



FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

RECONSTRUCCIÓN 3D DE OBJETOS USANDO SENSOR KINECT v.2.

Autor

Hernán Andrés Baldeón Guanín

Año  
2018



FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

RECONSTRUCCIÓN 3D DE OBJETOS USANDO SENSOR KINECT v.2.

Trabajo de Titulación presentado en conformidad a los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información.

Profesor Guía

MSc. David Fernando Pozo Espín

Autor

Hernán Andrés Baldeón Guanín

Año

2018

## DECLARACIÓN DEL PROFESOR GUÍA.

“Declaro haber dirigido el trabajo, RECONSTRUCCIÓN 3D DE OBJETOS USANDO SENSOR KINECT v.2, a través de reuniones periódicas con el estudiante, Hernán Andrés Baldeón Guanín, en el semestre 2018-2 orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

David Fernando Pozo Espín

Máster en Automática y Robótica

C.I. 171734014-3

## DECLARACIÓN DEL PROFESOR CORRECTOR

“Declaro haber revisado este trabajo, RECONSTRUCCIÓN 3D DE OBJETOS USANDO SENSOR KINECT v.2, del estudiante, Hernán Andrés Baldeón Guanín, en el semestre 2018-2 dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

Jorge Luis Rosero Beltrán

Máster en Ciencias con Especialidad en Automatización

C.I. 180361018-5

## DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de los autores vigentes”.

Hernán Andrés Baldeón Guanín

C.I. 172255400-1

## AGRADECIMIENTO

A Dios por guiarme siempre en el camino. A mi familia por su amor y apoyo en cada momento de mi vida. Al MSc. David Pozo y MSc. Jorge Rosero por su amistad, soporte y dirección incondicional en el desarrollo de este trabajo. A la Unidad de Innovación Tecnológica de la Universidad de las Américas y a su encargado principal Ing. Santiago Solórzano. A mis compañeros David Ponce y Kevin Jaramillo por su oportuna asistencia cuando fue requerida.

Andrés Baldeón.

## DEDICATORIA

Dedico este trabajo a mis padres por el esfuerzo que ha significado mi formación tanto personal como profesional. A mi hermana Natalia quien es mi cómplice, consejera y motivación para ser mejor. A María José quien ha sido un ejemplo y motivación para alcanzar mis metas.

Para todos ustedes con mucho cariño todo mi esfuerzo plasmado en la culminación de mis estudios de pregrado.

Andrés Baldeón.

## RESUMEN

El objetivo del presente trabajo de titulación es reconstruir objetos regulares e irregulares en tres dimensiones por medio de una nube de puntos obtenidos por un sensor Kinect v.2 desarrollado por Microsoft.

Para lograrlo se analiza las imágenes capturadas por los sensores de color y profundidad del mencionado dispositivo en el uso de la reconstrucción de objetos en tres dimensiones. Tomando como herramientas criterios matemáticos de matrices de rotación y traslación, uniéndolas para obtener matrices de transformación homogéneas, que conjugadas con las facilidades que nos entrega dicha herramienta tecnológica y el software Matlab 2017 para procesar la información.

En un inicio se realizó una captura de un florero ubicado sobre una mesa frente a una pared; es decir, con una fotografía en tres dimensiones de todo el espacio que se encontraba frente al dispositivo en ese momento. Dicha nube de puntos posee seis variables, entre las que se destaca la matriz *Location*, en la que se almacenan las coordenadas en los tres ejes de cada punto, esta matriz será manejada por medio de los criterios matemáticos previamente mencionados.

Una vez comprobado el funcionamiento del sistema se coloca en la plataforma diseñada y se ubica el objeto sobre el mecanismo de giro. Se realizan cuatro escaneos de un objeto, rotando  $90^\circ$  entre cada uno. Luego, cada captura se procesa de tal manera que se acumularán todas las matrices *Location* en una misma matriz con la intención que confluyan todas las capturas al final del proceso y se muestre en una interfaz gráfica de Matlab.

Finalmente, como resultado se obtiene un objeto digitalizado en tres dimensiones que puede ser rotado para observar sus lados, manteniendo sus proporciones y detalles visibles al sensor.



## ABSTRACT

The objective of this work is the reconstruction of regular and irregular objects in three dimensions by a mix of cloud of points obtained by a Kinect v.2 sensor developed by Microsoft.

To achieve this, the images captured by the color and depth sensors of the device are analyzed in the use of the reconstruction of objects in three dimensions. Taking as mathematical tools of rotation and translation matrices criteria, uniting them to obtain homogenous transformed matrices, which mixed with the facilities that this technological tool gives us and the Matlab 2017 software to process the information.

At the beginning, a vase was captured on a table in front of a wall; that is, a three-dimensional photograph of all the space that was in front of the device at that moment. This cloud of points has six variables, among which the *Location* matrix stands out, in which the coordinates are stored in the three axes of each point, this matrix will be handled by means of the mentioned mathematical criteria.

Once the functioning of the system has been checked, it is placed on the designed platform and the object is located on the turning mechanism. Four scans of an object are made, rotating  $90^\circ$  between each one. Then, each capture is processed in such a way that all the *Location* matrices are accumulated in the same matrix with the intention that all the captures converge at the end of the process and be displayed in a graphical interface of Matlab.

Finally, a digitalized object in three dimensions is obtained, that can be rotated to observe its sides, keeping its proportions and details visible to the sensor.

## ÍNDICE

1. Introducción.....	1
1.1 Tecnologías similares.....	1
1.1.1 Leap Motion .....	1
1.1.2 Softkinetic DepthSense .....	2
1.1.3 Xtion Pro.....	3
1.1.4 Zed 2K Stereo camera.....	4
1.1.5 Láser.....	5
1.1.6 Luz estructurada.....	6
1.2 Tecnología Kinect .....	6
1.3 Alcance .....	10
1.4 Justificación.....	10
1.5 Objetivo general .....	11
1.6 Objetivos específicos .....	11
2. Marco teórico.....	12
2.1 Hardware aplicado a reconstrucción 3D.....	12
2.1.1 Funcionamiento de una cámara.....	12
2.1.2 Imágenes digitales .....	12
2.1.3 Funcionamiento del dispositivo Kinect .....	14
2.1.3.1 Cámara Kinect v.2.....	16
2.1.3.2 Sensores infrarrojos Kinect v.2.....	17
2.1.3.3 Arreglo de micrófonos Kinect v.2.....	17
2.2 Matemáticas aplicadas a reconstrucción 3D .....	18
2.2.1 Matrices de rotación.....	18
2.2.2 Matrices de traslación .....	20
2.2.3 Matrices de transformación homogénea.....	21
2.3 Software aplicado a reconstrucción 3D.....	22
2.3.1 Datos obtenidos con información de varios sensores.....	22
2.3.1.1 Skeletons.....	22
2.3.1.2 Medidor de pulso cardíaco .....	23
2.3.2 Point Cloud .....	24
2.3.3 Procesamiento de imágenes 3D .....	24
2.3.4 Reconstrucción en tres dimensiones .....	26
3. Metodología.....	27
3.1 Conexión con el microcontrolador .....	28

3.2	Software y librería .....	28
3.3	Aplicación de Matlab para captura de datos .....	29
3.4	Manejo de los datos capturados .....	30
4.	Resultados y discusión.....	36
5.	Conclusiones y recomendaciones.....	49
5.1	Conclusiones.....	49
5.2	Recomendaciones.....	49
	Referencias .....	51
	Anexos .....	57

## 1. Introducción

La tecnología Kinect de Microsoft fue implementada como accesorio de la consola de videojuegos Xbox 360, el objetivo principal de este elemento es reconocer movimientos de las personas a través de una gran cantidad de rayos láser a una distancia determinada y transmitirlos a la consola para que el personaje realice los mismos movimientos.

La digitalización de personas, objetos o entornos se ha convertido en una necesidad para seguir a la par y dar abasto a las demandas de las nuevas tecnologías que avanzan a pasos agigantados, por lo cual existen diferentes productos en el mercado que poseen características parecidas a las del dispositivo Kinect.

### 1.1 Tecnologías similares

Como se mencionó anteriormente, existen diferentes tecnologías que tienen como objeto realizar captura de movimientos, elementos o entornos a través de diferentes sensores, mismos que se encuentra detalladas a continuación:

#### 1.1.1 Leap Motion

Es un pequeño dispositivo de escritorio que conectado a un computador se ubica bajo manos y brazos para interactuar con el equipo mediante movimiento de estas extremidades, sin necesidad de tocar ninguno de sus dispositivos de entrada. Tiene forma rectangular, como se observa en la Figura 1, y puede ser conectado a un puerto USB 2.0 o 3.0.

Su funcionamiento está basado en principios capturas de imágenes infrarrojas, por lo que no tienen incidencia de agentes externos y tampoco interferencias,

por lo que tiene una precisión de 0.01 milímetros. El *software* (SDK) se encarga de lo relacionado a la conexión USB y de transmisión de información del sensor al computador incluyendo detalles de bajo nivel, por otro lado, los sensores del dispositivo son capaces de capturar información espacial sobre la orientación y el movimiento de las manos, por lo que el SDK provee información del sensor en forma de eventos. Las tres categorías de eventos son: eventos gesticulares, eventos espaciales y velocidad de eventos. Esta tecnología fue aplicada en el manejo de un brazo robótico mediante gestos y señales (Chen, Chen, Zhou, & Yan, 2017).



*Figura 1.* Leap motion.

Tomado de Leap Motion, 2018.

### 1.1.2 Softkinetic DepthSense

Softkinetic es una empresa adquirida por Sony que desarrolló la tecnología *time-of-flight*, que utiliza el movimiento en tiempo real para emplearlo en el control de plataformas.

El principio de esta tecnología se deriva de un ángulo que se compone por los ejes ópticos, que envían haces de luz infrarroja, durante el movimiento del efector final. Este dispositivo posee un *nivel de confianza* mismo que indica qué rango de profundidad capturada es confiable. Este tipo de tecnología ha sido utilizado también en la manipulación de un robot por medio de movimientos de una persona guía (Wongwilai, Niparnan, & Sudsang).

En la Figura 2 se puede apreciar el dispositivo DS525, mientras que entre los modelos comercializados aparecen DS311 y DS325, mismos que poseen una resolución relativamente baja de 320x240 píxeles.



*Figura 2.* SoftKinetic DS525.

Tomado de SoftKinetic, 2018.

### 1.1.3 Xtion Pro

Es un dispositivo desarrollado por PrimeSense y ASUS que posee prácticamente las mismas características de hardware que la primera versión del Kinect con la particularidad de que no necesita alimentación de energía. Posee una cámara RGB-D que tiene la capacidad de enviar información por píxel. Tiene un proyector y un receptor infrarrojo que son los encargados de enviar y capturar la nube de puntos generada, como se observa en la Figura 3. Para procesar la información receptada se utilizan varios métodos de triangulación para lograr captar la imagen en tres dimensiones.

Es compatible con la gran mayoría de sistemas operativos. Se pensó para que los desarrolladores realicen soluciones para cualquier tipo de computadores.

Este dispositivo ha sido utilizado en el reconocimiento de un entorno en 360° de interiores, con la intención de utilizarlo en robots autónomos de navegación automática (Kundu, Mazumder, Dhar, & Bhaumik, 2016).



*Figura 3.* Xtion Pro LIVE.

Tomado de ASUS, 2018.

#### 1.1.4 Zed 2K Stereo camera

Se trata de un dispositivo que captura un mapa en 3D del entorno que se escanea, a la vez que también entiende el movimiento que se realiza. Al igual que la visión humana, esta cámara agrega percepción de profundidad y posición, como se muestra en la Figura 4. Puede procesar distancias de 0,5 metros a 20 metros, a 100 FPS indistinto si está ubicado en interiores o exteriores (Stereo Labs, s.f.).

Al ser una tecnología prácticamente nueva no se dispone de muchos datos y características técnicas sobre el dispositivo; de igual manera no se encuentran desarrollos empleando este dispositivo que se hayan realizado o se encuentren en curso.



*Figura 4.* ZED 2K Stereo Camera.

Tomado de Stereo Labs.

#### 1.1.5 Láser

Es un sistema compuesto por un emisor y uno o varios receptores, envía y recibe el rayo láser después de chocar contra la superficie del objeto, basado en el tiempo de vuelo del rayo se estima una distancia aproximada entre el sistema y el mencionado objeto (Ilbay Paca, 2014).

Uno de los dispositivos que emplea esta tecnología es el sensor URG-04-LX-UG01, una ilustración del mismo se aprecia en la Figura 5.



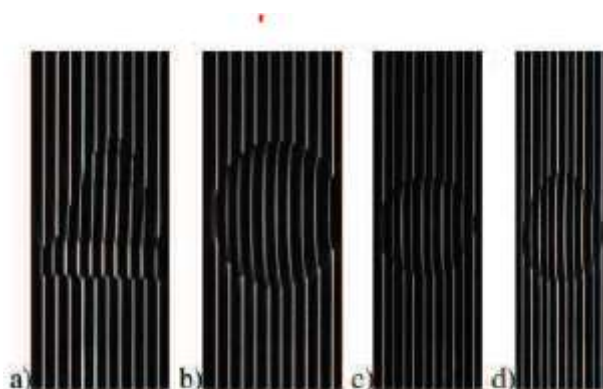
*Figura 5.* Sensor láser URG-04-LX-UG01.

Tomado de Ilbay Paca, 2014.



### 1.1.6 Luz estructurada

Es un tipo de tecnología que se basa en la proyección de un patrón de luz establecido hacia un objeto, analizando la deformación de dicho patrón para obtener un modelo aproximado como se observa en la Figura 6. El reflejo producido se captura con ayuda de una cámara fotográfica y luego con la aplicación de algoritmos se determina la posición aproximada de cada punto encontrado en el espacio (Ilbay Paca, 2014).



*Figura 6.* Imagen de Líneas proyectadas sobre diferentes objetos.

Tomado de Ilbay Paca, 2014









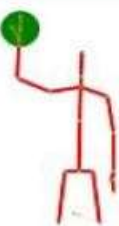



## 1.2 Tecnología Kinect

Aproximadamente un año atrás se realizó el lanzamiento de la segunda versión del dispositivo Kinect, que entre sus mejoras presenta una mayor resolución en tanto a la nube de puntos láser para detectar de mejor manera los movimientos y gestos de los usuarios; información más detallada se encuentra más adelante en el documento.

