

# **HERRAMIENTA SOFTWARE PARA LA EXTRACCIÓN AUTOMATIZADA DE UN MODELO 3D DE OBJETOS USANDO EL SENSOR KINECT - DOEVERYTHING3D**

**PRESENTADO POR:**

**JUAN SEBASTIAN ASTUDILLO ZAMBRANO**

**UNIVERSIDAD DEL VALLE  
FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA ELÉCTRICA Y  
ELECTRÓNICA  
PROGRAMA ACADEMICO DE INGENIERÍA  
ELECTRÓNICA**

**SANTIAGO DE CALI, JUNIO DE 2018**

# **HERRAMIENTA SOFTWARE PARA LA EXTRACCIÓN AUTOMATIZADA DE UN MODELO 3D DE OBJETOS USANDO EL SENSOR KINECT - DOEVERYTHING3D**

**PRESENTADO POR:**

**JUAN SEBASTIAN ASTUDILLO ZAMBRANO**

**DIRIGIDO POR:**

**ING. BLADIMIR BACCA CORTES, PHD.**

**UNIVERSIDAD DEL VALLE  
FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA ELÉCTRICA Y  
ELECTRÓNICA  
PROGRAMA ACADEMICO DE INGENIERÍA  
ELECTRÓNICA**

**SANTIAGO DE CALI, JUNIO DE 2018**

# **RESUMEN**

La tecnología de reconstrucción 3D es ampliamente utilizada en múltiples aplicaciones, algunas de estas se encuentran en los medios de comunicación, cine, videojuegos, industrias y en la enseñanza [4]. Lamentablemente el uso de estos dispositivos requiere cierta experticia en computación y los costos de adquisición estos son muy elevados ( $\sim$ 2000 USD) [9].

Este trabajo de grado presenta el diseño, desarrollo y validación, de una alternativa asequible y con una interfaz intuitiva para la extracción automatizada de modelos 3D de objetos mediante el uso del sensor Microsoft Kinect a la cual se le ha denominado DoEverything3D.

El diseño de DoEverything3D inició principalmente a la recolección y análisis de antecedentes sobre la generación de modelos 3D. Dicho análisis dio como resultado la extracción de una metodología común presente en cada uno de los antecedentes analizados para la generación de los modelos 3D, que luego sería implementada en la herramienta.

Por otra parte, para la validación de DoEverything3D se desarrolló un protocolo de pruebas de integración y cuantitativas. Las pruebas de integración resultaron exitosas evidenciando la correcta implementación de cada requerimiento del sistema. Las pruebas cuantitativas por su parte comprobaron la buena calidad de los modelos 3D generados por el sistema presentando un error medio de 0.3988 mm con una desviación estándar ( $5\sigma$ ) de 8.846 mm.

# **ABSTRACT**

3D reconstruction technology is widely used in multiple applications, some of these are found in media, film, videogames, industries and teaching [4]. Unfortunately, the use of these devices requires some computer expertise and the acquisition costs are very high ( $\sim$  2000 USD) [9].

This work presents the design, development and validation of an affordable alternative with an intuitive interface for the automated extraction of 3D models of objects by using the Microsoft Kinect sensor, which has been called DoEverything3D.

The design of DoEverything3D started mainly with the collection and analysis of the generation of 3D models. This analysis resulted in the extraction of a common methodology present at each of the analyzed antecedents for the generation of 3D models, which would then be implemented in the tool.

On the other hand, for the validation of DoEverything3D a protocol of integration and quantitative tests was developed. The integration tests were successful, evidenced by the correct implementation of each system requirement. The quantitative tests verified the good quality of the 3D models generated by the system, presenting an average error of 0.3988 mm with a standard deviation ( $5\sigma$ ) of 8,846 mm.

# **AGRADECIMIENTOS**

Después de un intenso período de doce meses, hoy es el día: escribo este apartado de agradecimientos para finalizar mi trabajo de grado. Ha sido un período de aprendizaje intenso, no solo en el campo académico, sino también a nivel personal. Escribir este trabajo ha tenido un gran impacto en mí y es por eso que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Primero de todo, a me gustaría agradecer al profesor y director de este trabajo de grado el Ing. Eval Bladimir Bacca PhD, por su disposición, conocimiento impartido, compromiso y motivación brindada durante todo este trabajo. Definitivamente me ha brindado todas las herramientas necesarias para completar mi trabajo de grado satisfactoriamente.

A Dios y a mi familia por brindarme el apoyo, el soporte, por ser mi motor día a día y mi motivación por siempre ser mejor en todos los aspectos de mi vida.

Gracias a mi pareja por entenderme en todo, gracias a ella porque en todo momento fue un apoyo incondicional en mi vida, fue la felicidad encajada en una sola persona, fue mi todo reflejado en otra persona a la cual yo amo demasiado, y por la cual estoy dispuesto a enfrentar todo y en todo momento.

A mis compañeros que de igual manera contribuyeron con sus palabras de apoyo y especialmente a Hanner Yesid por brindarme a lo largo de este proyecto de grado, su valiosas ideas y conocimiento.

¡Muchas gracias a todos!

# TABLA DE CONTENIDOS

1. Acercamiento a la propuesta .....	1
1.1. Introducción .....	1
1.2. Descripción del problema .....	1
1.3. Objetivos .....	2
1.3.1. Objetivo general .....	2
1.3.2. Objetivo específico .....	2
1.4. Solución implementada .....	2
1.5. Estructura del documento .....	3
2. Estado del arte y marco teórico .....	4
2.1. Introducción .....	4
2.2. Estado del arte .....	5
2.2.1. Análisis de las metodologías de generación de modelos 3D .....	7
2.3. Marco teórico .....	8
2.3.1. Escáner 3D .....	9
2.3.2. Microsoft Kinect .....	10
2.4. Transformación de imagen de profundidad a nubes de puntos 3D .....	10
2.4.1. Cálculo del mapa de profundidad .....	10
2.4.2. Modelo de cámara Pinhole.....	11
2.5. Nubes de puntos 3D.....	12
2.6. Descriptores de nubes de puntos 3D .....	12
2.6.1. Vóxeles.....	12
2.6.2. Normales de superficie .....	13
2.6.3. PHF .....	13
2.6.4. FPFH .....	15
2.7. Registro de nubes de puntos 3D .....	15
2.7.1. Transformación 3D.....	17
2.7.2. Algoritmo ICP .....	19
2.7.3. Algoritmo SAC-IA.....	20
2.8. Alineación global.....	21
2.8.1. Algoritmo utilizado .....	21
2.9. Segmentación de plano tierra.....	22
2.10. Reducción de ruido .....	23
2.10.1. Reducción de ruido mediante el algoritmo MLS .....	23
2.11. Generación de superficie .....	23
2.11.1. Triangulación de Delaunay 2D .....	24
2.11.2. Triangulación de Delaunay 3D .....	25
2.12. Proceso unificado racional (RUP) .....	25
2.12.1. Principios de desarrollo.....	26
2.12.2. Fases del RUP .....	26
2.13. Observaciones Finales.....	26
3. Desarrollo de 360-Platform-Assistant (360PA) .....	28
3.1. Introducción.....	28

3.2. Requerimientos de Diseño.....	28
3.2.1. Requerimientos funcionales y no funcionales .....	28
3.2.2. Diagrama conceptual.....	29
3.3. Modulos de 360-Platform-Assistant (360PA).....	30
3.3.1. Modulo PC.....	30
3.3.2. Modulo microcontrolador.....	30
3.3.3. Módulo Plataforma .....	32
3.4. Conclusiones .....	35
4. Desarrollo del aplicativo DoEverything3D.....	36
4.1. Introducción.....	36
4.2. Requerimientos de Diseño.....	36
4.2.1. Requerimientos funcionales y no funcionales .....	37
4.2.2. Diagrama Conceptual .....	38
4.2.3. Diagrama de Clases .....	39
4.2.4. Modelo de información.....	40
4.3. Interfaz Gráfica de DoEverything3D .....	41
4.3.1. Módulo de usuarios .....	41
4.3.2. Módulo de Configuración.....	43
4.3.3. Módulo de Calibración.....	44
4.3.4. Módulo de Escaneo.....	46
4.3.5. Módulo de Historial de trabajos.....	51
4.4. Conclusiones .....	53
5. Pruebas y resultados.....	55
5.1. Introducción.....	55
5.2. Pruebas de Integración.....	55
5.2.1. Registro de usuario, inicio de sesión y configuración de parámetros de funcionamiento del aplicativo .....	56
5.2.2. Calibración del sensor Kinect, escaneo 3D de un objeto y visualización del modelo 3D resultante. ....	58
5.2.3. Almacenamiento de un modelo 3D, visualización y exportado de un modelo 3D del historial de trabajos. ....	65
5.3. Pruebas Cuantitativas.....	69
5.3.1. Análisis de Error de Modelado .....	69
5.3.2. Extracción de Modelos 3D de diferentes Objetos .....	72
5.4. Conclusiones .....	73
6. Conclusiones y trabajo futuro .....	75
6.1. Conclusiones .....	75
6.2. Trabajo futuro .....	76
7. Anexos .....	77
8. Referencias .....	78

