/> [Consulta 22 de Marzo de 2012].

[7] WILLOW GARAGE. Module registration. [En línea]. Disponible en: <http://docs.pointclouds.org/trunk/group__registration.html> [Consulta 22 de Marzo de 2012].

[8] WILLOW GARAGE. Module Surface. [En línea]. Disponible en: <http://docs.pointclouds.org/trunk/group__surface.html> [Consulta 22 de Marzo de 2012].

[9] ARLEY BEJARANO M y ANDRES F. CALVO. Escáner 3D para el control de calidad de piezas metalúrgicas. Universidad Tecnológica de Pereira. Agosto de 2012.

[10] RADU BOGDAN RUSU, Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments, Computer Science department, Technische Universitaet Muenchen, Alemania, Octubre de 2009

[11] RADU B. RUSU. Downsampling a PointCloud using a VoxelGrid filter. [En línea]. Disponible en:<http://pointclouds. org/documentation/tutorials/voxelgrid.php#voxelgrid> [Consulta 1 de Abril de 2012].