Se han realizado un sin número de desarrollos con la primera versión de este sensor, pero debido a sus limitaciones no era posible realizar proyectos que demanden una mayor resolución, ahora con esta mejora se tiene un nuevo abanico de posibilidades.

Entre los desarrollos existentes con las dos versiones del dispositivo Kinect se encuentran los siguientes:

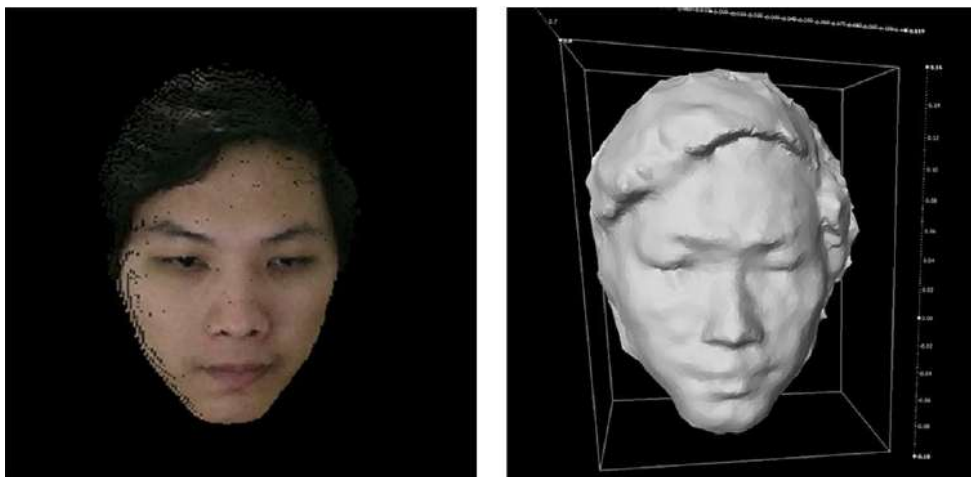
En el año 2015, se realizó un análisis de la percepción del sensor de profundidad de Kinect para la detección de distancia de vehículos, el cual tuvo como objetivo utilizar este dispositivo como ayuda al conductor en la detección de objetos para evitar colisiones, añadiendo una alarma para que el conductor pueda reaccionar oportunamente mientras se encuentre manejando (Abdullah, Zabidi, Yassin, & Hassan, 2015). Otro de los temas es el de desarrollar un pizarrón interactivo de bajo costo usando el sensor Kinect, mediante el uso de un lápiz LED ubicado en una posición determinada para que el dispositivo pueda analizar su profundidad y los datos infrarrojos recibidos (Zhang, He, Yu, & Zheng). Adicional al Kinect se utilizaron una plataforma CUDA y una librería OpenCV para un procesamiento veloz de las imágenes de tal manera que se pueda obtener un desempeño en tiempo real. Adicionalmente, se encuentra una implementación de sincronización de movimiento de múltiples personas y robot humanoide utilizando un dispositivo Kinect v.2; en aquel se muestra la manera en la que el sensor ubica a la persona que habla y captura el *skeleton* de su cuerpo y extremidades a la vez que envía a un microcontrolador para que este envíe las indicaciones al robot humanoide el cual como consiguiente realiza los movimientos que realizó la persona (Akhil & Parimi, 2016). Finalmente, se desarrolló también un sistema de control de vuelo de un vehículo aéreo no tripulado (*UAV por sus siglas en inglés*) basado en Kinect v.2; la intención es enviar indicaciones al *UAV* mediante gestos y posturas como las mostradas en la Figura 7 en lugar de utilizar un control remoto. La plataforma de evaluación fue basada en Kinect v.2 y Ar. Drone 2.0. (Dong, Ciao, Zhang, & Guo, 2016)

Command Up	Command Down	Command Forwards	Command Backwards
			
Command Leftwards	Command Rightwards	Command Turn Left	Command Turn Right
			
Command Take Off	Command Land	Command BLinkGreenRed	Command Flip Right
			

*Figura 7.* Gestos y posturas utilizados para controlar el UAV.

Tomado de Dong, Ciao, Zhang, & Guo, 2016.

Pasando al campo de modelado, existen investigaciones previas que se basaron en reconstrucción de interiores con ayuda de una cámara láser, obteniendo información no muy detallada de las zonas en cuestión (Atman & Trommer); así como también se tiene como ejemplo un desarrollo el cual se implementó un sistema para modelar 360° en interiores con color y profundidad utilizando el dispositivo Kinect v.2 (Dulko). Adicional a modelado de interiores, se puede puntualizar que las características del dispositivo permite modelar o reconstruir objetos en tres dimensiones, un ejemplo de esto es la reconstrucción de un rostro humano con todos sus detalles como se puede observar en la Figura 8, además se puede tener una imagen muy acercada a la realidad utilizando distintos métodos de filtrado de los datos capturados (Siv, Ardiyanto, & Hartanto, 2018).



*Figura 8.* Reconstrucción 3D de un rostro humano.

Tomado de Siv, Ardiyanto, & Hartanto, 2018.

Sin embargo, los sensores infrarrojos no son los únicos empleados en reconstrucciones en tres dimensiones, un ejemplo de aquello es la investigación sobre la reconstrucción de una habitación por medio de sensores audiovisuales, en el cual se ha utilizado un parlante para enviar las ondas y un micrófono para receptorlas, con este método se pretende detectar objetos de ciertos materiales que utilizando sensores infrarrojos sería muy complicado hacerlo como por ejemplo, vidrio, plástico transparente o ventanas (Kim, Remaggi, Jackson, Fazi, & Hilton, 2017). Finalmente, otro de los métodos es con el uso de sensores térmicos cuya característica es que poseen un escaneo de 360°, en dicha ocasión los objetivos son personas que se encuentran alrededor de dicho dispositivo (Benli, Rahlf, & Motai, 2018).

Estos dispositivos también tienen diversas aplicaciones en el campo de la medicina y como muestra se encuentra una investigación sobre telerehabilitación, término utilizado para señalar tratamientos de rehabilitación complementaria realizada a distancia, sin necesidad de un profesional que supervise dicho procedimiento; en esta ocasión está orientada a la telerehabilitación de pacientes con daño a nivel articular, muscular y de tendones en hombros mediante seis diferentes ejercicios (Çubukçu & Yüzgeç, 2017).

Adicionalmente, este dispositivo fue creado con el objetivo de ser un accesorio de entrada a una consola, por lo que en desarrollos de robots autómatas también ha sido utilizado de esta manera. Para muestra de eso se tiene la investigación de Kameyama y Hidaka, en la cual el Kinect es ubicado en un robot explorador, mismo que recorre un área determinada y almacena las rutas realizadas y los sectores ya recorridos para de esta manera cubrir el área determinada de una manera eficiente (Kameyama & Hidaka, 2017).

### 1.3 Alcance

Realizar un reconocimiento en tres dimensiones de un objeto mediante la adquisición de las nubes de puntos recibidas mediante el sensor Kinect v.2. Para cumplir este objetivo se contará con la ayuda de una plataforma y un mecanismo (rotor) que realice tres giros de 90° cada uno, de tal manera que se obtenga una reconstrucción total del cuerpo. Dicha reconstrucción se realizará por medio de un software de escritorio el cual fusionará las capturas realizadas por el sensor, que a la vez serán filtradas, para que el resultado desplegado en tres dimensiones a través de la interfaz gráfica del computador sea lo más precisa posible. Los objetos a reconstruirse serán cuerpos con una geometría establecida, con el fin de comprobar dimensiones y forma de los mismos. Además, se reconstruirán cuerpos con distintas formas para comprobar el correcto funcionamiento del sistema.

### 1.4 Justificación

La reconstrucción en tres dimensiones de un objeto es muy útil actualmente para que en el campo de la manufactura se simplifique el proceso de copiar elementos para fabricarlos nuevamente. Entre los nuevos desarrollos en tecnología es la base para conducción autónoma de robots que utilizan

inteligencia artificial, interacción con robots o *UAVs* y manipulación de los mismos.

Adicionalmente, será el inicio para reconstrucciones cada vez más grandes, más precisas y más fáciles de implementar que serán de utilidad para nuevos campos de desarrollo de aplicaciones o robots que sean desarrollados por la universidad o por agentes externos.

Un objeto reconstruido digitalmente puede ser de gran utilidad, desde una captura de datos de un robot autómatas hasta la creación de entornos virtuales para recorridos de información. Como se mencionó con anterioridad se han realizado muchos desarrollos e investigaciones con la versión uno del sensor, ahora con la nueva versión del Kinect no existen muchos aún debido a que es un tiempo corto desde que fue lanzado al mercado, por lo que es un proyecto ambicioso y actualizado.

### 1.5 Objetivo general

Reconstruir objetos regulares e irregulares en tres dimensiones por medio de una nube de puntos obtenidos por un sensor Kinect v.2.

### 1.6 Objetivos específicos

- Investigar el funcionamiento del sensor Kinect v.2 y la manera en que adquiere una nube de puntos.
- Diseñar e implementar un mecanismo que permita realizar tres giros de 90° cada uno frente al sensor Kinect v.2 para que este pueda escanear toda la figura.

- Realizar el procesamiento de la nube de puntos para obtener la reconstrucción en tres dimensiones.

## 2. Marco teórico

### 2.1 Hardware aplicado a reconstrucción 3D

### 2.2 Funcionamiento de una cámara

Una cámara fotográfica se encarga de recibir los rayos de luz del exterior y lo transforma en una imagen. Los rayos de luz son agrupados en un solo punto después de ingresar por el lente y la imagen final será más pequeña o más grande de su tamaño natural. Al igual que en el ojo humano, en las cámaras la imagen se captura al revés, esto se debe a fenómenos físicos de la reflexión de la luz como se puede apreciar en la Figura 9.

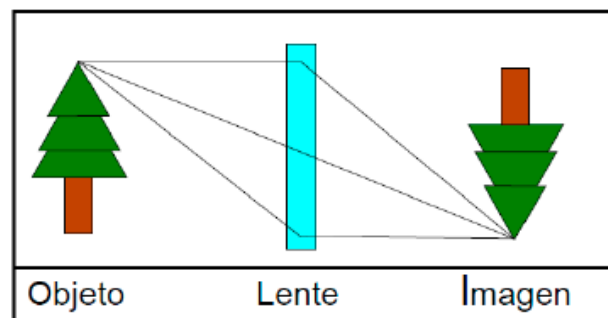


Figura 9. Funcionamiento de una cámara.

Tomado de Villegas, 2016.

### 2.3 Imágenes digitales

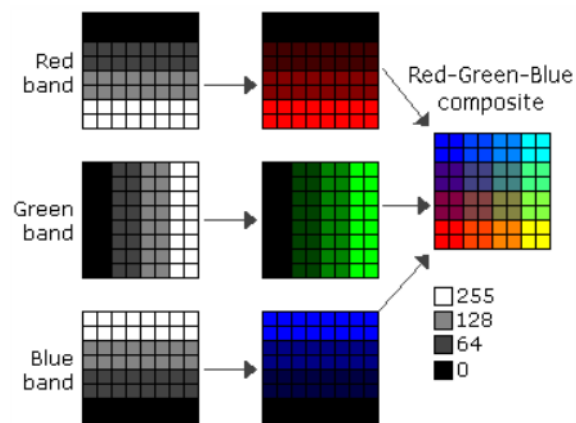
Una imagen digital es una matriz bidimensional ordenada de valores ubicados ordenadamente, estos valores son conocidos como *píxeles* y son la subdivisión más pequeña que puede tener la composición de una imagen. En imágenes que son mostradas en escala de grises se pueden tener 255 capas se observa en la Figura 10 de la siguiente manera:



*Figura 10.* Estructura de una imagen en escala de grises.

Tomado de Teledet.

Matlab permite capturar estas imágenes y extraerlas en tres matrices bidimensionales, una por cada color *Red*, *Green* y *Blue* según se muestra en la Figura 11, conocidos como imágenes RGB. Estos colores se mezclan para lograr una tonalidad, saturación o color de un píxel para que el mismo conforme una imagen de tal manera que formando parte de un todo tome el sentido.



*Figura 11.* Composición de una imagen RGB.

Tomado de Teledet.

Como se indica en la página web de *Universe*, el píxel es la unidad más mínima de información de la imagen, mismo que posee datos de color,



saturación y brillo. Este término es utilizado cuando se habla de imágenes digitales, como se observa en la Figura 12, ya que estas están conformadas por millones de píxeles, también conocido como mapa de bits (Rodríguez, 2017).

Es importante señalar que un píxel puede contener únicamente un color, y este color puede ser rojo, verde, azul o una mezcla entre estos colores. El arreglo ordenado de miles o millones de estos píxeles son las imágenes digitales que podemos observar en nuestros dispositivos electrónicos.



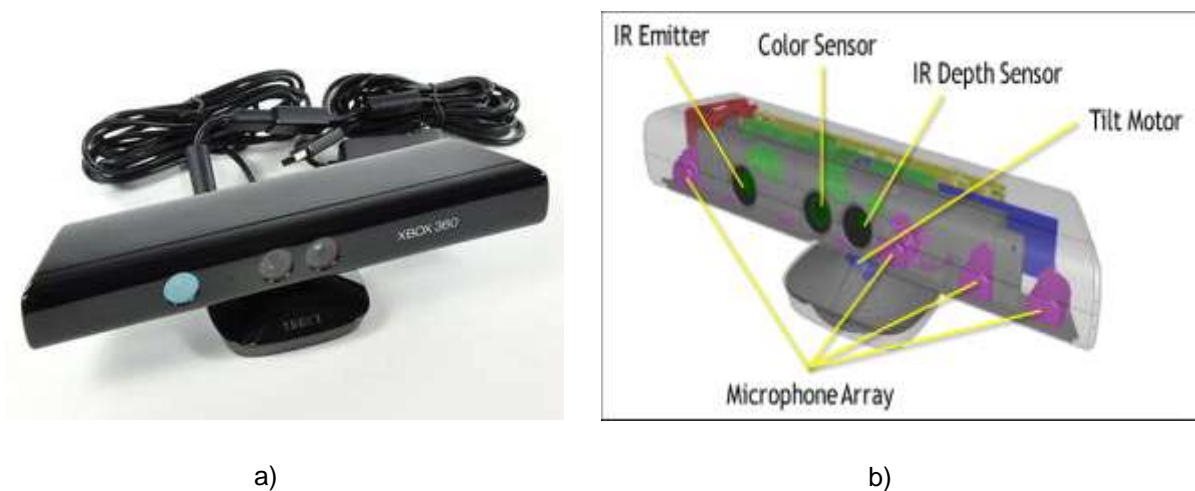
*Figura 12.* Demostración de píxeles en una imagen digital.