# TABLA DE FIGURAS

<b>Figura 1.1</b> Reconstrucción 3D de un edificio mediante visión estereoscópica [7].....	1
<b>Figura 1.2</b> Diagrama conceptual de la propuesta presentada (DoEverything3D).....	2
<b>Figura 2.1.</b> Diagrama de flujo en donde se condensan los patrones de metodología identificados de los trabajos analizados.....	8
<b>Figura 2.2.</b> Disposición de los sensores en el Microsoft Kinect [32].....	10
<b>Figura 2.3.</b> Calculo de profundidad desde la información de distancia.....	10
<b>Figura 2.4.</b> Modelo de cámara Pinhole [33].....	11
<b>Figura 2.5.</b> Nube de puntos de un cilindro al cual es escaneo desde una vista superior.....	12
<b>Figura 2.6.</b> Comparación geométrica entre un pixel (izquierda) y un voxel (derecha) [36].....	12
<b>Figura 2.7.</b> Comparación nube de puntos original (izquierda) y el resultado de su reducción mediante voxéles (derecha) [37].....	13
<b>Figura 2.8.</b> Vectores normales a una superficie 3D [38].....	13
<b>Figura 2.9.</b> Diagrama de región de influencia del cálculo de PFH para un punto de consulta $pq$ [39].....	14
<b>Figura 2.10.</b> Marco referencial de coordenadas $(u, v, w)$ propuesto en donde se grafican dos puntos $ps$ y $pt$ y sus normales asociadas $ns$ y $nt$ [40].....	14
<b>Figura 2.11.</b> Ejemplos de representaciones de Histogramas de Puntos para diferentes puntos en una nube [41].....	15
<b>Figura 2.12.</b> Diagrama de la región de influencia para un conjunto k-vecindario centrado en $pq$ [42]..	15
<b>Figura 2.13.</b> Registro de 3 nubes de puntos de un objeto (tetera) [43].....	16
<b>Figura 2.14.</b> Registro de 2 nubes de puntos de un rostro humano [44].....	16
<b>Figura 2.15.</b> Rotación alrededor del eje $x(\alpha)$ , eje $y(\beta)$ y el eje $z(\gamma)$ .....	17
<b>Figura 2.16.</b> Translación de $p1$ a $p2$ .....	18
<b>Figura 2.17.</b> Resultados obtenidos después del registro con SAC-IA en dos vistas parciales del modelo de conejito de Stanford [47].....	21
<b>Figura 2.18.</b> En (a), (b) se aprecia el resultado del registro sin y con alineación global. ....	21
<b>Figura 2.19.</b> Vectores normales a una superficie 3D. ....	22
<b>Figura 2.20.</b> En (a) y (b), se aprecia el modelo antes y después de realizar la reducción de ruido mediante la técnica MLS respectivamente.....	23
<b>Figura 2.21.</b> a) Relación entre la triangulación de Delaunay y un casco convexo 3D. Puntos proyectados a un paraboloide. b) Cálculo del casco convexo en el paraboloide. c) El casco convexo proyectado en el plano [54].....	24
<b>Figura 2.22.</b> Un punto específico se encuentra dentro de la circunferencia circunscrita por $p, q$ y $r$ , si solo si su proyección del espacio 3D está por debajo del plano que contiene las proyecciones $p', q'$ y $r'$ [56]..	25
<b>Figura 2.23.</b> Resultado obtenido al realizar el algoritmo de triangulación 3D de Delaunay a una nube de puntos 3D correspondientes a una mano humana .....	25
<b>Figura 2.24.</b> Esfuerzo en actividades según la fase del proyecto siguiendo la metodología RUP [58]. ...	26
<b>Figura 3.1.</b> Diagrama conceptual del sistema 360-PLATFORM-ASSISTANT (360PA).....	29
<b>Figura 3.2.</b> Ejemplo de uso de la librería pySerial para Python 3 en donde se envían ordenes de rotaciones indefinidas de una posición ( $18^\circ$ ) cada 2 segundos. ....	30
<b>Figura 3.3.</b> Componentes principales del módulo microcontrolador. Placa de desarrollo Arduino UNO (izquierda) y extensión LCD DFROBOT compatible con Arduino UNO (derecha). ....	30
<b>Figura 3.4.</b> Diagrama de flujo sobre la metodología que rige a el modulo microcontrolador para analizar y vigilar que una orden recibida del módulo PC se ejecute adecuadamente en el módulo plataforma.....	31
<b>Figura 3.5.</b> Diagrama esquemático del módulo microcontrolador. ....	32
<b>Figura 3.6.</b> Montaje del módulo microcontrolador. ....	32
<b>Figura 3.7.</b> Modelo conceptual de la estructura giratoria.....	33
<b>Figura 3.8.</b> a) Servomotor Tower Pro MG996R, .....	33
b) Kit módulo de encoder y disco de 20 ranuras ADAFRUIT FC-03 .....	33
<b>Figura 3.9.</b> Módulo de encoder y disco de 20 ranuras ADAFRUIT. ....	34
<b>Figura 3.10.</b> Montaje del módulo plataforma conjunto al módulo microcontrolador. ....	34
<b>Figura 4.1.</b> Diagrama conceptual del sistema DoEverything3D. ....	38
<b>Figura 4.2.</b> Diagrama de clases del sistema DoEverything3D. ....	40
<b>Figura 4.3.</b> Estructura del modelo de información del sistema DoEverything3D. ....	41
<b>Figura 4.4.</b> a) Interfaz gráfica pestaña para inicio de sesión de usuario. ....	41
b) Interfaz gráfica pestaña para registro de usuarios.....	41
<b>Figura 4.5.</b> Procedimiento para iniciar sesión a un usuario registrado.....	42
<b>Figura 4.6.</b> Procedimiento para registrar nuevos usuarios al aplicativo. ....	42
<b>Figura 4.7.</b> Interfaz gráfica principal, pestaña configuración. ....	43

<b>Figura 4.8.</b> Procedimiento para actualizar y almacenar los parámetros de operación del aplicativo asociado a un usuario.....	43
<b>Figura 4.9.</b> Interfaz gráfica principal, pestaña calibración. ....	44
<b>Figura 4.10.</b> Descripción grafica de cómo debe ir posicionada el sensor Microsoft Kinect al realizar la calibración. ....	44
<b>Figura 4.11.</b> Procedimiento que se realiza para la calibración del sensor Microsoft Kinect y almacenado del resultado de dicha calibración de manera asociada a un usuario. ....	45
<b>Figura 4.12.</b> a) Regresión lineal de un conjunto de puntos. ....	46
b) error $ei$ en la coordenada $y$ entre la regresión lineal y el punto real. ....	46
<b>Figura 4.13.</b> Interfaz gráfica principal, pestaña escaneo. ....	47
<b>Figura 4.14.</b> Procedimiento que se realiza para la calibración del sensor Microsoft Kinect y almacenado del resultado de dicha calibración de manera asociada a un usuario. ....	48
<b>Figura 4.15.</b> Ventana dedicada a limitar la ROI mediante 4 marcadores rojos (La ROI es el área contenida por los 4 marcadores). ....	48
<b>Figura 4.16.</b> a) Nube completa con un cubo como objeto de interés. ....	49
b) Plano tierra de la nube identificado mediante el algoritmo. ....	49
c) Nube con el objeto de interés segmentado del plano de tierra. ....	49
<b>Figura 4.17.</b> a) Par de nubes contiguas de un cubo (Azul y amarillo) antes de registrar. ....	50
b) Par de nubes contiguas de un cubo (Azul y amarillo) luego de registrar mediante las transformadas relativas. ....	50
<b>Figura 4.18.</b> a) Veinte nubes contiguas de un cubo (diferentes colores) antes de registrar. ....	50
b) Veinte nubes contiguas de un cubo (diferentes colores) luego de registrar mediante las transformadas globales.....	50
<b>Figura 4.19.</b> a) Nube de puntos 3D de una caja sin filtrar.....	51
b) Nube de puntos 3D de una caja después del filtrado.....	51
<b>Figura 4.20.</b> Superficie generada por el aplicativo a la nube de puntos filtrada de la figura 4.18.b.....	51
<b>Figura 4.21.</b> Interfaz gráfica principal, pestaña Historial de modelos.....	52
<b>Figura 4.22.</b> Procedimiento que se realiza para la visualización de los modelos 3D asociados al usuario y también para exportar dichos modelos a una ubicación específica en el ordenador. ....	52
<b>Figura 5.1.</b> Modelo 3D generado por el sistema DoEverything3D de una caja de dimensiones 10 cm de largo, 7 cm de ancho y 6 cm de alto superpuesto sobre su modelo ideal (Negro). El pseudocolor aplicado al modelo generado por el sistema DoEverything3D viene dado en función al error presente respecto al modelo ideal (Valor del error en metros e histograma del error en la columna derecha). .....	70
<b>Figura 5.2.</b> Modelo 3D generado por el sistema DoEverything3D de una caja de dimensiones 17 cm de largo, 13 cm de ancho y 11 cm de alto superpuesto sobre su modelo ideal (Negro). El pseudocolor aplicado al modelo generado por el sistema DoEverything3D viene dado en función al error presente respecto al modelo ideal (Valor del error en metros e histograma del error en la columna derecha). .....	70
<b>Figura 5.3.</b> Muestra de diferentes modelos 3D extraídos mediante el sistema DoEverything3D.....	72
<b>a)</b> Modelo 3D generado por el sistema DoEverything3D de una bota.....	72
<b>b)</b> Modelo 3D generado por el sistema DoEverything3D de una caja. ....	72
<b>c)</b> Modelo 3D generado por el sistema DoEverything3D de un juguete en forma de pingüino. ....	72
<b>d)</b> Modelo 3D generado por el sistema DoEverything3D de una tetera. ....	73
<b>Figura 5.4.</b> Modelo 3D generado por el sistema DoEverything3D de objeto simétrico (rollo de papel)... <td>73</td>	73
<b>Figura 7.1.</b> Distribución de anexos. ....	77