Tomado de Rodríguez, 2017.

#### 2.4 Funcionamiento del dispositivo Kinect

La primera versión del dispositivo de Microsoft fue presentada en el año 2009 en la convención E3 bajo el nombre de *Proyecto Natal*. Su venta inició en noviembre de 2010 con su nombre comercial *Kinect*, lo que causó un gran

revuelo ya que fue el primer accesorio en su tipo, que era manejado sin necesidad de dispositivo de entrada. En su primera versión, el dispositivo presentaba un proyector infrarrojo y una cámara RGB con una resolución 640x480 píxeles, la información capturada era procesada por un microchip diseñado para capturar el movimiento de objetos y personas en tres dimensiones. Además, un conjunto de micrófonos instalados en los costados. En la Figura 13 a continuación se puede apreciar un ejemplo del dispositivo, así como su diagrama de sensores.

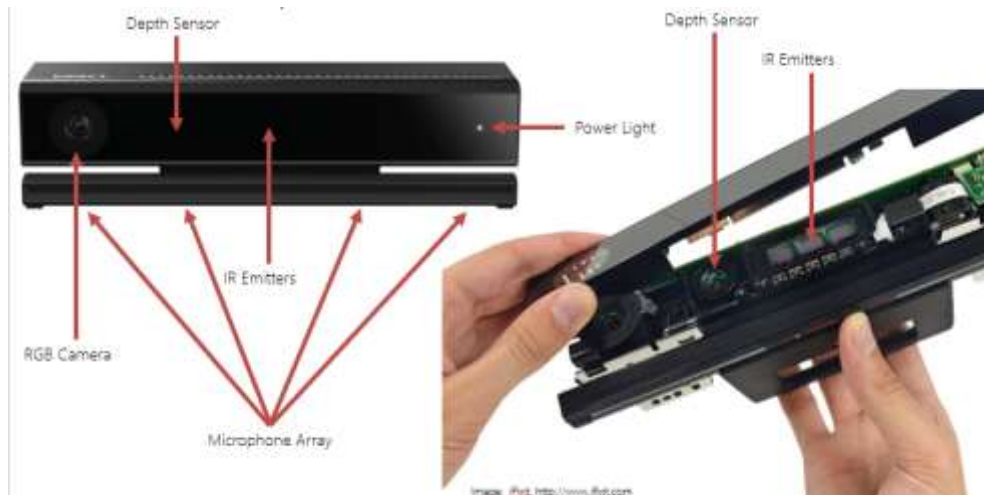


*Figura 13.* Tomas del sensor a utilizar.

Tomado de Microsoft, 2009.

- a) Dispositivo Kinect v.1 para Xbox 360.
- b) Diagrama de sensores del Kinect v.1.

En febrero de 2012, se lanzó un sensor Kinect más enfocado a desarrolladores, con una cámara RGB de mayor resolución, un emisor de rayos infrarrojo, un sensor de profundidad infrarrojo, arreglos de micrófonos para capturar sonido y encontrar la fuente sonora y un acelerómetro de 3 ejes capaz de medir la aceleración en 3 dimensiones. Además, incorpora un time-of-flight (TOF) de alta resolución lo que permite obtener detalles con mayor precisión, además de tener un campo de visión 60% más amplio logrando que se pueda registrar hasta 6 personas en simultáneo, reconociéndolas, así como sus movimientos (Microsoft, 2018). En la Figura 14 se puede observar un ejemplo del dispositivo Kinect v.2 y sus sensores interiores.



*Figura 14.* Componentes de un dispositivo Kinect v.2.

Tomado de Duque, 2015.

Este dispositivo utiliza una interfaz natural de usuario (NUI) a través de la cual interactúa con un sistema. Interfaz natural es la que no utiliza ningún elemento de entrada (mouse, teclado, etc.), el usuario realiza movimientos naturales del cuerpo con sus extremidades, gestos o incluso comandos de voz (Mendoza Rivera & Guamushig Lasluisa, 2017).

A continuación, se tiene una descripción de los sensores que conforman el dispositivo Kinect v.2:

#### 2.4.1 Cámara Kinect v.2

La cámara que se encuentra en el dispositivo es RGB y tiene una resolución de 1920 x 1080 píxeles, logrando una calidad de imagen notablemente alta. Adicionalmente por el alta resolución y amplio tamaño de imagen se logra captar un cuadro mucho más grande en relación al logrado en el Kinect v.1.

### 2.4.2 Sensores infrarrojos Kinect v.2

Como se indica en el sitio web de Microsoft:

“Un emisor de infrarrojos (IR) y un sensor de profundidad IR. El emisor emite rayos de luz infrarroja y el sensor de profundidad lee los rayos IR reflejados de vuelta al sensor. Los haces reflejados se convierten en información de profundidad que mide la distancia entre un objeto y el sensor. Esto hace posible la captura de una imagen de profundidad” (Microsoft, 2009).

La diferencia entre los rayos IR del Kinect v.1 y v.2 es que su resolución aumentó a 512x424 píxeles, lo que permite tener una mucha mejor resolución en cuanto a profundidad se refiere.

### 2.4.3 Arreglo de micrófonos Kinect v.2

Al igual que en la primera versión se encuentra un arreglo de 4 micrófonos que se encuentran a lo largo del dispositivo para una mejor interacción.

Según una investigación para el manejo de robots industriales por medio de gestos de mano (Cueva, Torres, & Kern, 2018) se obtuvieron otras características relevantes que posee el dispositivo, mismas que se encuentran detalladas en la siguiente tabla:

*Tabla 1. Características Kinect V.2.*

<b>Características</b>	<b>Kinect V.2</b>
Cuadros por segundo	30
Distancia de profundidad máxima.	4,5 m
Distancia de profundidad mínima.	50 cm
Campo de visión horizontal.	70 grados
Campo de visión vertical.	60 grados

Motor tilt.	no
Articulaciones detectadas de Skeleton	26 articulaciones
Skeletons detectables.	6
Estándar USB.	3.0
Sistema operativo soportado.	Win 8/Win 10

Adaptado de Cueva, Torres, & Kern, 2018.

Muchas personas trataron de aprovechar sus características más allá de los videojuegos, llevándolo a inventos y nuevos desarrollos, pero para lograrlo los *pirateaban* para que puedan ser utilizados fuera de la consola; debido a esto Microsoft se vio obligado a entregar un Kit de Desarrollo para Kinect (SDK) a inicios de 2011 lo que fue una puerta de entrada para investigaciones, proyectos y demás utilizando este dispositivo (Microsoft, 2009).

## 2.5 Matemáticas aplicadas a reconstrucción 3D

### 2.6 Matrices de rotación

Una matriz de rotación o matriz rotacional es aquella que tiene una variación en cuanto a su orientación con respecto a un eje de coordenadas establecido, teniendo coincidencia en el origen con el mencionado sistema. En este caso se hablará de las matrices rotacionales básicas, ya que serán las aplicadas en este proyecto. En la Figura 15 a) se encuentran los dos sistemas coincidentes en el origen y en sus ejes, mientras que en la Figura 15 b) se encuentra los ejes OW y OV de los ejes establecidos OY y OZ, razón por la cual es evidente que el eje OU está rotando sobre el eje establecido OX, manteniendo la coincidencia en el origen.

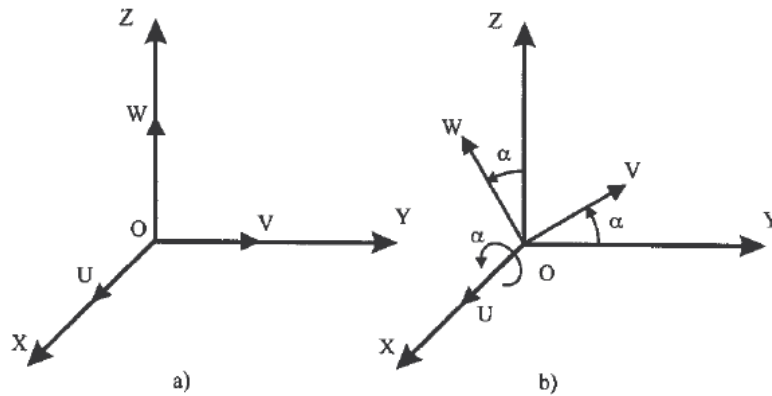


Figura 15 Explicación gráfica de lo que es una matriz de rotación sobre el eje X.

Tomado de Barrientos, 1996.

La explicación matemática de la imagen anterior se amplía a continuación (Barrientos, 1996):

$$R(x, \alpha) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\text{sen } \alpha \\ 0 & \text{sen } \alpha & \cos \alpha \end{pmatrix} \quad (\text{Ecuación 1})$$

De la matriz anterior podemos deducir que las matrices rotacionales de los ejes OV y OW sobre los correspondientes ejes OY y OZ son las siguientes (Barrientos, 1996):

$$R(y, \varphi) \begin{pmatrix} \cos \varphi & 0 & \text{sen } \varphi \\ 0 & 1 & 0 \\ -\text{sen } \varphi & 0 & \cos \varphi \end{pmatrix} \quad (\text{Ecuación 2})$$

$$R(z, \theta) \begin{pmatrix} \cos \theta & -\text{sen } \theta & 0 \\ \text{sen } \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{Ecuación 3})$$

Respectivamente, la demostración gráfica de las expresiones anteriores se encuentra en la Figura 16 a continuación:

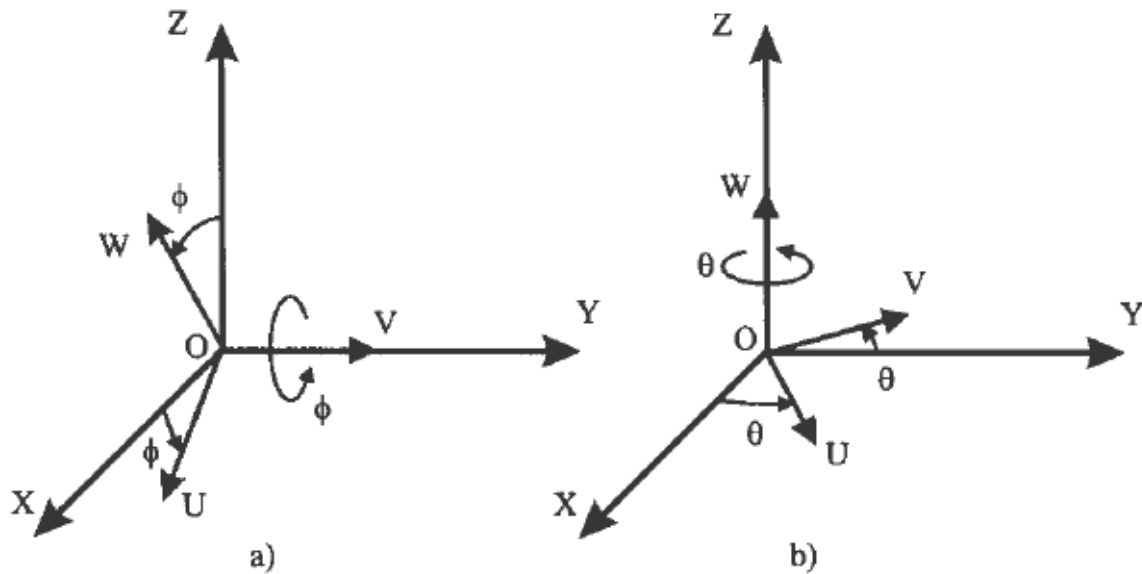


Figura 16. Explicación gráfica de los ejes de coordenadas a) OY y b) OZ.

Tomado de Barrientos, 1996.

## 2.7 Matrices de traslación

Una vez que se ha explicado el tema de rotación sobre un punto fijo, se procede a realizar la explicación de matrices de traslación, concepto que también es necesario para la obtención de matrices de transformación homogénea.

Como indica Barrientos, a una matriz que se encuentre trasladada con respecto a un eje de coordenadas establecido le corresponderá la siguiente matriz:

$$T(p) \begin{pmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{Ecuación 4})$$

La matriz mencionada anteriormente es conocida como *matriz básica de traslación*, ya que su concepto no presenta mayor complejidad; es necesario tener en cuenta que el concepto de esta matriz es respecto a un sistema de coordenadas establecido previamente.

## 2.8 Matrices de transformación homogénea

Según indica Barrientos en su libro *Fundamentos de robótica* capítulo tercero, una matriz de transformación homogénea se encuentra compuesta por cuatro diferentes matrices de la siguiente manera:

$$T = \begin{pmatrix} R_{3 \times 3} & P_{3 \times 1} \\ F_{1 \times 3} & W_{1 \times 1} \end{pmatrix} = \begin{pmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{pmatrix} \quad (\text{Ecuación 5})$$

La matriz de rotación tiene un tamaño de 3x3, mientras la matriz de traslación es de 3x1. Adicionalmente las matrices de perspectiva y escalado tienen un tamaño de 1x3 y 1x1 respectivamente, pero se tiene una particularidad, la matriz de perspectiva se ubicará como nula ya que el objeto se encuentra de frente con respecto al sensor; y la matriz de escalado se asignará un 1 ya que se tratará el tamaño real del objeto escaneado, es decir una escala de 1:1. De esta manera la matriz de transformación homogénea se establece de la siguiente manera (Barrientos, 1996):

$$T = \begin{pmatrix} R_{3 \times 3} & P_{3 \times 1} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \text{Rotación} & \text{Traslación} \\ 0 & 1 \end{pmatrix} \quad (\text{Ecuación 6})$$

En este punto es necesario realizar una aclaración, no es lo mismo rotación seguida de traslación y traslación seguido de rotación. Barrientos justifica lo anterior aclarando de la siguiente manera y se explica en la Figura 17:



“Se parte de un sistema OUVW coincidente con OXYZ al que se va a aplicar una traslación según un vector  $p_{xyz}$  y una rotación de  $180^\circ$  alrededor del eje OZ. Si primero se rota y después se traslada se obtiene otro sistema final O''U''V''W'', que representa una localización totalmente distinta a la del sistema final anterior. Se tendrá, por tanto, matrices homogéneas distintas según se realice una traslación seguida de rotación o una rotación seguida de traslación.” (Barrientos, 1996)

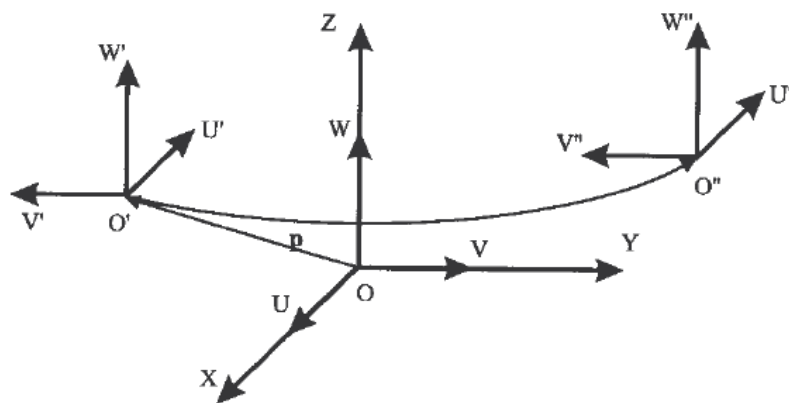


Figura 17. Distintos sistemas finales según el orden de las transformaciones.

Tomado de Barrientos, 1996.

## 2.9 Software aplicado a reconstrucción 3D

### 2.10 Datos obtenidos con información de varios sensores

Mediante la mezcla de los datos recibidos de cada sensor el dispositivo Kinect puede realizar varios *productos*, entre ellos se puede mencionar los siguientes:

#### 2.10.1 Skeletons

Con este término se conoce al diagrama de articulaciones que realiza el dispositivo Kinect, como se indica en el desarrollo de Pox y Wallhoff el sistema captura información en forma de posiciones corporales tridimensionales, estas

posiciones son transformadas en articulaciones basado en una máquina de vectores de soporte (*SVM* por sus siglas en inglés – *Support Vector Machine*) (Vox & Wallhoff, 2017). En la segunda versión del dispositivo el mismo puede reconocer el diagrama de hasta seis personas simultáneamente, cada skeleton compuesto por 26 articulaciones como se puede observar en la Figura 18, entre las que podemos destacar muñecas, cuello, hombros, codos, rodillas y tobillos.

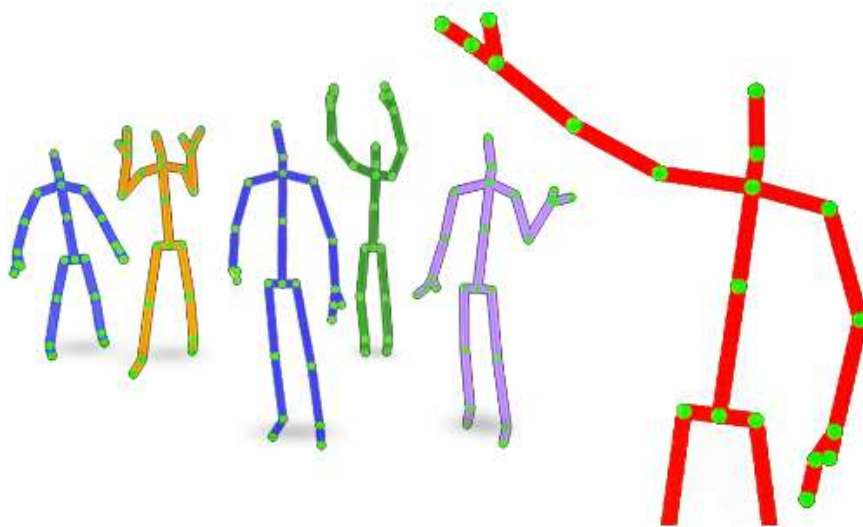


Figura 18. Muestra de captura de skeletons.

Tomado de Microsoft, 2018.

### 2.10.2 Medidor de pulso cardíaco

Uno de los usos más innovadores que se ha dado a este dispositivo de Microsoft está el medidor de pulso cardíaco como se realizó en la investigación de Gambi y otros, en el cual se menciona la manera en la cual se puede realizar una medición de pulso cardíaco sin necesidad de contacto añadiendo ventajas económicas y tecnológicas versus tipos de métodos de mediciones. Este proceso se lleva a cabo con la ayuda del *EVM* (*Eulerian Video Magnification*), *PPG* (*Photoplethysmography*) y *VPG* (*Videoplethysmography*)

con el fin de alcanzar un desempeño comparable a las tecnologías clásicas o de contacto (Gambi, y otros, 2017).

### 2.11 Point Cloud

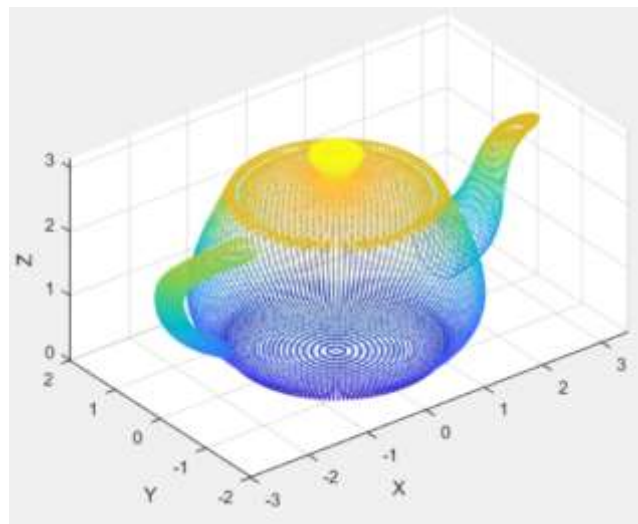
Para capturar información de una escena tridimensional los distintos dispositivos utilizan haces de luz infrarroja, mismos que son el conjunto de varias luces, desde sus emisores para realizar escaneos de profundidad de las escenas de interés, al regresar estos haces de luz al receptor infrarrojo se obtiene una distancia estimada del dispositivo al lugar contra el que chocó dicho haz de luz infrarroja, este principio se replica para todos los haces que son enviados por el emisor, de esta manera se captura una cantidad de información importante, a esta se le denomina *nube de puntos* o *point cloud (PC)*, nube que debe ser entendida correctamente y procesada con el fin de obtener detalles de la escena capturada (Calvo Salcedo, Bejarano Martínez, & Quintero Salazar, 2012). Existen una gran variedad de métodos para interpretar esta información y desplegarlos de una manera entendible, entre ellos se encuentran mayormente programas de computación ya que el nivel de procesamiento necesario para esta información es alta. Entre algunos nombres se tiene Matlab, MeshLab, LabView, entre otros. En este caso y debido a las librerías existentes para manipulación y compatibilidad con el dispositivo Kinect v.2 prestadas por el software se utilizará Matlab. Es importante mencionar que la nube de puntos utilizada en esta ocasión no será a color, debido a que las funciones utilizadas no permiten tal característica.

### 2.12 Procesamiento de imágenes 3D

En la actualidad existe una gran variedad de software que ayuda a reconstrucciones, entre ellos se encuentran Microsoft Visual Studio, Microsoft Kinect Studio, Matlab, etc. En esta ocasión se utilizará este último debido a que

Matlab posee el Computer Vision System Toolbox el cual facilita algoritmos para reducción de resolución, eliminación de ruido y transformación de nubes de puntos. A la vez provee de registro de PC, ajuste de forma geométrica a PC 3D y la habilidad de leer, escribir, almacenar, desplegar y comparar PC. Finalmente se puede combinar múltiples PCs para reconstruir escenas 3D.

Entre las funciones más destacadas se encuentra `pointCloud class`, cuya función principal es almacenar las PCs 3D. Cada punto de la PC tiene coordenadas xyz, las mismas que se pueden almacenar en una matriz de  $M \times 3$  o de  $M \times N \times 3$ , a esta última comúnmente se la conoce como PC organizada. Los valores de z, que regularmente corresponden a profundidad o elevación, determinan el color de cada punto. Los puntos xyz de entrada deben ser numéricos (Matlab, 2015). En la Figura 19 se tiene un ejemplo de una nube de puntos.



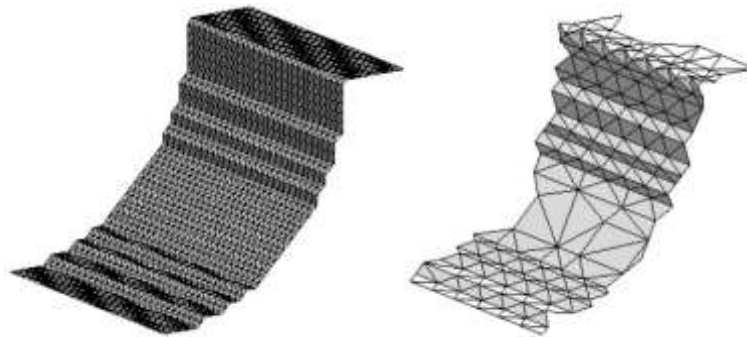
*Figura 19.* Reconstrucción de una tetera.

Tomado de Matlab, 2015.

### 2.13 Reconstrucción en tres dimensiones

Una reconstrucción tridimensional es el procedimiento mediante el cual objetos reales son reproducidos en una computadora o dispositivo manteniendo sus dimensiones, volumen y forma (Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada, s.f.).

Existen una gran variedad de métodos de reconstrucción y mallado tridimensional que tienen como objetivo aplicar un algoritmo capaz de conectar de manera correcta los puntos más representativos en formas geométricas, triángulos, cuadrados, entre otras como se puede observar en la Figura 20. Entre los temas a considerar para reconstrucción y mallado se encuentran el coste computacional y la calidad de imagen que se pretende tener, mientras mayor sea la calidad mayor será el coste computacional.



*Figura 20.* Ejemplos de mallado.

Tomado de Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada, s.f.

Según la investigación de reconstrucción 3D de Forero, se encuentran varios métodos geométricos de reconstrucción, entre ellos se encuentran los siguientes:

- Triangulación de superficies por medio de contornos paralelos.

- Aproximaciones de estructuras poliedrales por medio de la tetrahedrización del volumen.
- Diagramas de Voronoi.
- Triangulación de Delaunay.
- Modelamiento geométrico por medio de funciones matemáticas.

Cada método de los anteriores tiene sus propios principios y es aplicable para todo tipo de reconstrucciones, obviamente la selección del método idóneo para la reconstrucción depende del nivel de conocimiento en el manejo de cada método, tecnología con la que se realiza la captura de información y objeto o entorno escaneado (Forero, Aranzazu Buitrago, & Flórez Larrahondo, 2001).

### 3. Metodología

Para elaborar el presente proyecto de titulación se utiliza una metodología analítica experimental, pues el trabajo propone analizar y diseñar los procedimientos para la reconstrucción digital de un objeto en tres dimensiones con la ayuda de un sensor Kinect v.2 y software dedicado al tema. Además de esta metodología, se utiliza como método de desarrollo científico una estrategia deductiva, pues existen algunos desarrollos previos desde los que se enfoca al tema de estudio propuesto.

Los materiales necesarios son:

- Dispositivo Kinect v.2.
- Motor a pasos.
- Microcontrolador.
- Computador personal.
- Software Matlab R2017b.
- Cable adaptador de conexión Kinect – PC

- Plataforma en forma de L.
- Objeto a reconstruir.

Para medir la calidad del producto se realizarán pruebas con objetos de diferentes dimensiones y formas.

### 3.1 Conexión con el microcontrolador

El mecanismo de rotación se encuentra compuesto por un microcontrolador, un motor a pasos, un *driver* para dicho motor y un cable de conexión con el computador personal. Dicha conexión tiene como objeto enviar una señal desde el computador hacia el motor para que realice los giros de 90° exactos, para que se sincronice con los escaneos realizados por el dispositivo Kinect, manteniendo así un orden y evitando desfases entre las imágenes capturadas.

El microcontrolador utilizado en esta ocasión fue el 328p, mismo que se encuentra en una placa de programación (Arduino). El código realizado en lenguaje Arduino se puede observar en la sección de Anexos al final del presente documento.

### 3.2 Software y librería

En primera instancia se debe instalar un plug-in sobre el programa Matlab que lleva por nombre *Kinect 2 Interface for Matlab*, disponible para descarga gratuita en la pestaña Add-ons. Este paquete contiene funciones y herramientas que son de gran ayuda para procesar y manejar los datos adquiridos por los sensores que se encuentran en el interior del Kinect, en la Figura 21 se encuentra una captura de la documentación de la librería utilizada.



The screenshot shows the MATLAB File Exchange page for the 'Kinect 2 Interface for Matlab' package. The package is version 3.2.2 (1.5 MB) by Juan R. Terven, part of the Kin2 Toolbox. It has 50 ratings (all 5 stars) and 79 downloads, updated on 06 Jul 2017. The page includes an 'Overview' tab and detailed documentation. The documentation is written in C++ and uses the Microsoft Kinect 2 SDK. It lists requirements: Kinect2 SDK, Visual Studio 2012 or newer compiler, MATLAB 2013a or newer, and MATLAB 2015b or newer for pointCloudDemo2. Usage instructions include setting the compiler, opening compile\_cpp\_files, adding the Kinect20 Fusion.dll and Kinect20.Face.dll to the Windows path, and running compile\_cpp\_files.m. Demos include videoDemo.m, mappingDemo.m, and mapping2CamDemo.m. The 'Requires' section lists MATLAB and Kinect 2 SDK. The 'Tags' section includes 'kinect2' and 'kinect'. The 'Acknowledgements' section mentions inspiration from an Example MATLAB class wrapper and MATLAB AND SIMULINK.

Figura 21. Documentación Kinect 2 Interface for Matlab.

Tomado de Terven, 2017.

### 3.3 Aplicación de Matlab para captura de datos

Una vez terminada la instalación se inicia el programa de escritorio Matlab y se conecta el dispositivo Kinect al computador mediante el adaptador.

Cuando el dispositivo haya sido reconocido por el computador se procede a la elaboración de *scripts* mismos que interactúan con el dispositivo, mediante ellos se inicializan los sensores y se envían distintos comandos para que el Kinect realice las funciones que se necesiten. En la Figura 22 se encuentra un diagrama del proceso que se ejecuta.