# GLOSARIO

<b>2D</b>	Representación gráfica que sólo emplea dos de las tres dimensiones del espacio.
<b>3D</b>	Representación gráfica que emplea las tres dimensiones del espacio.
<b>360PA</b>	Plataforma asistente 360 (acrónimo inglés de 360 Platform Assitant), sistema desarrollado en este documento para rotaciones controladas a objetos.
<b>CloudCompare</b>	Software utilizado para comparar y calcular diferencias entre nube de puntos
<b>CSG</b>	Modelado con geometría sólida reconstructiva (acrónimo inglés de Constructive Solid Geometry), técnica utilizada para generar la superficie digital del modelo 3D.
<b>CUDA</b>	Arquitectura unificada de dispositivos de computo, paralelismo de altas cantidades de unidades de computo.
<b>FPFH</b>	Histogramas de características de punto rápido (acrónimo inglés de Fast Point Feature Histograms), optimización del descriptor de forma PHF que tiene como finalidad representar las propiedades geométricas y la curvatura de la vecindad de un punto en una nube 3D.
<b>GUI</b>	Interfaz gráfica de usuario (acrónimo inglés de Grafical User Interface), comúnmente la parte donde interactúa el usuario de software.
<b>ICP</b>	Puntos más cercanos iterativo (acrónimo inglés de Iterative Closets Point), algoritmo usado para el registro de nubes de puntos 3D.
<b>LCD</b>	Pantalla de cristal líquido (acrónimo inglés de Liquid Crystal Display), sistema que utilizan determinadas pantallas electrónicas para mostrar información visual.
<b>MI</b>	Información mutua (acrónimo inglés de Mutual Information), información que comparte dos grupos de datos o imágenes.
<b>MLS</b>	Moviendo el cuadrado menor (acrónimo inglés de Moving Least Square), algoritmo utilizado para eliminar el ruido en modelos 3D.
<b>MysqlConnector</b>	API del lenguaje de programación Python para el manejo de base de datos SQL.
<b>Numpy</b>	API aritmetica del lenguaje de programación Python para el manejo de vectores, matrices y realización de diferentes procedimientos matemáticos.
<b>NURBS</b>	B-splines racionales no uniformes o NURBS (acrónimo inglés de non-uniform rational B-spline) es un modelo matemático muy utilizado en la computación gráfica para generar y representar curvas y superficies.
<b>Open3D</b>	API del lenguaje de programación Python para la visualización, realización de operaciones espaciales y registro entre nubes de puntos 3D.
<b>OpenGL</b>	API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
<b>PCD</b>	Formato para almacenar nubes de puntos 3D.
<b>PCL</b>	API multilenguaje para el procesamiento de imágenes 2D y nubes de puntos 3D.
<b>PFH</b>	Histogramas de características de punto (acrónimo inglés de Point Feature Histograms), descriptor de forma que tiene como finalidad representar las propiedades geométricas y la curvatura de la vecindad de un punto en una nube 3D.
<b>PLY</b>	Formato de archivo de polígono, desarrollado en la Universidad de Stanford.
<b>Pymesh</b>	API del lenguaje de programación Python para la generación de superficies sobre nubes de puntos 3D.
<b>PyQt5</b>	API del lenguaje de programación Python para la creación de interfaces gráficas.
<b>Pyqtgraph</b>	API del lenguaje de programación Python para la visualización de gráficos 2D.
<b>PySerial</b>	API del lenguaje de programación Python para el manejo y control del Puerto USB.

<b>Python</b>	Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
<b>RANSAC</b>	Muestra aleatoriedad de consenso (acrónimo inglés de Random sample Consensus), algoritmo para la extracción de primitivas geométricas.
<b>RGB</b>	Rojo, verde y azul (siglas en inglés de red green blue), composición del color a partir de la intensidad de los colores rojo, verde y azul.
<b>RGBD</b>	Rojo, verde, azul y distancia (siglas en inglés de red green blue distance), composición del color a partir de la intensidad de los colores rojo, verde y azul con información de profundidad añadida.
<b>ROI</b>	Región de interés (Region of Interest).
<b>RUP</b>	Proceso unificado racional, metodología para el desarrollo de software
<b>SAC-IA</b>	Alineación inicial del consenso de muestreo (acrónimo inglés de Sample Consensus Initial Alignment), algoritmo usado para el registro de nubes de puntos 3D.
<b>SDF</b>	Campo de distancia signado (acrónimo de Signed Distance Field), función que asocia un punto 3D con la distancia signada más cercana a una superficie.
<b>SDK</b>	Kit de desarrollo de software (acrónimo de Software Development Kit), grupo de herramientas para el desarrollo de software.
<b>SGM</b>	Emparejamiento semi-global (acrónimo de Semi-global matching), algoritmo usado para encontrar vecindad de píxeles semejantes de una manera de poco costo computacional
<b>SIRUNS</b>	Segmentación de información de rango usando normales de superficie (acrónimo de SIRUNS), algoritmo usado para segmentar planos de una nube de puntos 3D.
<b>SQL</b>	Lenguaje de consulta estructurada (acrónimo inglés de Structured Query Language), es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.
<b>USB</b>	Bus universal en serie (acrónimo inglés de Universal Serial Bus), es un bus de comunicaciones que sigue un estándar que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica.

# CAPÍTULO 1

## 1. ACERCAMIENTO A LA PROPUESTA

- 
- 1.1. Introducción
  - 1.2. Descripción del problema
  - 1.3. Objetivos
    - 1.3.1. Objetivo general
    - 1.3.2. Objetivos específicos
  - 1.4. Solución implementada
  - 1.5. Estructura del documento
- 

### 1.1. INTRODUCCIÓN

El escaneo o modelamiento 3D es el proceso de desarrollar una representación matemática de la superficie tridimensional de un objeto o un ambiente [1]. Un modelo 3D puede representar un objeto real usando una colección de puntos en el espacio 3D conocido como una nube de puntos 3D [2]. Estos puntos también pueden estar conectados por varias entidades geométricas tales como triángulos, líneas y superficies curvas [3] por ejemplo, la figura 1.1 muestra una reconstrucción 3D de un edificio generado mediante la técnica de visión estereoscópica. Hay muchas de aplicaciones de modelado 3D en los medios de comunicación, cine, videojuegos e industrias, en donde la generación de modelos 3D de objetos se hace de una labor muy común [4].

A pesar de la gran aplicabilidad de estos sistemas de extracción de modelos 3D de objetos [4], no existen alternativas a un costo asequible, además de poseer interfaces con mucha información abrumadora para el usuario haciendo que el uso de estas aplicaciones sea costoso y de difícil uso. Dada la introducción del dispositivo Microsoft Kinect, sensor que puede entregar datos de profundidad con una exactitud aceptable (alrededor de 5 mm a 9 mm dependiendo de la distancia entre el sensor y el objetivo a escanear) [5], y con un precio bastante asequible (aproximadamente 200 USD) [6]. Esto ha proporcionado a muchos entusiastas e industrias la herramienta necesaria para realizar sus ideas y dar entrada a nuevas implementaciones y enfoques a partir de las aplicaciones ya dichas.



El propósito de este trabajo es desarrollar un aplicativo con interfaz de fácil uso para la extracción automatizada de modelos 3D de objetos usando el sensor Microsoft Kinect, tratando de explotar de la mejor manera sus características para obtener la mejor calidad posible del modelo 3D. Dando así una solución para la problemática planteada.

**Figura 1.1** Reconstrucción 3D de un edificio mediante visión estereoscópica [7].

### 1.2. DESCRIPCIÓN DEL PROBLEMA

El escaneo o modelamiento 3D es un término bastante amplio que abarca diferentes tipos de tecnologías y procedimientos, pero, en general, consiste en la técnica para capturar datos del mundo real y convertirlos en un modelo digital 3D [1]. Dicho método se ha utilizado durante años en procesos industriales [4] y comerciales que van desde la inspección de sitios de construcción [4], la catalogación de obras de arte [4], excavación de fósiles [4] y la toma de medidas precisas de una habitación [4] con el fin, por ejemplo, de preservar la instantánea digital de una escena de crimen.

Sin embargo, debido al aumento de la impresión 3D, especialmente para los fabricantes y entusiastas, el deseo de capturar objetos reales con la intención de modificar sus características mediante software para luego convertirlos en objetos fácilmente imprimibles se ha hecho mucho más grande.

Dado lo anterior, el mercado ha ido creciendo conforme al aumento en la demanda de escáneres 3D [8], lo que contribuye al conocimiento de muchas opciones para todo tipo de presupuesto, con gran variedad de tecnologías [9][10] como son: la fotogrametría, la triangulación láser, las cámaras RGB con sensores de profundidad, la luz estructurada, entre otras.

A pesar del numeroso catálogo de escáneres no existe una alternativa asequible que produzca resultados con buena relación costo-resolución además de poseer una plataforma intuitiva y sagaz para el usuario. Dichas limitaciones hacen que la alternativa de adquirir un escáner 3D para el modelamiento de objetos, para lo cual se requiere de una alta precisión, sea una opción bastante costosa y de difícil acceso para cualquier usuario.

De acuerdo con lo expuesto anteriormente, este proyecto se encamina con base en la siguiente pregunta problema:

¿Cómo diseñar un aplicativo con interfaz intuitiva que permita la reconstrucción 3D de un objeto en alta precisión, por medio de un dispositivo de adquisición asequible?

### 1.3. OBJETIVOS

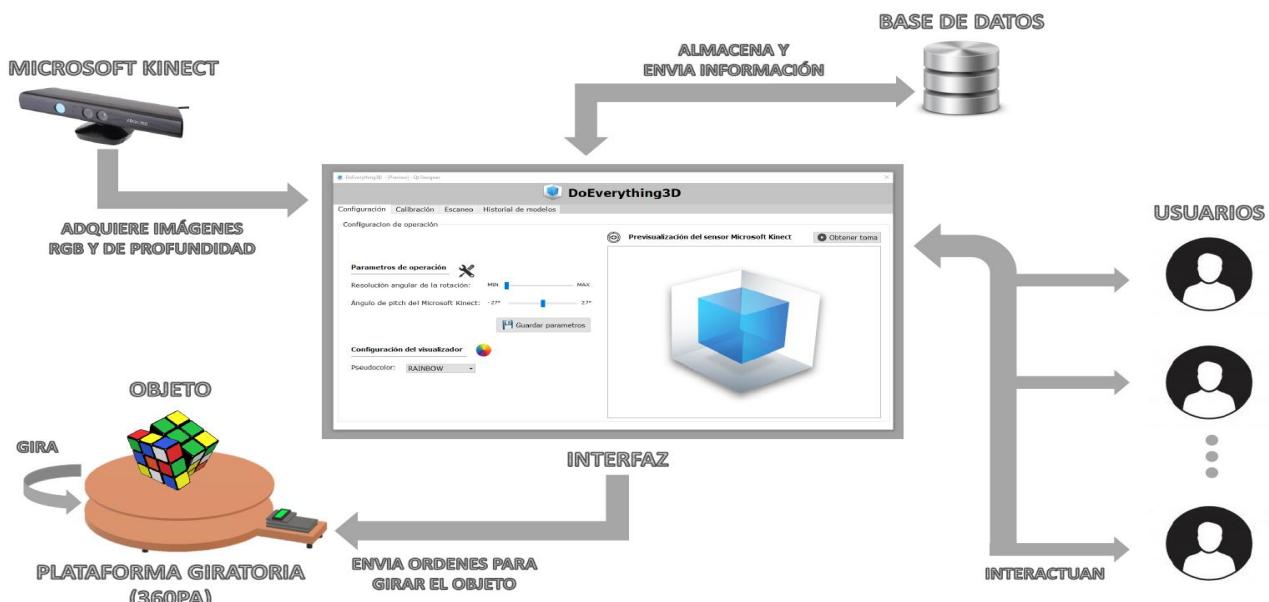
#### 1.3.1. Objetivo general

Desarrollar un aplicativo con interfaz de fácil uso para la extracción automatizada de un modelo 3D de objetos usando un sensor Microsoft Kinect.