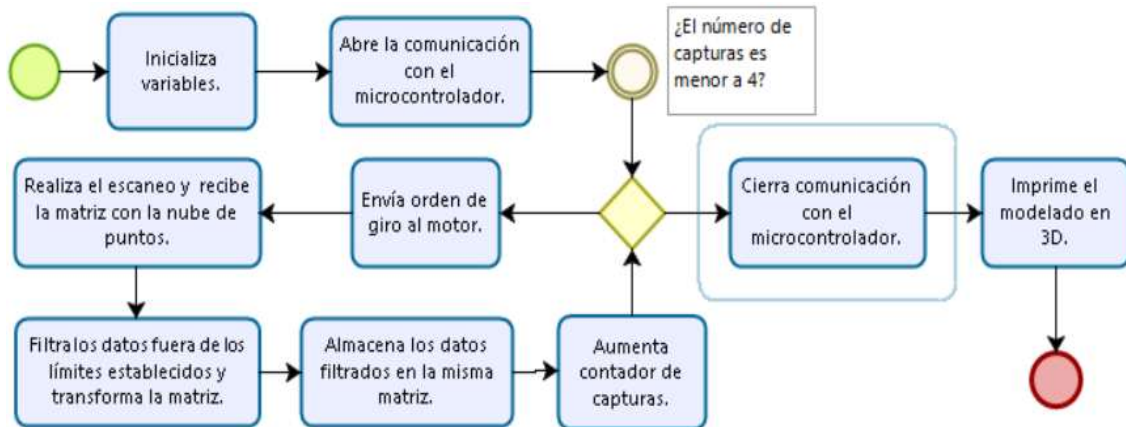


Figura 22. Diagrama general del proceso de reconstrucción.

### 3.4 Manejo de los datos capturados

Como se pudo observar en la explicación anterior de toda la información que provee el dispositivo Kinect v.2, se utiliza únicamente la función *Location*, misma que entrega las coordenadas de cada punto de la nube capturada, dichas coordenadas se encuentran dentro de una matriz de dimensiones  $M \times 3$ . Donde  $M$  representa el número de filas y 3 las coordenadas de puntos en  $x$ ,  $y$ ,  $z$  en ese orden. La dimensión de  $M$  es variable pues depende de la cantidad de puntos que encuentre el dispositivo Kinect en su campo de visión en el escaneo.

Una vez que se ha obtenido la matriz de puntos de la nube de puntos es posible manipularla, con el fin de rotarlas y trasladarlas para que se acoplen con las nubes de puntos siguientes producto de los distintos escaneos que serán realizados. Para realizar la rotación y traslación requeridas de las matrices en cuestión se aplican los argumentos matemáticos indicados previamente en el marco teórico, a través de los cuales se puede manejar dicha información según las necesidades que se presenten.

Antes de desarrollar las matrices de rotación se debe considerar que el objeto a escanear rotará sobre la plataforma que gira sobre su propio eje, por lo que se utiliza el caso de rotación seguido de traslación. Tomando en cuenta que el sistema de coordenadas del dispositivo Kinect v.2 es de la manera en que se muestra en la Figura 23, únicamente se utilizará rotación sobre el eje  $y$ .

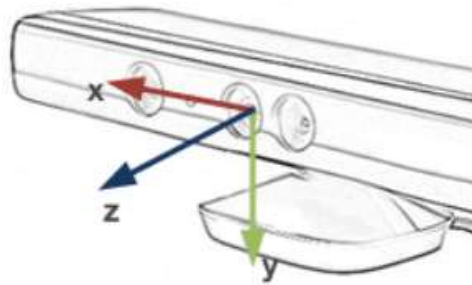


Figura 23 Ejes de coordenadas del dispositivo Kinect.

Tomado de Doc-Ok.org, 2013.

$$T((y, \varphi), p) = \begin{pmatrix} \cos \varphi & 0 & \text{sen } \varphi & px \\ 0 & 1 & 0 & py \\ -\text{sen } \varphi & 0 & \cos \varphi & pz \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{Ecuación 7})$$

Adicionalmente es importante recalcar que la única distancia que existe es la que se encuentra entre el sensor y el objeto. Después de la puntualización anterior la matriz transformada homogénea es la siguiente:

$$T((y, \varphi), p) = \begin{pmatrix} \cos \varphi & 0 & \text{sen } \varphi & 0 \\ 0 & 1 & 0 & py \\ -\text{sen } \varphi & 0 & \cos \varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{Ecuación 8})$$

Finalmente, el ángulo  $\varphi$  que se menciona en las matrices es el ángulo de rotación que realizará el objeto entre cada toma; el mismo será variable y su valor dependerá del número de tomas que se realicen de los objetos.

Para capturar y procesar la información se programaron los siguientes scripts:

- a. `escaneoKinect`: En esta función se encuentran los comandos básicos para realizar un escaneo mediante el dispositivo Kinect v.2, entre ellos: iniciar el sensor de profundidad, enviar la nube de puntos a la función principal, liberar los sensores y capturar en una matriz la nube de puntos.
- b. `TransformaMatrices`: En esta función se realizan los cálculos matemáticos, donde se aplican los criterios de matrices de transformación homogénea. Como resultado de esta función se tiene las nubes de puntos rotadas un ángulo establecido y trasladadas una distancia determinada. Es importante resaltar que la matriz inicial es de dimensiones  $M \times 3$  y que luego de este proceso matemático de trasposición resulta una matriz de  $4 \times M$ , donde la fila excedente está formada por únicamente por números uno (1).
- c. `LimpiaNube`: Realiza un filtro en el que se retiran puntos de la nube que se encuentren fuera de los límites establecidos.
- d. `abrePuerto`: Envía los comandos para que el puerto USB del computador se abra a la comunicación con el microcontrolador.
- e. `enviaOrdenGiro`: Es el encargado que enviar la señal al arduino Nano para rotar el motor a pasos  $90^\circ$  exactos.

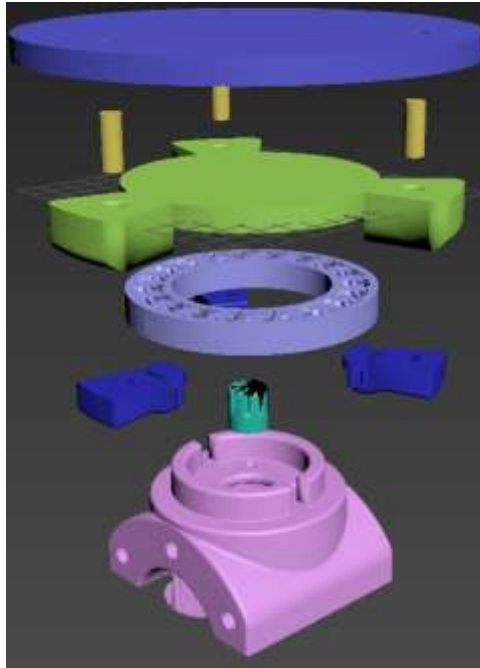
- f. `cierraPuerto`: Envía los comandos para cerrar el puerto por el cual se estableció la comunicación.
- g. `TomaMatricesDeKinect`: Es la función principal del programa desde la cual se llama a las funciones explicadas anteriormente y se juntan de tal manera que los resultados sean los mejores; adicionalmente se declara las variables principales que ayudan a controlar el flujo del programa.

El código de programación de cada script se encuentra en Anexos, al final del presente documento.

Adicional al tema de software se diseñó una base para el motor a pasos para mover la plataforma sobre la que se va a posar el objeto a ser reconstruido y una plataforma sobre el cual se va a montar el sistema. La base fue construida en una impresora 3D, el modelo puede observarse en la Figura 24, sobre esta base se posará un rulimán que ayudará a girar la plataforma que debe posarse sobre sí, el sistema completo de rotación con todas sus partes puede observarse en la Figura 25.



*Figura 24.* Modelo 3D de la base del motor.



*Figura 25.* Modelo de todo el sistema del motor para girar la plataforma.

La parte superior del sistema de la Figura 25 está comprendido por el rulimán y la parte alta de la plataforma, esto ayuda para los giros que se realicen, el mecanismo se puede observar de mejor manera en la Figura 26.



*Figura 26.* Sistema de rotación.

Todo este sistema debe posarse sobre una plataforma con un sistema de rieles. Las medidas de la misma son: 0.12 m de ancho por 1.00 m de longitud. En los extremos se encuentran piezas de plástico para mantener las varillas de metal a la misma distancia. Adicionalmente se utiliza un trípode para mantener fijo el dispositivo Kinect v.2, Figura 27, con la ventaja de poder variar su altura a la cual se encuentre el sistema de rieles para tomar las mejores capturas posibles.



*Figura 27.* Trípode con el cual se sostiene el dispositivo Kinect v.2.

Una vez que se encuentran listos todos los materiales se procede a armar el sistema, en el que se utiliza el computador donde corre el programa Matlab

2017 para interpretar la información receptada por el dispositivo Kinect, quedando como resultado la Figura 29:



*Figura 28.* Fotografía del sistema de rieles y computador.

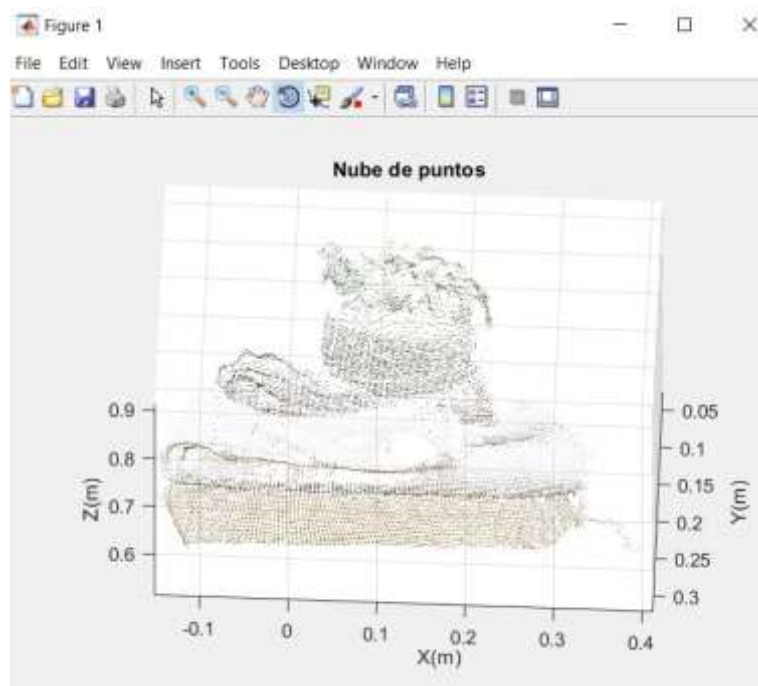
#### **4. Resultados y discusión**

Inicialmente para verificar que el escaneo realizado por el dispositivo Kinect se ajuste a lo requerido se ejecuta únicamente la función `escaneoKinect` cuyos comandos realizan acciones básicas como inicializar sensores y desplegar la nube de puntos capturada. Es importante mencionar que el dispositivo Kinect únicamente capta la información de los objetos con los que tiene una línea de vista directa. En primera instancia se procede a escanear el siguiente florero sobre una superficie de tela, con la intención de observar la calidad de la imagen con respecto a los distintos materiales, el objeto se aprecia en la Figura 29.



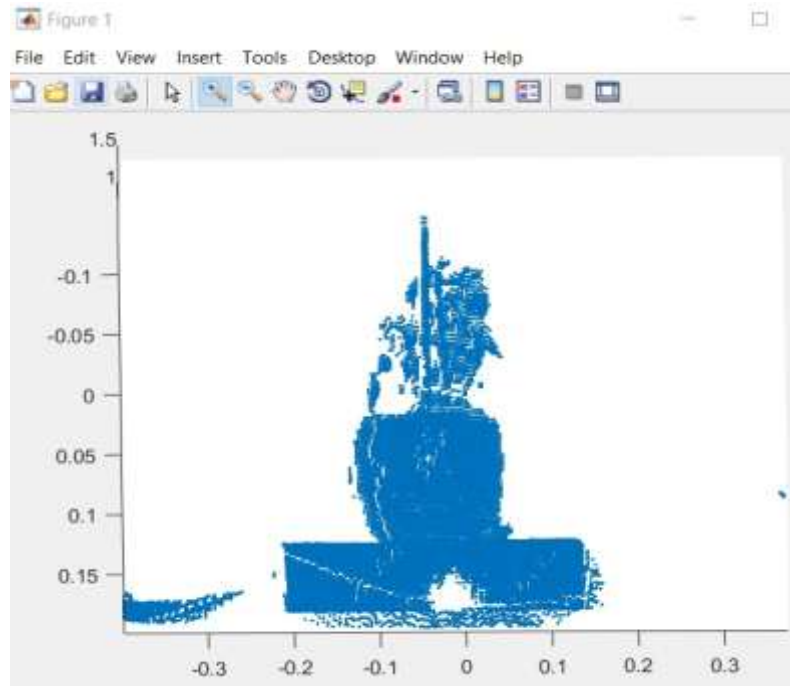
*Figura 29.* Fotografía del objeto de prueba para comprobar escaneo y encuadre.

Luego de realizar el escaneo mencionado se logra un encuadre del objeto de interés se tiene como resultado la Figura 30, desplegando en una imagen la nube de puntos del mismo objeto se obtiene lo mostrado en la Figura 31.



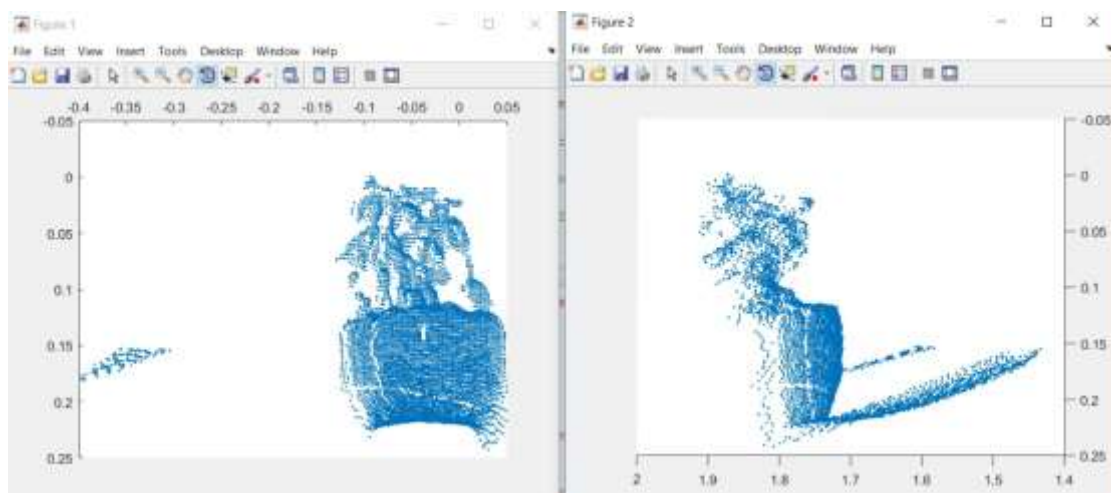
*Figura 30.* Imagen enfocada del objeto de interés.





*Figura 31.* Nube de puntos del objeto de interés.

Posteriormente, se realiza una prueba para evidenciar el correcto funcionamiento del giro de las imágenes basado en el criterio matemático de matrices transformadas homogéneas, obteniendo como resultado la imagen de la Figura 32, el giro para esta prueba fue de  $90^\circ$ .



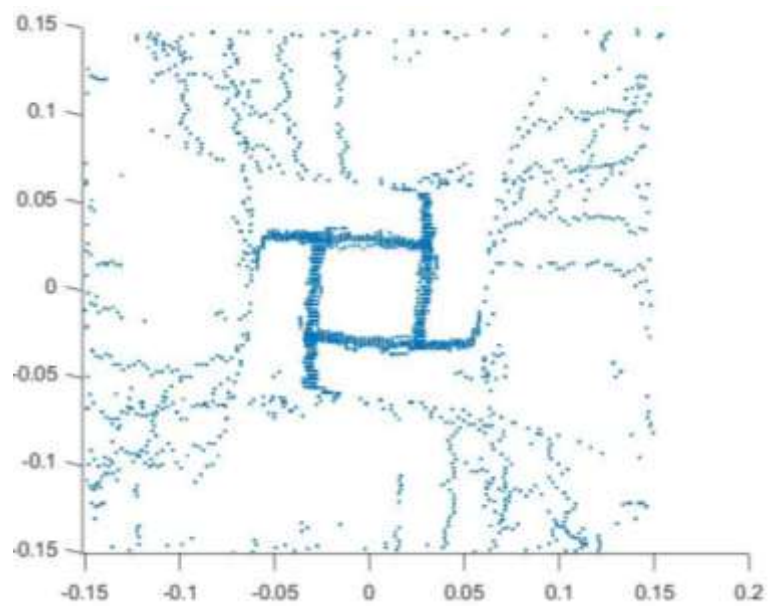
*Figura 32.* Evidencia de funcionamiento de matrices transformadas homogéneas.

Para probar el funcionamiento correcto del sistema se colocó un objeto cúbico en la plataforma para escanear y se ubica el Kinect a una altura aproximada de 0.2 m y a unos 0.90 m del objeto como se observa en la Figura 33. La caja utilizada es de dimensiones 0.10 m x 0.10 m. x 0.15 m., mismas que deben ser visibles en los resultados digitalizados del objeto.



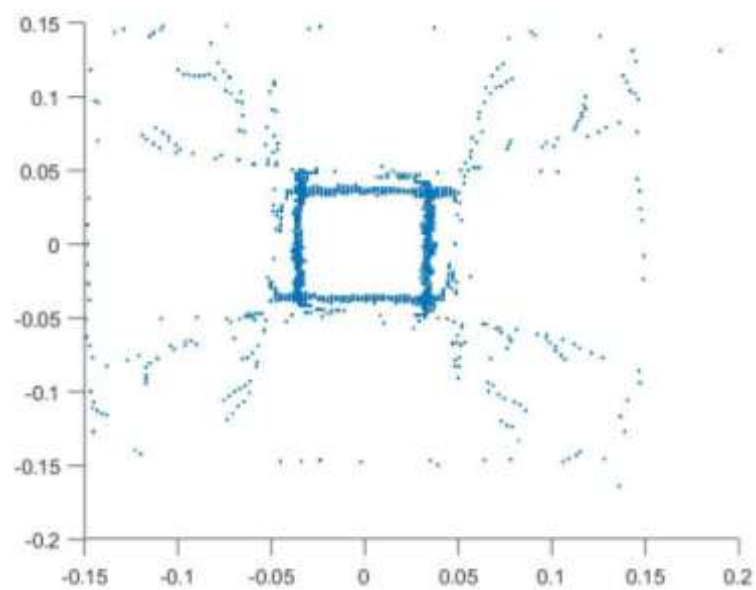
*Figura 33.* Muestra de plataforma lista para escaneo.

En el primer escaneo se produjo un inconveniente en la reconstrucción, pues en la toma superior de la imagen obtenida no fue precisa, la unión de las cuatro nubes de puntos se encontró desfasada como se observa en la Figura 34, esto se debe a que el objeto no se encuentra exactamente alineado con la cámara del sensor de profundidad. Otra causa es que la distancia entre el sensor y el objeto no es exactamente 0.90 m., por lo que es necesario que las distancias sean exactas.



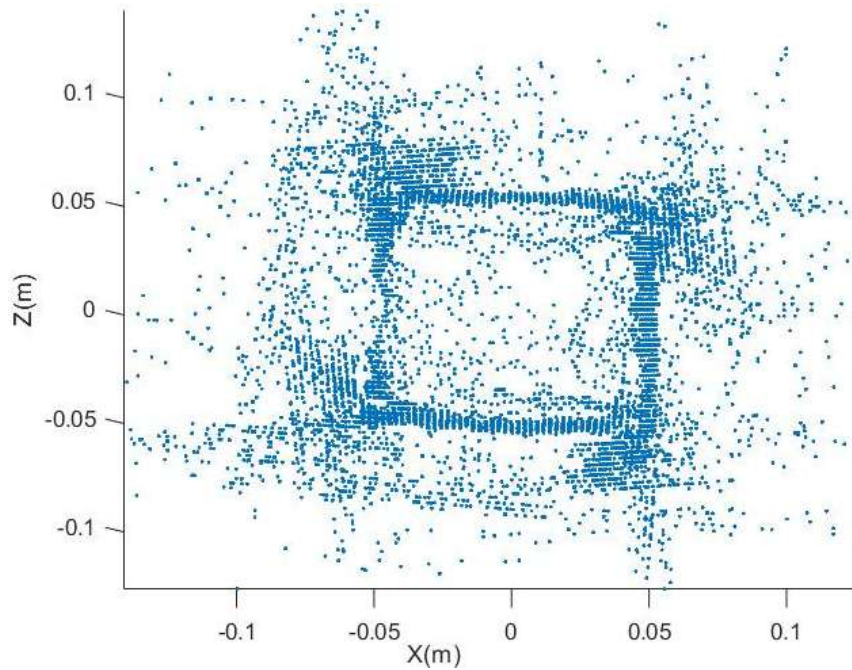
*Figura 34.* Toma superior de la unión de las capturas desfasadas.

Una vez que se mitigó el alineamiento del objeto con el dispositivo y la distancia entre ellos, se volvió a ejecutar el escaneo encontrando que la toma superior tuvo una mejora importante, como se puede observar en la Figura 35.



*Figura 35.* Toma superior mejorada de imagen capturada.

Después de realizar un ajuste más minucioso, se pudo obtener un resultado con una notable mejora de calidad, como se observa en las Figuras 36, 37 y 38:



*Figura 36.* Vista superior de la caja escaneada.

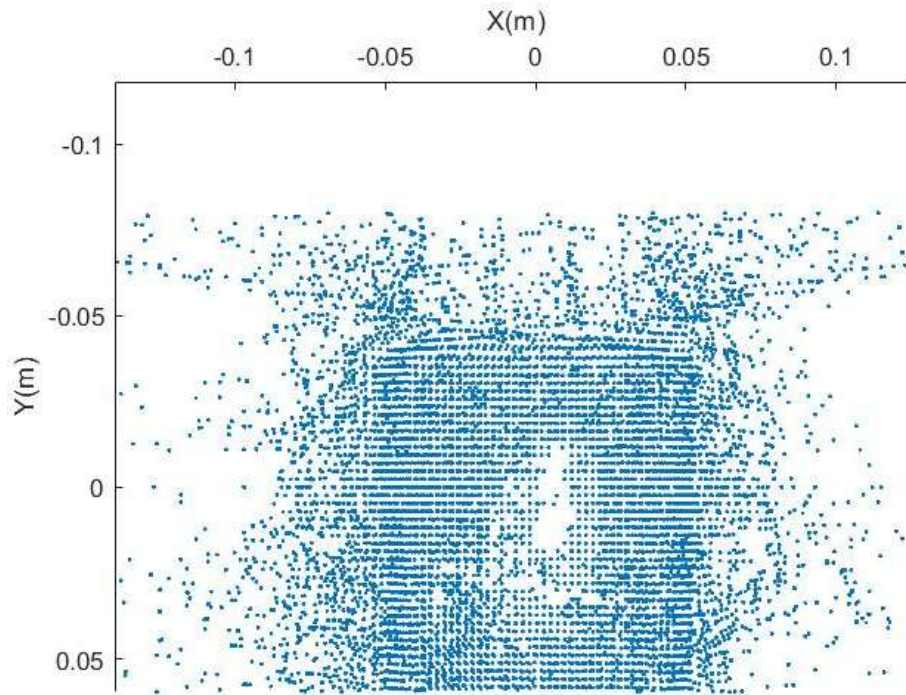


Figura 37. Vista lateral de la caja escaneada.

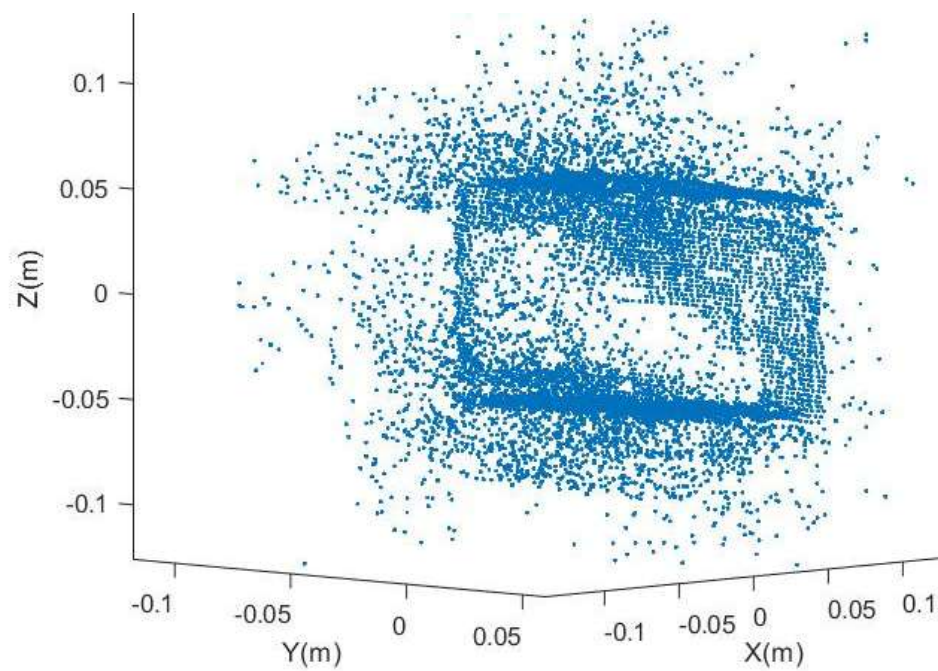


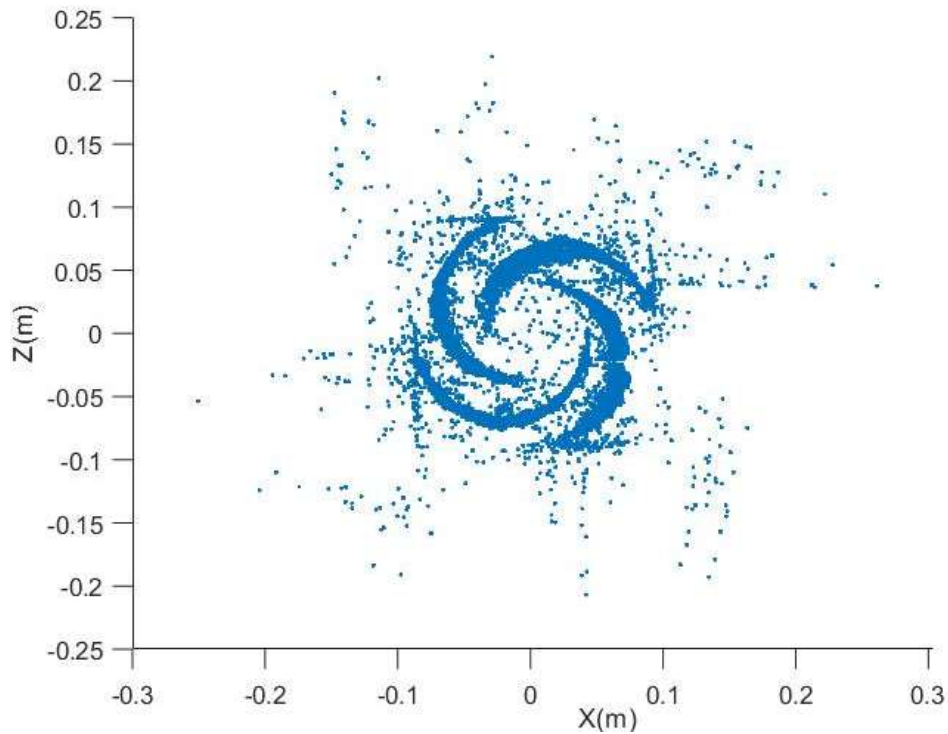
Figura 38. Vista diagonal de la caja escaneada.

Posteriormente, se ubicó un objeto cilíndrico de diámetro 0.10 m, como se muestra en la Figura 39, de la misma manera se alinea exactamente con el sensor de profundidad y se mantiene la distancia preestablecida.



*Figura 39.* Cilindro utilizado para pruebas de escaneo.

En las pruebas con este objeto se encontró un desfase de aproximadamente 0.25 m, por lo que las nubes de puntos de las cuatro capturas no convergieron correctamente, como se aprecia en la Figura 40.



*Figura 40.* Toma superior de la imagen del cilindro desfasada.