#### 1.3.2. Objetivo específico

1. Realizar el estado del arte acerca de la obtención de un modelo 3D de objetos usando el sensor Microsoft Kinect.
2. Diseñar una interfaz intuitiva que permita ejercer el control automatizado de la plataforma para el modelamiento 3D del objeto.
3. Desarrollar una plataforma motora para realizar la captura de las imágenes mediante las cuales se va a realizar la reconstrucción 3D.
4. Implementar la interfaz de control automatizada en conjunto con un algoritmo que permita extraer el modelo 3D del objeto a reconstruir a partir de las imágenes capturadas por el sensor Microsoft Kinect.
5. Realizar la validación funcional del sistema desarrollado.

### 1.4. SOLUCIÓN IMPLEMENTADA



**Figura 1.2** Diagrama conceptual de la propuesta presentada (DoEverything3D).

La **figura 1.2** es un acercamiento conceptual a la propuesta presentada en donde se puede observar el papel que ejerce cada módulo y la asociación existente entre ellos. A continuación, se explica brevemente la propuesta:

Se implementó un sistema con interfaz intuitiva para la extracción modelo 3D de objetos usando un sensor Microsoft Kinect a la cual se le da el nombre de "DOEVERYTHING3D". DOEVERYTHING3D consta principalmente de 4 grandes módulos:

- ✓ **Módulo de adquisición:** el módulo es conformado únicamente por el sensor Microsoft Kinect, encargado de realizar la adquisición de imágenes de profundidad sobre el objetivo a escanear. La orden de capturar estas imágenes de profundidad es enviada desde el modulo del aplicativo para luego proceder a enviar a este mismo modulo las imágenes capturadas.
- ✓ **Módulo de rotación (360PA - 360 PLATFORM ASSITANT):** el módulo lo conforma una plataforma giratoria basada en un microcontrolador, capaz de monitorizar la resolución angular de sus movimientos. El papel principal de esta plataforma es realizar la rotación controlada del objeto a modelar, con el objetivo de obtener información de profundidad sobre todas las caras del objeto mediante el sensor Microsoft Kinect. La ordenes que indican cuando la plataforma debe girar y con qué resolución angular son enviadas desde el modulo del aplicativo.
- ✓ **Módulo de aplicativo:** el módulo del aplicativo se desarrolló bajo el lenguaje de programación Python y es la integración de múltiples API's (PCL, Open3D, Pymesh, PyQt5, mysqlConnector, pySerial, numpy y pyqtgraph) conjunto a una interfaz gráfica intuitiva, que trabajan de manera asociada para generación de un modelo 3D a partir de las imágenes de profundidad recibidas del módulo de adquisición. El modulo también tiene permisos para acceder y modificar la información asociada a cada usuario, alojada en el módulo de persistencia, además de gestionar la calibración de la cámara de profundidad del sensor Microsoft Kinect y controlar de manera sincrónica el módulo de adquisición y rotación.
- ✓ **Módulo de persistencia:** el módulo es desarrollado bajo tablas SQL, las cuales se encargan de almacenar información asociada a cada usuario como son los parámetros de configuración y sus trabajos realizados. El módulo del aplicativo tiene permisos para pedir y modificar información asociada a un usuario específico.

## 1.5. ESTRUCTURA DEL DOCUMENTO

A lo largo del documento, se tienen diferentes capítulos que recorren todos los objetivos específicos planteados, explicando el desarrollo y el cumplimiento de cada uno de ellos.

Primeramente, en el capítulo 2 llamado estado del arte y marco teórico se aborda el objetivo específico número 1, en el figuran conceptos de vital importancia para el entendimiento de los tecnicismos empleados a lo largo del documento, que buscan contextualizar al lector en el marco teórico de la visión artificial orientada a la reconstrucción tridimensional, sus aplicaciones, prestaciones y funcionalidades, así como las técnicas y mecanismos de programación utilizados. Además, se estudian diferentes sistemas y propuestas desarrolladas para la extracción de modelos tridimensionales, en donde se extrae una metodología que es común en la mayoría de propuestas, para luego ser implementada en nuestra solución.

Luego, en el capítulo 3 se desarrollan el objetivo específico 3, en donde se presenta el diseño y el desarrollo del hardware y del firmware de la plataforma motora, que se encargará de rotar el objetivo a modelar sobre todas sus caras.

Más adelante, desarrollando los objetivos específicos 2 y 4, se presentan las etapas de diseño y desarrollo del aplicativo para la extracción de modelos 3D de objetos, además se presenta la solución propuesta para adaptar y dar cumplimiento a las funcionalidades especificadas para el aplicativo.

Como última parte, en el capítulo 4 se desarrollan las pruebas al sistema, se presentan las conclusiones del proyecto y se plantean los posibles trabajos futuros.

# CAPÍTULO 2

## 2. ESTADO DEL ARTE Y MARCO TEÓRICO

---

- 2.1. Introducción
- 2.2. Estado del arte
  - 2.2.1. Análisis de las metodologías de generación de modelos 3D
- 2.3. Marco teórico
  - 2.3.1. Escáner 3D
  - 2.3.2. Microsoft Kinect
- 2.4. Transformación de imagen de profundidad a nubes de puntos 3D
  - 2.4.1. Calculo del mapa de profundidad
  - 2.4.2. Modelo de cámara Pinhole
- 2.5. Nubes de puntos 3D
- 2.6. Descriptores de nubes de puntos 3D
  - 2.6.1. Vóxeles
  - 2.6.2. Normales de superficie
  - 2.6.3. Descriptor PHF
  - 2.6.4. Descriptor FPHF
- 2.6. Registro de nubes de puntos 3D
  - 2.6.1. Transformación 3D
  - 2.6.2. Algoritmo SAC-IA
  - 2.6.3. Algoritmo ICP
- 2.7. Alineación global
- 2.8. Segmentación de plano tierra
- 2.9. Reducción de ruido
  - 2.9.1. Reducción de ruido mediante algoritmo MLS
- 2.10. Generación de superficie
  - 2.10.1. Triangulación de nubes de puntos 2D
  - 2.10.1. Triangulación de nubes de puntos 3D
- 2.11. Observaciones finales

---

### 2.1. INTRODUCCIÓN



Este capítulo, busca enmarcar la propuesta DOEVERYTHING3D, respecto a los demás sistemas y alternativas de extracción de modelos 3D existentes, realizando una investigación sobre las especificaciones, algoritmos y metodología que emplea cada una de éstas.

Posteriormente, en el marco teórico se hace una revisión de conceptos importantes, tales como: el funcionamiento de un cámara de profundidad, las nubes de puntos, la obtención de estas nubes a partir de imágenes de profundidad, algunas las operaciones espaciales que se pueden realizar a estas nubes, descriptores de puntos, el registro de un par de nube de puntos con algunas formas para llevarlo a cabo, la reducción de ruido y segmentación de plano tierra en nubes de puntos y como último concepto la generación de superficie

a partir de nubes de puntos. Todo esto, con el fin que el lector pueda tener una lectura fluida comprendiendo completamente cada uno de los capítulos del documento. El marco teórico, se hace un recorrido por la mayoría de sensores, técnicas y algoritmos más utilizados en la visión artificial para la reconstrucción 3D.

## 2.2. ESTADO DEL ARTE

Para la implementación de un sistema de reconstrucción 3D existen gran variedad de soluciones en las que se utilizan diferentes sensores y técnicas. Escáneres láser, escáneres de luz estructurada y escáneres de visión estéreo son algunos de ellos. Algunas de las soluciones y métodos relevantes para nuestro trabajo se discuten brevemente en esta sección y se encuentran condensados en la Tabla 2.1 en donde se exponen sus principales características.

**Tabla 2.1:** Antecedentes investigados relacionados con la temática del problema a resolver.

REFERENCIA	DISPOSITIVO DE ADQUISICIÓN UTILIZADO.	TECNOLOGÍA DE ADQUISICIÓN.	ALGORITMOS IMPLEMENTADOS EN LA GENERACIÓN DEL MODELO.	OBJETIVO A ESCANEAR	PLATAFORMA
[5]	Microsoft Kinect v1.	Luz estructurada.	Segmentación mediante color, registro de dos etapas (RANSAC e ICP), alineación global, reducción de ruido.	Objeto.	PC.
[11]	Microsoft Kinect v1.	Luz estructurada.	Registro de dos etapas (Registro no rígido e ICP), alineación global, reducción de ruido.	Cuerpo Humano.	PC.
[12]	Cámara digital convencional.	Luz estructurada.	Calibración de la cámara RGB, Registro mediante triangulación, reducción de ruido.	Objeto.	PC.
[13]	Microsoft Kinect v1.	Luz estructurada, visión estéreo.	Registro de dos etapas (disparidad entre imágenes e ICP), reducción de ruido, generación de superficie mediante triangulación.	Ambiente.	PC.
[14]	Microsoft Kinect v1.	Luz estructurada.	Segmentación mediante color, ICP, reducción de ruido.	Objeto.	PC.
[15]	Omrom ZG.	Triangulación Laser.	Registro de dos etapas(RANSAC e ICP), CSG.	Objeto.	PC.
[16]	Manufactura Propia.	Triangulación Laser.	ICP y Superficies de Bézier.	Objeto.	PC.
[17]	Cámaras RGB convencionales.	Estéreo Visión.	Disparidad entre imágenes, SGM con MI, ICP.	Ambiente.	PC.
[18]	Microsoft Kinect v2.	Luz estructurada.	ICP, reducción de ruido.	Ambiente.	PC.
[19]	Microsoft Kinect v2.	Luz estructurada.	Segmentación, ICP, generación de superficie mediante triangulación, reducción de ruido.	Objeto.	PC.
[20]	Microsoft Kinect v2.	Luz estructurada.	Segmentación mediante color, ICP, alineación Global, reducción de ruido.	Objeto.	PC.
[21]	Microsoft Kinect v1.	Luz estructurada.	Segmentación mediante color , registro de superficie SDF-2-SDF, alineación global, triangulación de puntos.	Objeto.	PC.
[22]	Microsoft Kinect v1.	Luz estructurada.	Segmentación mediante color, uso de librerías de OpenGL para el registro y dibujado de la superficie del modelo.	Parte inferior del cuerpo humano.	PC.
[23]	Microsoft Kinect v2.	Luz estructurada.	Segmentación mediante color, ICP y alineación global.	Objeto.	PC.