Luego de calibrar la distancia del dispositivo Kinect y de la alineación del mismo, se encontró que el error se debía a mediciones, tema que fue mitigado con la ayuda de un flexómetro, corrigiendo los 0.25 m de error, aproximadamente. Como resultado pudo obtener una imagen notablemente mejor y que se aprecia de una mejor manera como se observa en las Figuras 41, 42 y 43:

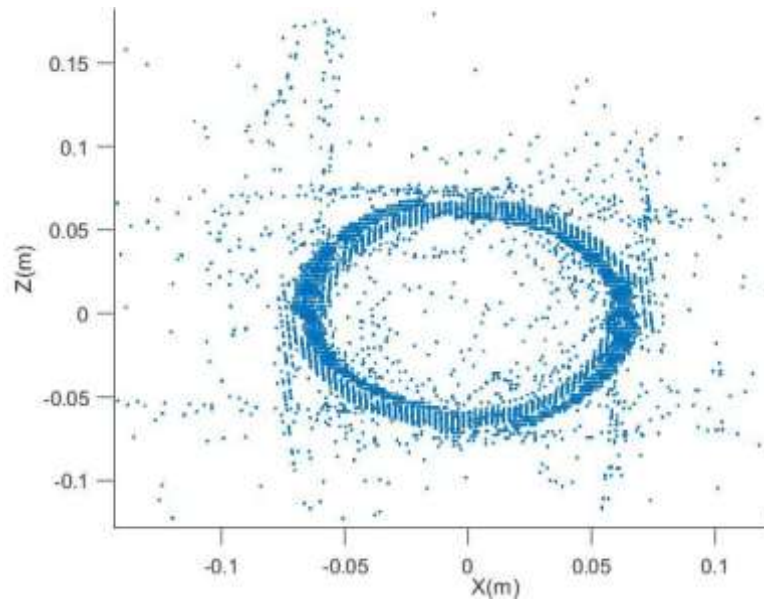


Figura 41. Vista superior del cilindro escaneado.

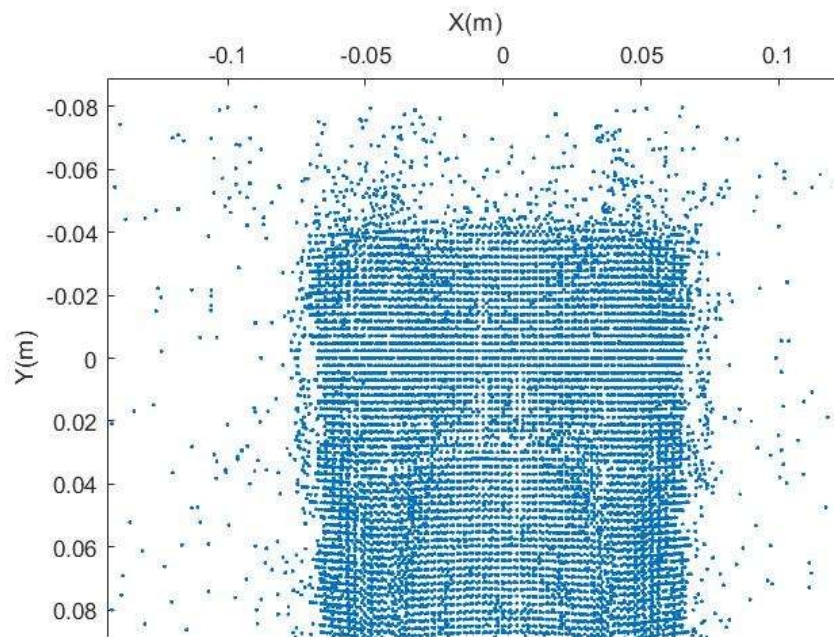
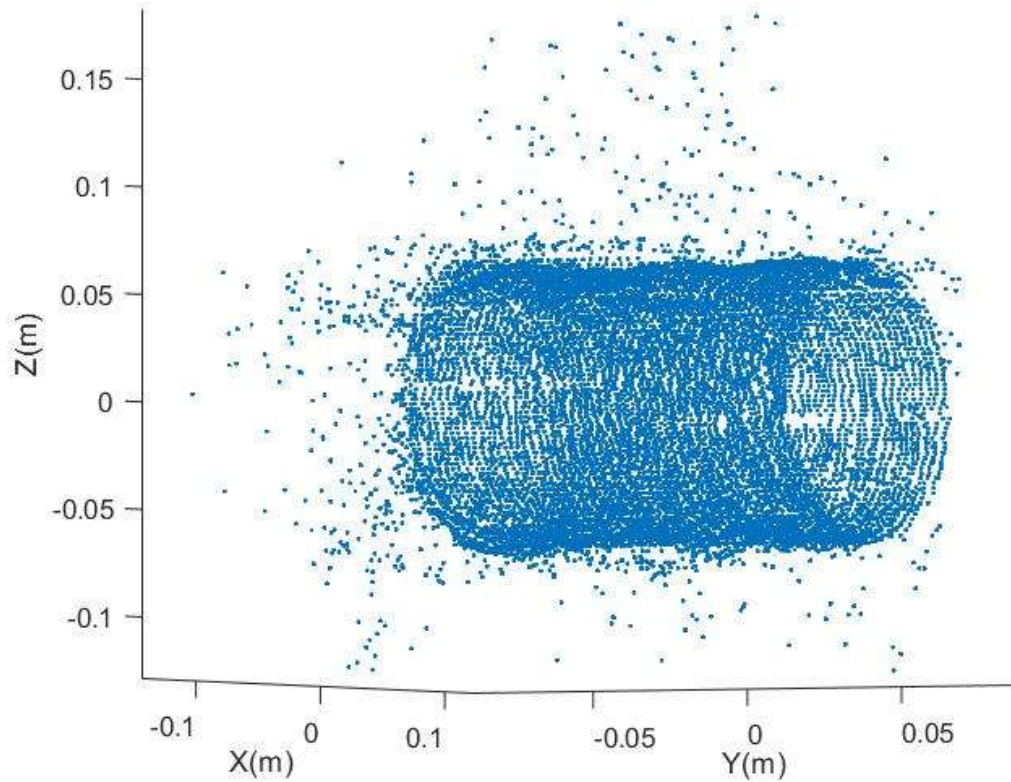


Figura 42. Vista lateral del cilindro escaneado.





*Figura 43.* Vista diagonal del cilindro escaneado.

Por último, se realizó el escaneo de una figura asimétrica e irregular entre sus distintas partes, un robot de juguete de cuatro patas, tal como se observa en la Figura 44 a continuación:



Figura 44. Robot de juguete a escanear.

Al ya tener identificado la distancia y la alineación con el sensor de profundidad se procedió a obtener resultados sobresalientes como se puede observar en las Figuras 45, 46 y 47:

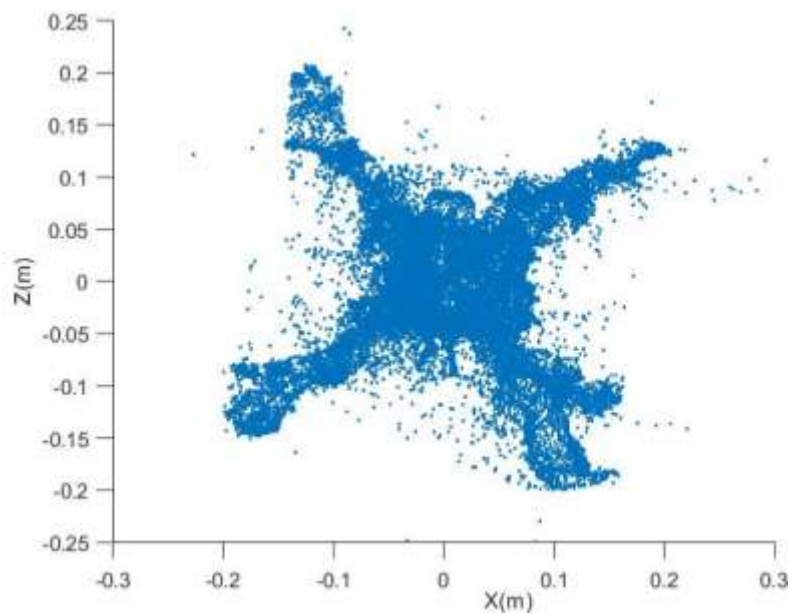


Figura 45. Vista superior de robot de juguete escaneado.

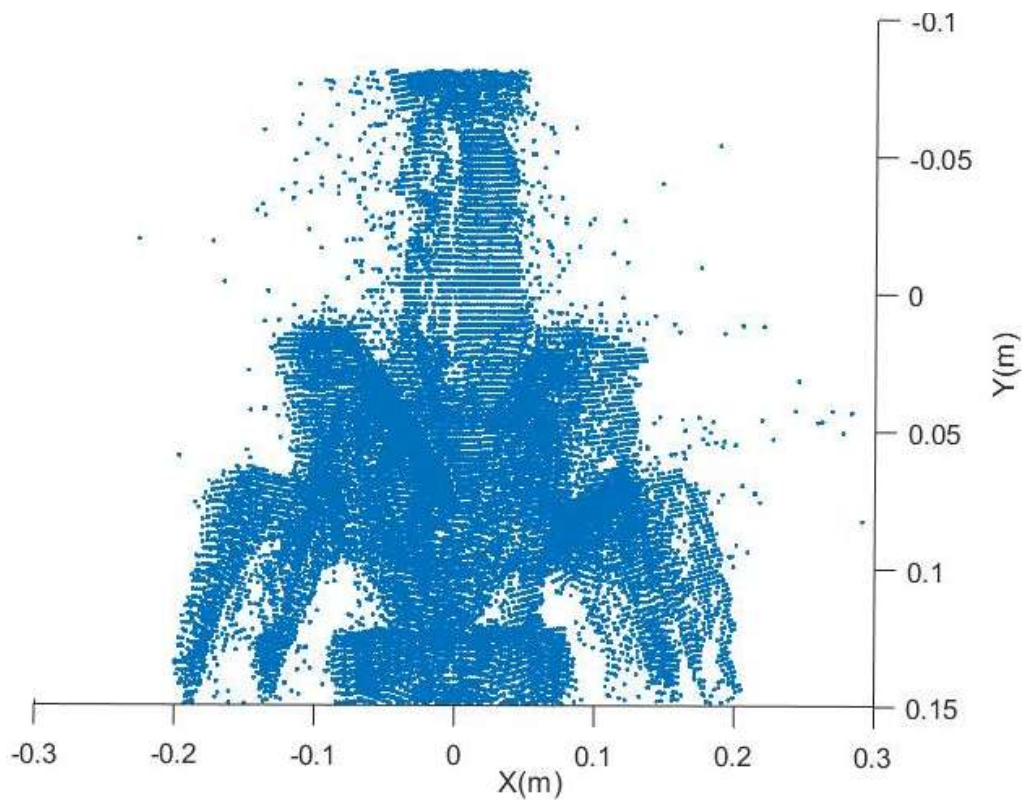


Figura 46. Vista lateral del robot de juguete escaneado.

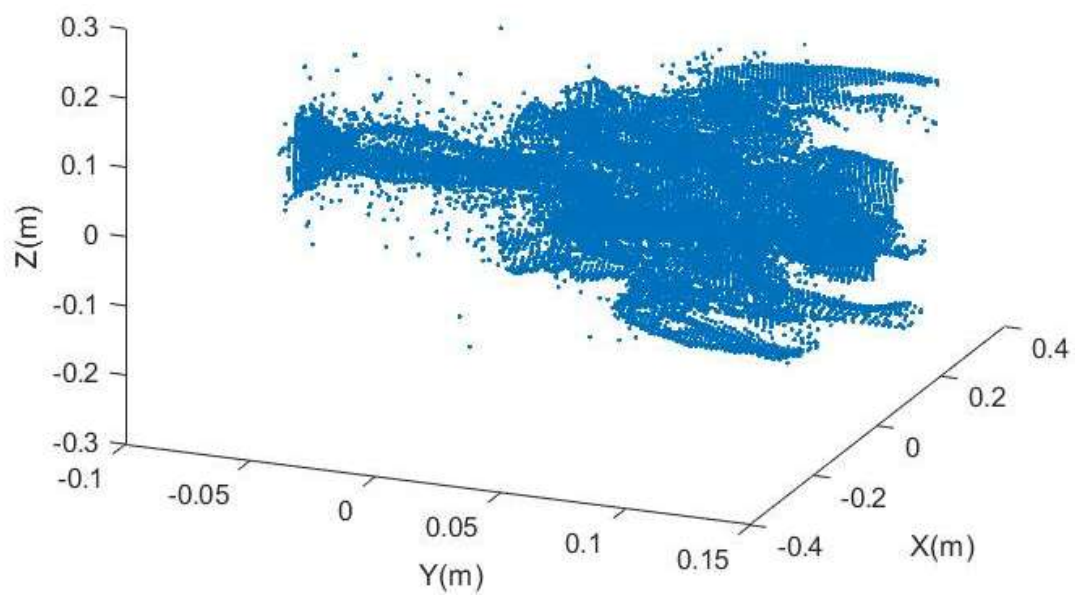


Figura 47. Vista diagonal del robot de juguete escaneado.

## 5. Conclusiones y recomendaciones

### 5.1 Conclusiones

El dispositivo Kinect posee una resolución muy alta en cuanto a los detalles en centímetros que se es capaz de detectar, en cuanto a los rayos infrarrojos, se pueden escanear objetos pequeños, no obstante, los objetos transparentes no es posible escanearlos debido a que los rayos infrarrojos los atraviesan y no retornan al receptor. Este fenómeno se produce incluso si existe exceso de luz, ya que por ende hay reflejo de la misma en el objeto se obtendrán espacios vacíos en la nube de puntos.

Las funciones de la librería de Matlab Kinect 2 Interface for Matlab son únicamente de lectura (read-only), razón por la cual no es posible modificar las nubes de puntos con todas las variables que engloban. La función Location es una de esas variables y al almacenarla en una matriz la vuelve manipulable según la necesidad que se presente.

Con la realización de cuatro escaneos se obtienen resultados de una calidad notable, ya que se demostró con las pruebas que el número de escaneos mencionado se cubre aproximadamente un 90% de la superficie de los objetos.

### 5.2 Recomendaciones

Es importante realizar un filtrado correcto, ya que de esta manera se depuran objetos externos que no son de interés.