[24]	Microsoft Kinect v1.	Luz estructurada.	Segmentación mediante color, registro de dos etapas (Técnica basada en teselación e ICP), alineación global.	Parte superior del cuerpo humano.	PC.
------	----------------------	-------------------	--	-----------------------------------	-----

Observando la Tabla 2.1, en [5] se presenta el modelamiento 3D de objetos a partir del sensor Microsoft Kinect. La contribución principal de este trabajo consiste en la investigación que se realiza sobre el uso de distintos tipos de sensores con tecnología de luz estructurada entre ellos el Kinect, analizando las ventajas y desventajas que presentan cada una de ellas para este aplicativo. También se realizan comparaciones entre diferentes algoritmos que dan solución a algunos de los principales desafíos cuando se realizan estos modelos a partir de imágenes de profundidad, como lo son: la segmentación de objetos, el registro fino y la alineación global, además de realizar la implementación del algoritmo ICP (Iterative Closest Point) conjunto con el algoritmo RANSAC (Random Sample Consensus) para dar solución al problema de registro.

Otra solución interesante es el que se presenta en [11], en donde se utilizan múltiples sensores Microsoft Kinect para generar un modelo 3D de un cuerpo humano completo. La principal contribución de este trabajo consiste en la investigación e implementación de algoritmos de alineación global no rígida en conjunto al tradicional ICP. Técnicas que se utiliza como solución al problema de registro para los datos escaneados. Además de implementar un algoritmo de alineación global con el objetivo de atenuar cualquier error de alineación ocurrido en el registro.

En [12], se muestra un escáner 3D de objetos con plataforma giratoria bajo la tecnología de luz estructurada. El principal aporte de este documento son los procedimientos usados para la calibración de la plataforma giratoria con el dispositivo de adquisición, además de la implementación del algoritmo de registro mediante triangulación, el cual cumple la función de converger diferentes nubes de puntos obtenidas al escanear el objeto bajo distintos ángulos generando así un modelo 3D parcial.

En [13], se expone un proceso de reconstrucción de escenas 3D utilizando un par de sensores Microsoft Kinect. El principal aporte de este trabajo es la implementación del registro de puntos mediante la disparidad existente entre ambas imágenes adquiridas, dando como resultado un modelo de nube de puntos unificado del ambiente.

En [14] se realiza un sistema de reconstrucción en 3D utilizando el sensor Microsoft Kinect. Este sistema implementado puede reconstruir modelos 3D de alto nivel y geométricamente precisos en tiempo real con características de texturas. El principal aporte de esta publicación es la implementación de técnicas de segmentación, registro mediante ICP, alineación global, generación de la superficie del objeto y reducción de ruido que son imprescindibles para la realización del modelo. Todas las técnicas mencionadas están incluidas en la API de "3D reconstruction" lanzada por Microsoft en el SDK del dispositivo Kinect.

En [15] se presenta el diseño y construcción de un escáner basado en un sensor láser 2D comercial de alta precisión. La implementación de un registro de dos etapas mediante los algoritmos RANSAC e ICP, los cuales hacen de este trabajo un gran aporte para la solución del problema de registro de nubes de puntos.

Otra alternativa es el trabajo presentado en [16], el cual presenta el diseño e implementación de un escáner 3D de objetos bajo un sensor láser 2D con cámara de manufactura propia. En el documento se trabaja la caracterización y calibración de su sensor, además de la implementación de los algoritmos ICP y NURBS, encargados de la del registro y la generación de superficie del respectivamente.

En [17] se presenta el diseño e implementación de un sistema de reconstrucción 3D de escenas usando un par de cámaras RGB. En el trabajo se hace uso del algoritmo SGM (Semi-global Matching) que utiliza la información mutua (MI) entre las dos imágenes para luego evaluar la disparidad entre ambas y generar nubes de puntos del medio fotografiado. Por último, se realiza el registro de diversas nubes de puntos del objetivo a escáner mediante el algoritmo ICP.

En [18] se realiza un sistema de reconstrucción 3D mediante el uso del sensor Microsoft Kinect v2. En primer lugar, este documento presenta los parámetros técnicos y el principio de medición del Kinect v2, luego analiza las ventajas y desventajas de Kinect v2 respecto a la primera generación de Kinect (Kinect v1). Luego se analizan las características del ruido de los datos de profundidad que arroja el sensor Kinect v2 y se presenta un diseño de un método de eliminación de desniveles para datos de profundidad para después implementarlo al sistema de reconstrucción 3D. Por último, se desarrolla el sistema de reconstrucción 3D propuesto con ayuda del algoritmo ICP que se encarga del registro de las nubes de puntos que arroja el sensor Kinect.

En [19] se realiza el diseño e implementación de un sistema de reconstrucción 3D usando el sensor Microsoft Kinect. El principal de aporte de este trabajo es la implementación de técnicas de: segmentación basada en color, registro mediante el algoritmo ICP, generación de la superficie del modelo mediante triangulación y reducción de ruido.

En [20] se describe e implementa una metodología para la reconstrucción 3D utilizando Kinect v2. Esta metodología hace uso de algoritmos de segmentación mediante color, registro de nubes de puntos 3D bajo ICP, alineación global y reducción de ruido del modelo. Además, se analizan las diferentes variantes de los algoritmos ICP presentando las ventajas de cada uno de ellos.

En [21] se aborda el tema de la reconstrucción 3D de objetos mediante cámaras RGB-D en este caso el dispositivo Microsoft Kinect. La principal contribución de este trabajo es el diseño e implementación de un esquema de registro bajo SDFs (Signed Distance Field) que remplaza al clásico algoritmo de registro ICP, además de implementar técnicas de segmentación, refinamiento global o alineación global y generación de superficie del modelo mediante triangulación.

En [22] se presenta un sistema de reconstrucción 3D para cuerpo humano basado en el dispositivo Microsoft Kinect. Su principal aporte es la implementación de técnicas correspondientes al registro y dibujado de la nube de puntos del modelo, las cuales están soportadas por librerías públicas ya existentes en OpenGL.

En [23] se presenta una manera de abordar el problema de reconstrucción 3D mediante el dispositivo Microsoft Kinect en este caso la versión 2. El procedimiento que realiza el sistema propuesto en el documento sigue la siguiente metodología: segmentación de la información de profundidad con ayuda de la cámara RGB (segmentación por color), registro de las nubes de puntos con el uso del algoritmo ICP, alineación global con el objetivo de minimizar los errores en el modelo provenientes de la etapa de registro.

Finalmente, en [24] se presenta el desarrollo de un sistema de reconstrucción 3D de cuerpo humano (parte superior) usando el dispositivo Microsoft Kinect. La principal contribución de este trabajo es la técnica utilizada para registro grueso basada en teselación la cual usa información de color y profundidad además de proponer dos nuevas métricas para evaluar el registro de nubes de puntos. También implementa algoritmos de segmentación mediante color y registro fino bajo el algoritmo ICP.

### *2.2.1. Análisis de las metodologías de generación de modelos 3D*

Luego de analizar los procesos mencionados para el modelamiento 3D, se logró identificar dos patrones de metodología presentes en la mayoría de ellos, los cuales se condensan en la figura 2.1. La diferencia entre ambas metodologías es básicamente, que una suple la necesidad de tener información RGB en el modelo 3D y la otra no, por lo que esta diferencia afecta directamente el momento en que se aplica la etapa de segmentación. Cabe recalcar que cada sistema investigado usa algoritmos y técnicas diferentes para dar solución a cada una de las etapas.

El diagrama de la figura 2.1 explica la metodología de los sistemas que realizan modelamiento 3D tanto como los que asocian información de color al modelo como lo que no lo hacen.

Como primera etapa se encuentra la captura de imágenes de profundidad al objeto de interés sobre diferentes vistas, si el sistema también obtiene información RGB del objeto ambas cámaras (profundidad y RGB) se han de haber calibrado entre ellas, con el fin de asociar los datos adquiridos de profundidad y RGB. Posteriormente, con la finalidad de obtener información espacial 3D (nube puntos 3D) únicamente del objeto, se procede a realizar la transición de imágenes de profundidad a nubes de puntos 3D de todas las vistas capturadas del objeto de interés, para luego eliminar el plano tierra presente en cada una de las nubes (en el caso de no tener información RGB del objeto). En cambio, si se posee información RGB el proceso de obtener información espacial 3D (nube puntos 3D) únicamente del objeto se resume en, eliminar toda la información que no pertenezca a el objeto de interés de las imágenes de profundidad mediante segmentación por color de su imagen RGB asociada, para luego realizar la transición a nubes de puntos de las imágenes de profundidad ya segmentadas.



**Figura 2.1.** Diagrama de flujo en donde se condensan los patrones de metodología identificados de los trabajos analizados.

Continuando con la siguiente etapa correspondiente al registro, las nubes de puntos 3D segmentadas de diferentes vistas se registran juntas con el fin de obtener una nube de puntos 3D combinada que represente completamente el objeto sobre las vistas tomadas. Normalmente, el proceso de registro de las diferentes nubes puntos 3D logra acumular una cantidad de error considerable lo cual impide que la última vista del objeto no logre alinearse correctamente con la nube de puntos 3D de la primera vista. Con la finalidad de solucionar este problema se realiza la etapa denominada alineación global del modelo.

Con el objetivo de eliminar datos erróneos debido al ruido, se realiza la etapa de reducción de ruido a toda la nube 3D combinada. Y como último paso, se realiza la última etapa que consiste en la representación de superficie o malla poligonal sobre la nube de puntos 3D para obtener el modelo final del objeto.

Debido a que el sistema que se implementará prescindirá de información RGB en el modelo 3D obtenido, se utilizará la metodología descrita la cual realiza la etapa de segmentación mediante la eliminación del plano tierra. En la siguiente sección de marco teórico, se ampliará los detalles de cada etapa como la técnica que se usará en este proyecto en cada una de estas.

### 2.3. MARCO TEÓRICO



Para la digitalización de objetos o ambientes del mundo real es necesario de dispositivos que puedan capturar la forma superficial de estos y devolver un conjunto de datos que puedan describir tal superficie. Estos dispositivos se denominan escáneres 3D y el conjunto de dato que devuelven se les denomina nube de puntos 3D [2]. Luego para generar una reconstrucción 3D completa del objetivo se necesita relacionar distintas nubes de puntos capturadas sobre el objetivo bajo diferentes ángulos, este procedimiento se le denomina registro [25].

Por último, se genera una superficie digital para el modelo 3D mediante diferentes técnicas de generación de polígonos [3]. A continuación, se explicarán con más detalles estos estos pasos mencionados para la generación de un modelo 3D a partir de datos de un escáner 3D, además se introducirá a la metodología RUP la cual será utilizada para para el desarrollo del proyecto.

### 2.3.1. Escáner 3D

Un escáner 3D es un dispositivo que recolecta características geométricas y espaciales de objetos o ambientes reales con el fin de generar un modelo digital tridimensional a partir de dichas características [1]. Un escáner 3D puede ser catalogado en dos grandes grupos en función si hay contacto o no con el objetivo a escanear. Los escáneres 3D sin contacto se subdividen en escáneres pasivos y escáneres activos dependiendo de la tecnología que usen [26]. A continuación, se discutirán sobre las dos clasificaciones existentes en los escáneres 3D.

#### ✓ Escáneres de contacto

Estos escáneres funcionan mediante la exploración que realiza un elemento llamado palpador que recorre la superficie del objetivo a escanear [26]. Mediante un conjunto de sensores internos que estima las coordenadas espaciales del palpador, estas luego se usan para generar el modelo 3D del objetivo [26].

Una de sus principales ventajas es la precisión conseguida del modelo 3D extraído, pero su mayor desventaja es que requiere el contacto físico con el objetivo para ser escaneado esto puede conllevar a la modificación o el daño del objetivo [26].

#### ✓ Escáneres de no contacto

A diferencia de los escáneres de contacto estos tienen la ventaja de no requerir contacto físico con el objetivo para ser escaneado lo que garantizada la integridad de este [26]. Estos escáneres se subdividen en dos grupos según si emiten o no algún tipo de radiación para lograr capturar datos: los escáneres de no contacto activos y pasivos.

Primeramente, se discutirá sobre los escáneres 3D de no contacto activos. Este tipo de escáner emite algún tipo de radiación para analizar su retorno capturando la geometría de un objeto o ambiente [26]. A continuación, se mencionan algunas de las principales técnicas que usan los escáneres activos sin contacto:

- ✓ *Tiempo de vuelo*: este tipo de escáner 3D estima la distancia entre el dispositivo y el objetivo mediante la toma del tiempo de ida y vuelta de un pulso de luz de frecuencia conocida [27]. Estos dispositivos se caracterizan por su alta precisión (milésimas de milímetro) [27].
- ✓ *Triangulación*: este escáner 3D usa un haz de luz láser y un detector de posicionamiento, la luz es emitida y una porción de ella se refleja en el detector formando un triángulo entre la fuente de luz, el objetivo y el detector de posicionamiento. La distancia es obtenida a partir de la diferencia en el detector de posicionamiento ya que los ángulos que se forman entre ellos son conocidos [28]. Se caracterizan por tener una precisión bastante elevada (milésimas de milímetro) [28], aunque una de sus principales limitaciones es su alcance máximo el cual ronda en los 20 a 30 cm entre el dispositivo y el objetivo.
- ✓ *Luz estructurada*: este tipo de escáner 3D emite un patrón de luz sobre el objetivo y analiza la deformación del patrón producida por la escena para determinar la distancia del objetivo. Su principal ventaja es la velocidad [29]. En vez de registrar un punto a la vez, se registran una gran cantidad de estos, lo que a su vez atenúa el problema de la deformación por el movimiento del objetivo.
- ✓ *Diferencia de fase*: este escáner 3D mide la diferencia de fase entre la luz que emite y la recibida, luego utilizar esta lectura para estimar la distancia entre el dispositivo y el objetivo [30]. Estos dispositivos presentan una resolución media la cual ronda los 2 mm por cada 25 m [30]. Su principal ventaja es su alta velocidad de adquisición, consiguiendo los modelos actuales velocidades que oscilan entre los 100000 y 1 millón de puntos por segundo [30].

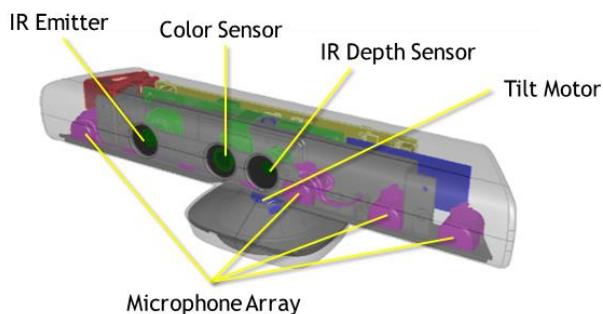
Ahora es el turno de discutir sobre los escáneres 3D de no contacto pasivos. Estos tipos de escáneres 3D no emiten ninguna clase de radiación por sí mismos, en lugar de ellos tratan de capturar la radiación reflejada del ambiente [26]. A continuación, se mencionan algunas de las técnicas principales que usan los escáneres pasivos sin contacto:

- ✓ *Estereoscópicos*: estos dispositivos utilizan dos cámaras levemente separadas apuntando hacia el mismo objetivo [17]. Analizando las diferencias leves entre las imágenes vistas por cada cámara, es posible determinar la distancia en capada punto en las imágenes. Este método es basado en la visión estereoscópica humana.

- ✓ *Detección de silueta:* estos tipos de escáner 3D usan bocetos creados de la sucesión de fotografías alrededor del objetivo contra un fondo muy bien contrastado [26]. Estas siluetas luego sirven para formar la aproximación visual de la superficie del objetivo. Una limitación de este tipo de escáner son los errores obtenidos en el modelo 3D al escanear objetos que presenten concavidades.

### 2.3.2. Microsoft Kinect

El Microsoft Kinect que se muestra en la Figura 2.2 es un sensor de movimiento PrimeSense incorporado en un dispositivo de juego para Xbox 360 y que entra en la categoría de escáneres 3D de no contacto activos que funcionan mediante luz estructurada. El dispositivo contiene una cámara RGB, un emisor infrarrojo que proyecta una cuadrícula invisible al ojo humano, un sensor de profundidad infrarrojo, micrófonos y acelerómetros. El sensor de profundidad usa el patrón proyectado para extraer la información de profundidad de la escena. Las imágenes de profundidad tienen una resolución de 640x480 y las imágenes en color se capturan con una resolución de 1280x960 [31]. La cámara RGB es capaz de transmitir imágenes de 640x480 a 30 fotogramas por segundo. El dispositivo se puede conectar a una PC mediante la interfaz USB y el SDK proporcionado por Microsoft [31].

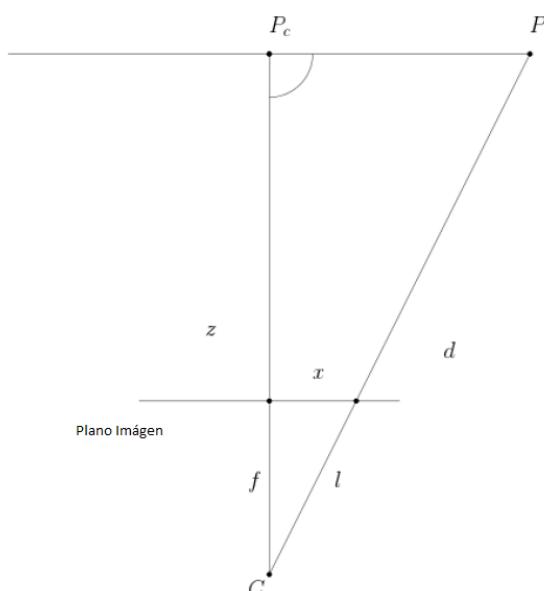


**Figura 2.2.** Disposición de los sensores en el Microsoft Kinect [32].

## 2.4. TRANSFORMACIÓN DE IMAGEN DE PROFUNDIDAD A NUBES DE PUNTOS 3D

El sensor basado en luz estructurada descrito en la sección anterior proporciona una imagen de profundidad donde cada píxel es una medida de distancia desde el centro de la cámara hacia el objetivo donde se apunta. Para modelar el espacio real y los objetos, se desean coordenadas 3D para los puntos que representan al objeto. Esta conversión de imagen de profundidad a nube 3D de puntos se puede obtener utilizando el modelo estándar de cámara Pinhole [33]. Primero, se presenta el cálculo del mapa de profundidad desde el mapa de distancia en donde se obtiene la coordenada Z de los puntos de la nube 3D y luego se discute sobre el modelo de cámara pinhole en donde se obtienen las coordenadas faltantes de la nube, X y Y.