Un error recurrente fue no alinear el centro del objeto con el sensor de profundidad, la convergencia de las nubes de puntos puede no ser la correcta, debido a que en los cálculos de las matrices de transformación no se encuentra compensada esta variación. En caso de que no exista manera de alinearlos, se recomienda compensar las distancias (del sensor al centro del objeto y la alineación del sensor con el centro del objeto) en las respectivas matrices matemáticas para que con los cálculos pueda lograrse la convergencia de las nubes de puntos.

Es recomendable que las pruebas iniciales se realicen con objetos simétricos para confirmar que el centro del mismo se encuentre alineado con la cámara del sensor.

Tomar en cuenta que los límites del encuadre del objeto no sean menores que las dimensiones de la plataforma, pues se puede incurrir en una pérdida de información.

Verificar que los ángulos de rotación sean exactos, pues el cálculo matemático está basado en ángulos precisos para superponer una nube de puntos con otra.

## Referencias

- Abdullah, Zabidi, Yassin, & Hassan. (2015). Analysis of Microsoft Kinect Depth Perception for Distance Detection of Vehicles. IEEE. Recuperado el 20 de abril de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>
- Akhil, P., & Parimi, A. (2016). Synchronization of Motion From Multiple Humans and Humanoid Robot based on Kinect V2. IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>
- ASUS. (2018). Imagen de Xtion PRO LIVE. Recuperado de [https://www.asus.com/3D-Sensor/Xtion\\_PRO\\_LIVE/](https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/)
- Atman, J., & Trommer, F. G. (s.f.). Laser-Camera Based 3D Reconstruction of Indoor Environments. IEEE, 255-260. Recuperado el 5 de abril de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Barrientos, A. (1996). Fundamentos de robótica. McGraw-Hill. España.
- Benli, E., Rahlf, J., & Motai, Y. (2018). Dynamic 3D Reconstruction of Human Targets via an Omni-Directional Thermal Sensor. IEEE. Recuperado el 25 de abril 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Calvo Salcedo, A. F., Bejarano Martínez, A., & Quintero Salazar, E. A. (2012). Procesamiento de nubes de puntos por medio de la librería PCL. IEEE. Recuperado el 9 de abril de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Chen, C., Chen, L., Zhou, X., & Yan, W. (2017). Controlling a Robot Using Leap Motion. IEEE. Recuperado el 2 de abril de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Çubukçu, B., & Yüzgeç, U. (2017). A Physiotherapy Application with MS Kinect for Patients with Shoulder Joint. IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Cueva, W., Torres, H., & Kern, J. (2018). 7 DOF Industrial Robot Controlled by Hand Gestures. Automatic Control (CCAC), 2017 IEEE 3rd Colombian Conference on (pág. 6). Cartagena: IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada. (s.f.). Recuperado el 15 de mayo de 2018 de Ingeniería y diseño industrial de [http://www.elai.upm.es/webantigua/spain/Investiga/GCII/personal/lrodriguez/web3D/reconstruccion\\_3d.htm#Introducci%C3%B3n%20a%20la%20Reconstrucci%C3%B3n%203D](http://www.elai.upm.es/webantigua/spain/Investiga/GCII/personal/lrodriguez/web3D/reconstruccion_3d.htm#Introducci%C3%B3n%20a%20la%20Reconstrucci%C3%B3n%203D)

Doc-Ok.org. (2013). Ejes coordenados dispositivo Kinect. Recuperado de Multi-Kinect camera calibration de <https://docplayer.es/16576055-Segmentacion-de-imagenes-obtenidas-a-traves-de-un-sensor-kinect-con-criterios-morfologicos-y-atributos-visuales-de-profundidad.html>.

Dong, M., Ciao, L., Zhang, D.-M., & Guo, R. (2016). UAV flight controlling based on Kinect for Windows V2. IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Dulko, F. (s.f.). 360° Color and Depth Mapping of an Indoor Room: Microsoft Kinect 2.0. IEEE. Recuperado el 2 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Duque, E. (2015). ¿Qué es el Microsoft Kinect?, Toward a brand-NUI World. Recuperado el 26 de abril de 2018 de <https://edwinnui.wordpress.com/2015/02/03/qu-es-el-microsoft-kinect/>

- Espinosa Bernal, Satterthwaite, Napoli, Glass, Tucker, & Obeid. (s.f.). Kinect v2 Accuracy as a Body Segment Measuring Tool. IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Forero, M. G., Aranzazu Buitrago, N. F., & Flórez Larrahondo, G. (2001). Reconstrucción 3D Usando Superficies Trianguladas Dados Contornos Paralelos. Revista Ingeniería e Investigación, 64-67. Recuperado el 25 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Gambi, E., Agostinelli, A., Belli, A., Burattini, L., Cippitelli, E., Fioretti, S., Spinsante, S. (2017). Heart Rate Detection Using Microsoft Kinect: Validation and Comparison to Wearable Devices. MDPI. Recuperado el 24 de mayo de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Ilbay Paca, L. G. (2014). Reconstrucción activa de objetos 3D mediante escaneo láser y generación de la vista por medio del software Matlab. Quito. Recuperado el 25 de abril de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Kameyama, N., & Hidaka, K. (2017). A Sensor-Based Exploration Algorithm for Autonomous Map. IEEE. Recuperado el 25 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Kim, H., Remaggi, L., Jackson, P. J., Fazi, F. M., & Hilton, A. (2017). 3D Room Geometry Reconstruction Using Audio-Visual Sensors. IEEE, 621-629. Recuperado el 25 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Kundu, A. S., Mazumder, O., Dhar, A., & Bhaumik, S. (2016). Occupancy Grid Map Generation using 360°. IEEE. Recuperado el 20 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.



- Leap Motion. (2018). Imagen Leap Motion Controller. Recuperado de <http://es.ccm.net/faq/11108-leap-motion#que-es-leap-motion>.
- Matlab. (2015). Nube de puntos de una tetera. Recuperado el 2 de junio de 2018 de MathWorks de <https://la.mathworks.com/help/vision/3-d-point-cloud-processing.html>
- Mendoza Rivera, M. J., & Guamushig Lasluisa, B. G. (2017). Revisión del estado del arte de aplicaciones de control usando tecnología Kinect. Quito. Recuperado el 12 de mayo de 2018 de [http://biblioteca.udla.edu.ec/client/es\\_EC/default](http://biblioteca.udla.edu.ec/client/es_EC/default).
- Microsoft. (2009). Diagrama sensores Kinect. Recuperado de <https://www.microsoft.com/spain/accesabilidad/microsoft/research.aspx>
- Microsoft. (2018). Imagen Kinect. Recuperado de <https://msdn.microsoft.com/en-us/library/jj131033.aspx>
- Navas, D., Vargas, J., & Morales, L. (2017). The nearest object localization through 3D LiDAR reconstruction using an embedded. Quito. Recuperado el 24 de abril de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Riofrío Hidalgo, S. J. (2014). Implementación de un sistema de reconocimiento de patrones de movimiento con las extremidades superiores del cuerpo humano. Quito. Recuperado el 20 de mayo de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Riofrío, S., Pozo, D., Rosero, J., & Vásquez, J. (2017). Gesture Recognition using Dynamic Time Warping and Kinect. 7. Recuperado el 22 de mayo de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Rodríguez, C. (2017). Sony Latin America. Recuperado el 10 de abril de 2018 de Universe: <https://alphauniverse-latin.com/notas/elementos-que-componen-la-imagen>

- Shen, B., Yin, F., & Chou, W. (2017). A 3D Modeling Method of Indoor Objects Using Kinect Sensor. 5. Recuperado el 20 de mayo de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Siv, R., Ardiyanto, I., & Hartanto, R. (2018). 3D Human Face Reconstruction Using Depth Sensor. IEEE. Recuperado el 24 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- SoftKinetic. (2018). Imagen de dispositivo DepthSense. Recuperado de <https://www.softkinetic.com/Store/ProductID/29>
- Stereo Labs. (s.f.). Imagen de dispositivo Zed. Recuperado de <https://www.stereolabs.com/>
- Teledet. (s.f.). Matrices de colores de imágenes digitales. Recuperado de <http://www.teledet.com.uy/tutorial-imagenes-satelitales/estructura-imagenes-digitales.htm>
- Terven, J. (2017). File Exchange. Recuperado el 10 de junio de 2018 de Matlab:  
<https://www.mathworks.com/matlabcentral/fileexchange/53439-kinect-2-interface-for-matlab>
- Villegas, J. D. (2016). Red de las preguntas. Recuperado de Universidad EAFIT:  
<http://www.eafit.edu.co/ninos/reddelaspreguntas/maquinasyenergia/Paginas/como-funcionan-las-camaras-fotograficas.aspx>
- Vox, J. P., & Wallhoff, F. (2017). Recognition of Human Motion Exercises Using Skeleton Data and SVM for Rehabilitative Purposes. IEEE. Recuperado el 2 de junio de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.
- Wongwilai, N., Niparnan, N., & Sudsang, A. (s.f.). Calibration of an Eye-in-Hand System using SoftKinetic DepthSense. IEEE. Recuperado el 24 de mayo de 2018 <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

Zhang, S., He, W., Yu, Q., & Zheng, X. (s.f.). Low-Cost Interactive Whiteboard Using the. IEEE. Recuperado el 25 de mayo de 2018 de <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9907>.

## **Anexos**

## Función escaneoKinect

```
function [A] = escaneoKinect()
%ESCANEEO DE NUBES DE PUNTOS CON KINECT V2
%AUTOR: ANDRÉS BALDEÓN
%MAYO 2018

colorDevice = imaq.VideoDevice('kinect',1);
depthDevice = imaq.VideoDevice('kinect',2);

step(colorDevice);
step(depthDevice);

colorImage = step(colorDevice);
depthImage = step(depthDevice);

ptCloud =
pcfromkinect(depthDevice,depthImage,colorImage);

maxDistance = 7;

roi = [-inf,inf;-1,inf;-inf,inf];
sampleIndices = findPointsInROI(ptCloud,roi);

[model2,inlierIndices,outlierIndices] =
pcfitplane(ptCloud,...

maxDistance,'SampleIndices',sampleIndices);
plane2 = select(ptCloud,inlierIndices);

release(colorDevice);release(depthDevice);

A = plane2.Location;
%ptcloudResult = ptCloud;
end
```

## Función de TransformaMatrices

```
function Transformada =  
TransformaMatrices(A,d,angulo)  
%Transforma la nube de puntos  
%Autor: Andrés Baldeón  
% Transforma la nube de puntos con respecto al eje  
Y  
  
MatrizTransformacion = [cos(angulo),0,sin(angulo),0;  
                        0,1,0,0;  
                        -  
sin(angulo),0,cos(angulo),d;  
                        0,0,0,1];  
  
Traspuesta = A;  
  
Traspuesta(4,:) = 1;  
  
Transformada = inv(MatrizTransformacion) *  
Traspuesta;  
  
end
```

## Función LimpiaNube

```
function [final] = LimpiaNube(final)
%LimpiaNube elimina los puntos dispersos de la nube
encontrados por el
%Kinect
% Se eliminan puntos que se encuentren dentro de
unos límites
% establecidos para lograr un filtrado de la imagen
que se necesita

jj = 1;

for ii = 1:size(final,2)

    if final(1,ii)>0.15 || final(2,ii)>0.17 ||
final(3,ii)>1.2 || final(1,ii)<-0.15 ||
final(2,ii)<0.03 || final(3,ii)<0.6

        eliminacion(1,jj) = ii;
        jj = jj+1;

    end

end

final(:,eliminacion) = [];
```

## Función abrePuerto

```
function [puerto_serial] = abrePuerto()
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
com='COM4';
%Se inicializa el puerto serial a utilizar
delete(instrfind({'Port'}, {com}));
puerto_serial=serial(com);
puerto_serial.BaudRate=9600;
warning('off', 'MATLAB:serial:fscanf:unsuccessfulRead')
;

fopen(puerto_serial);
end
```

5.3



## Función enviaOrdenGiro

```
function [] = enviaOrdenGiro(puerto_serial)
%Envía orden al motor para girar
% Envía el caracter determinado por comunicación
serial para que el motor
% gire el ángulo establecido

%Abro el puerto serial
pause(3);
dato='a';
fwrite(puerto_serial,dato,'char');      % se envía un
dato

pause(3);

end
```

## Función cierraPuerto

```
function [] = cierraPuerto(puerto_serial)
%UNTITLED2 Summary of this function goes here
% Detailed explanation goes here
```

```
fclose(puerto_serial);
delete(puerto_serial)
clear puerto_serial
```

```
end
```

5.4

## Función TomaMatricesDeKinect

```
%CODIGO DE ESCANEEO KINECT V2
clear all; clc;
i = 0; %inicializa contador
d = 0.9; %distancia del sensor
al objeto
numeroTomas = 4; %numero de capturas que
se realizarán
angulo = deg2rad(360)/numeroTomas; %angulo de
rotacion
puerto_serial = abrePuerto();
j = 1;

while(i < 360)

    enviaOrdenGiro(puerto_serial);
%envia orden al motor para girar
    A = escaneoKinect(); %ejecuta la función de
escaneo
    % A = ptCloudResult.Location; %almacena la
matriz de la nube de puntos

    A = LimpiaNube(A');

    Transformada = TransformaMatrices(A,d,deg2rad(i));
%rota y traslada la matriz

    final(:,j:(size(Transformada,2)+j-
1))=Transformada; %almacena en la misma matriz los
puntos de todas las matrices
    j=size(final,2)+1;
    i = i + 90;

end

cierraPuerto(puerto_serial);

figure
plot3(final(1,:),final(2,:),final(3,:),'.');
```

## Programación del microcontrolador

```
#include <Arduino.h>
#include "A4988.h"

// using a 200-step motor (most common)
#define MOTOR_STEPS 200
// configure the pins connected
#define DIR 5
#define STEP 6
#define MS1 10
#define MS2 11
#define MS3 12
A4988 stepper(MOTOR_STEPS, DIR, STEP, MS1, MS2, MS3);

void setup()
{
  // Set target motor RPM to 1RPM and microstepping to 1 (full step mode)
  stepper.begin(5, 1);
  Serial.begin(9600);
}

void loop()
{
  if(Serial.read()=='a')
  {
    stepper.rotate(90);
    Serial.println("b");
  }
  delay(100); }
```