### 2.4.1. Cálculo del mapa de profundidad



**Figura 2.3.** Calculo de profundidad desde la información de distancia.

La figura 2.3 es una representación de una cámara enfocando un punto denominado  $P_c$ , por lo tanto si queremos determinar la distancia desde la cámara al punto denominada  $Z$  se requiere entrar al siguiente razonamiento: dado el punto  $P$ , el centro de la cámara  $d$ , la distancia focal  $f$  y la distancia desde el punto principal al punto  $P$  en el plano de la imagen  $x$ , la distancia  $z$  desde el centro de la cámara  $C$  al punto  $P_c$  se puede calcular. Mediante la ecuación 2.1. se puede obtener la distancia de la proyección  $C$  a  $P$  en el plano de la imagen.

$$l = \sqrt{f^2 + x^2} \quad (2.1)$$

Entonces, se tiene lo que se puede observar en la figura 2.3. Y si se hace uso de la propiedad de semejanza de triángulos se obtiene la ecuación 2.2.

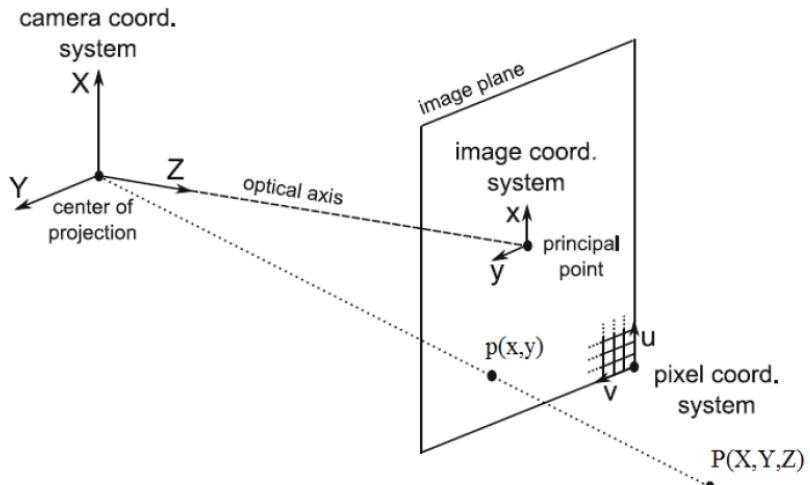
$$\frac{z}{d} = \frac{f}{l} \quad (2.2)$$

Combinando la ecuación 2.1 y 2.2 se obtiene la ecuación 2.3 que explica cómo se obtiene la coordenada  $Z$  en una nube de puntos a partir de una imagen de profundidad.

$$z = d \frac{f}{l} = d \frac{f}{\sqrt{f^2+x^2}} \quad (2.3)$$

El sensor Microsoft Kinect emite los valores de profundidad que ya están en el formato de mapa de profundidad. Esto significa que, en lugar de la distancia del centro de la cámara al punto, cada pixel representa un valor de la distancia al plano que tiene al punto real correspondiente, y este plano es perpendicular al eje principal de la cámara.

#### 2.4.2. Modelo de cámara Pinhole



**Figura 2.4.** Modelo de cámara Pinhole [33].

El modelo Pinhole plantea que el centro del sistema de coordenadas esté en el centro de la cámara,  $f$  la distancia entre el centro de proyección y el plano de la imagen (distancia focal de la cámara), línea que comienza desde el centro de proyección y perpendicular al plano de la imagen se denomina eje principal como se muestra en la figura 2.4. El punto principal es la intersección del eje principal y el plano de la imagen. Dado el punto  $P (X, Y, Z)$  en el espacio 3D su punto proyección en el plano coordenadas la imagen hacia el centro de proyección se denomina  $p (x, y)$ . Por la propiedad de triángulos similares, las coordenadas del plano de la imagen se pueden escribir como se puede ver en la ecuación 2.4.

$$x = f \frac{X}{Z-f} \quad , \quad y = f \frac{Y}{Z-f} \quad (2.4)$$

Ahora  $x$  y  $y$  describen las coordenadas del mundo real y  $X$  y  $Y$  son las coordenadas imágenes en píxeles. De 2.4. se pueden despejar las coordenadas reales faltantes para finalizar la conversión de una imagen de profundidad a nubes de puntos, las coordenadas del mundo real despejadas se pueden escribir como se ve en la ecuación 2.5.

$$X = \frac{x(z-f)}{f} \quad , \quad Y = \frac{y(z-f)}{f} \quad (2.5)$$

## 2.5. NUBES DE PUNTOS 3D

Los archivos de nubes de puntos son un conjunto de vértices en un sistema de coordenadas tridimensionales [2]. Estos vértices en la mayoría de los casos son definidos en las coordenadas (X, Y, Z) y son empleados por la descripción de la superficie exterior de un objetivo. Los archivos de nubes de puntos son generados por escáneres 3D. Estos dispositivos pueden generar de manera automatizada una gran cantidad de puntos de la superficie del objetivo en análisis y ponerlo en un archivo digital de datos. En la figura 2.5, se muestra una nube de puntos de un cilindro escaneado desde una vista superior.



**Figura 2.5.** Nube de puntos de un cilindro al cual es escaneo desde una vista superior.

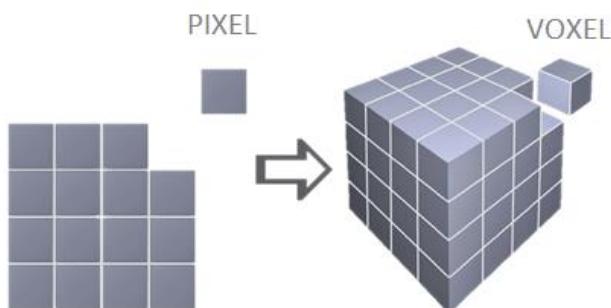
## 2.6. DESCRIPTORES DE NUBES DE PUNTOS 3D

La determinación de la similitud entre las formas 3D es una tarea fundamental en el reconocimiento, la recuperación, la agrupación, la clasificación basados en formas y además es una muy buena ayuda en el registro de nubes de puntos aumentando la eficiencia de este proceso [34]. Los descriptores 3D, en resumen, hacen la labor de extraer características de una nube de puntos 3D a partir de todos los puntos de la nube (nuestro caso) o de ciertos puntos a los que se les denomina "puntos clave" que fundamentalmente agrupan las características más relevantes presentes en la nube, en nuestro caso es importante obtener la característica de forma, por lo que discutiremos sobre el descriptor 3D FPFH (Fast Point Feature Histograms) que retorna propiedades geométricas y la curvatura sobre la nube analizada.

Debido a la gran cantidad de puntos que puede contener una nube 3D (miles o millones de puntos), es conveniente reducir la cantidad de puntos de la nube a analizar. Con el objetivo de disminuir el tiempo de computo que requiere el descriptor y eliminar información redundante o innecesaria, en la siguiente sección hablaremos de los Vóxeles que cumplen dicha función de particionamiento espacial y reducción de información. Luego, se discutirá sobre las normales de superficie, para luego si introducir el descriptor 3D PFH (Point Feature Histograms) y su optimización conocida como FPFH.

### 2.6.1. Vóxeles

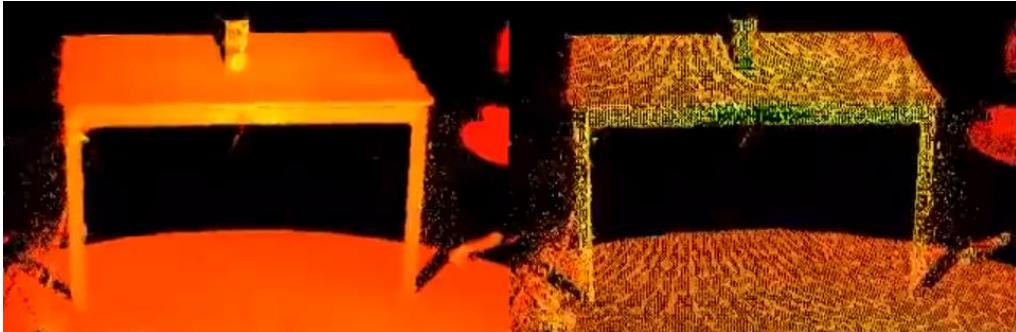
Un voxel [35], que es un acrónimo de las palabras volumetric y pixel, es un elemento de volumen, que representa un valor en una cuadrícula regular en un espacio tridimensional. Esto es análogo a un píxel, que representa datos de imágenes 2D. Al igual que con los píxeles, los voxels generalmente no contienen su posición en el espacio (sus coordenadas), sino que se infieren en función de su posición relativa a otros voxels, es decir, su posición en la estructura de datos que conforma una sola imagen de volumen. En la figura 2.6 se aprecia una comparación geométrica entre un pixel y un voxel.



**Figura 2.6.** Comparación geométrica entre un pixel (izquierda) y un voxel (derecha) [36].

Como primer paso para la reducción de puntos en una nube 3D a partir de voxels es la elección del tamaño de la arista  $l$  que tendrán los voxels que dividirán la nube. Luego se realiza la división del volumen total que ocupa la nube 3D en voxels de volumen  $l^3$ . Entonces, los puntos que estén contenidos en cada voxel

serán aproximados a un único punto mediante su centroide, también es posible aproximar los puntos al centro del voxel, pero la primera solución del centroide da como respuesta una nube de reducida más fiable a la original. En la figura 2.7 se aprecia una nube 3D que ha sido reducida en cantidad de puntos mediante voxels.

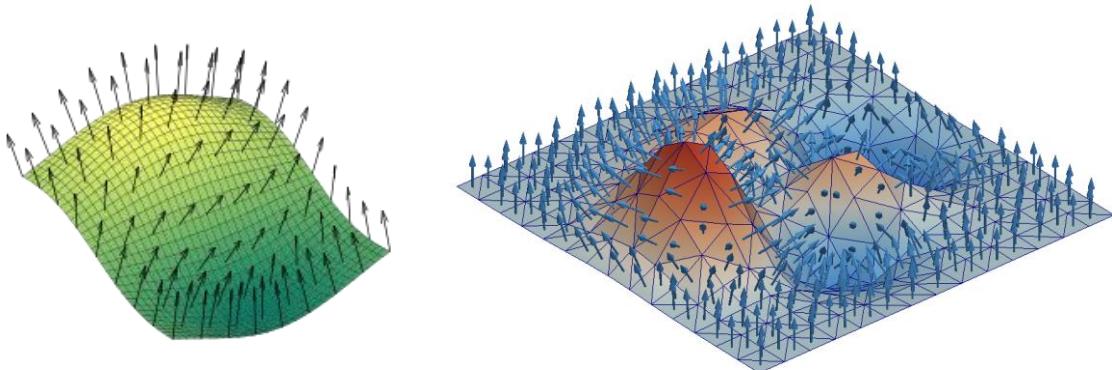


**Figura 2.7.** Comparación nube de puntos original (izquierda) y el resultado de su reducción mediante voxels (derecha) [37].

Ahora, se introducirá el concepto de normales de superficie debido a su importancia para la comprensión del funcionamiento de los descriptores 3D de forma.

### 2.6.2. Normales de superficie

El vector normal (en un ámbito 3D) a una superficie en un punto determinado, es el vector perpendicular al plano tangente en ese punto de la superficie, esto es representado en la Figura 2.8. Esta característica describe la orientación de cada una de las vecindades locales a el punto de interés. El vector normal es una medida de la curvatura y tiene aplicaciones en la construcción de descriptores 3D, identificación de planos, además en la creación de las gráficas computacionales permitiendo la aplicación de niveles de intensidad de brillo según su distribución y orientación [34].



**Figura 2.8.** Vectores normales a una superficie 3D [38].

Para una superficie  $S$  determinada implícitamente satisfaciendo  $F(x, y, z) = 0$ , la normal puede ser estimada en cada punto sencillamente mediante el gradiente  $\nabla F(x, y, z)$  evaluado en ese punto. Para una superficie  $S$  determinada explícitamente  $F(x, y)$ , puede hallarse la normal transformándose a una determinación implícita  $F(x, y, z) = z - f(x, y) = 0$ , y utilizando el gradiente  $\nabla F(x, y, z)$ . Otra vía, es usando el gradiente de la forma explícita  $\nabla F(x, y)$  para hallar el de la implícita como  $\nabla F(x, y, z) = k^{\wedge} - \nabla f(x, y)$ . Donde  $k^{\wedge}$  es el vector unitario del eje Z.

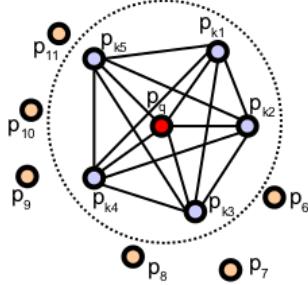
Ahora con los conceptos previos de voxels y normales de superficie ya claros, entraremos a discutir un tipo de descriptor 3D de forma con su versión optimizada, que es el que se empleara en este trabajo.

### 2.6.3. PHF

El descriptor PFH [34] (Point Feature Histograms), tiene como meta representar las propiedades geométricas y la curvatura de la vecindad de cada punto componente de la nube de puntos en tres dimensiones. Esto es logrado mediante la obtención de histogramas multidimensionales a partir de las relaciones entre las normales de superficie en cada punto, acumulado las características propias de dicha vecindad y generando una firma propia para la nube analizada. En pocas palabras, intenta capturar de la mejor manera posible las variaciones de la superficie de la nube teniendo en cuenta todas las interacciones

entre las direcciones de las normales estimadas, por lo tanto, el hiperespacio resultante depende de la calidad de las estimaciones normales de superficie en cada punto.

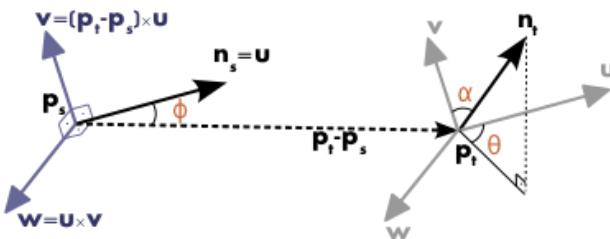
La figura 2.9 presenta un diagrama de región de influencia del cálculo de PFH para un punto de consulta  $p_q$ , marcado con rojo y ubicado en el medio de un círculo (esfera en 3D) con radio  $r$ , y todos sus  $k$  vecinos (puntos con distancias menores que el radio  $r$ ) que están completamente interconectados en una malla. El descriptor PFH final se calcula como un histograma de relaciones entre todos los pares de puntos en el vecindario, y por lo tanto tiene una complejidad computacional de  $O(k^2)$ .



**Figura 2.9.** Diagrama de región de influencia del cálculo de PFH para un punto de consulta  $p_q$  [39].

Para calcular la diferencia relativa entre dos puntos  $p_s$  y  $p_t$  y sus normales asociadas  $n_s$  y  $n_t$ , definimos un marco de coordenadas fijo ( $u, v, w$ ) en uno de los puntos como se puede apreciar en la figura 2.10 y descrito matemáticamente por la ecuación 2.6.

$$\begin{cases} u = n_s \\ v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|^2} \\ w = u \times v \end{cases} \quad (2.6)$$



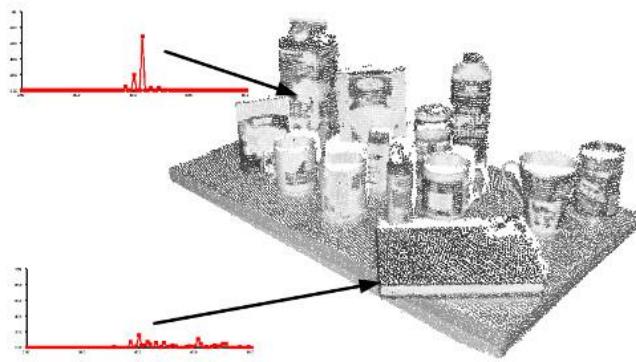
**Figura 2.10.** Marco referencial de coordenadas ( $u, v, w$ ) propuesto en donde se grafican dos puntos  $p_s$  y  $p_t$  y sus normales asociadas  $n_s$  y  $n_t$  [40].

Usando el marco ( $u, v, w$ ) anterior, la diferencia entre las dos normales  $n_s$  y  $n_t$  se puede expresar como un conjunto de características angulares como se puede apreciar en la ecuación 2.7.

$$\begin{cases} \alpha = v \cdot n_t \\ \phi = u \cdot \frac{(p_t - p_s)}{d} \\ \theta = \arctan(w \cdot n_t, u \cdot n_t) \end{cases} \quad (2.7)$$

Donde  $d$  es la distancia euclídea entre los dos puntos  $p_s$  y  $p_t$ ,  $d = \|p_t - p_s\|^2$ . El cuadruplete  $(\alpha, \phi, \theta, d)$  se calcula para cada par de dos puntos en k-vecindario, por lo tanto, se reducen los 12 valores ( $xyz$  y su información normal) de dos puntos y sus normales a 4.

Luego del cálculo de todos los cuadrupletes para cada punto, el conjunto de todos los cuadrupletes se agrupa en un histograma para crear la representación final de PFH para el punto de consulta. El proceso de agrupamiento divide el rango de valores de cada característica en  $b$  subdivisiones, y cuenta el número de ocurrencias en cada subintervalo. Finalmente es generado un descriptor total que contiene todas las combinaciones posibles de los sub-histogramas calculados para cada ángulo característico, obteniéndose un histograma de 125 contenedores. Este procedimiento es computacionalmente lento con una complejidad  $O(n * k^2)$  siendo  $n$  la cantidad de puntos presentes en la nube y  $k$  la cantidad de vecinos por vecindario. La figura 2.11 presenta ejemplos de representaciones de Histogramas de Puntos para diferentes puntos en una nube.



**Figura 2.11.** Ejemplos de representaciones de Histogramas de Puntos para diferentes puntos en una nube [41].

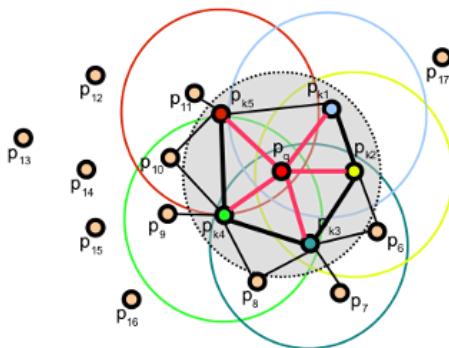
#### 2.6.4. FPFH

Este descriptor FPFH [34] (Fast Point Feature Histograms), presenta un proceso de optimización de la técnica basada en histogramas de características de puntos (PHF) con la finalidad de aplicarlo al procesamiento en tiempo real, reduce la complejidad a  $O(n * k)$  mediante la obtención de características entre el punto característico del descriptor local y los puntos integrantes de su  $k$ -vecindad, en vez de considerar todas las combinaciones de pares en esa vecindad como lo hace el método original.

El método se divide en 2 etapas, en primer lugar, realiza la versión simplificada ya mencionada, la cual es denominada SPFH, posteriormente para cada  $k$ -vecino se obtiene su respectiva relación SPFH y se realiza un promedio ponderado teniendo en cuenta la distancia de cada vecino respectivo hacia el punto de consulta  $p_q$  como se puede observar en la ecuación 2.8.

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k} \sum_{i=1}^k \frac{1}{\omega_k} \cdot SPFH(p_k) \quad (2.8)$$

donde el peso  $\omega_k$  representa una distancia entre el punto de consulta  $p_q$  y un punto vecino  $p_k$  en un espacio métrico dado, anotando así el  $(p_q, p_k)$  par, pero podría ser seleccionado como una medida diferente si es necesario. Para comprender la importancia de este esquema de ponderación, la figura 2.12 a continuación presenta el diagrama de la región de influencia para un conjunto  $k$ -vecindario centrado en  $p_q$ .



**Figura 2.12.** Diagrama de la región de influencia para un conjunto  $k$ -vecindario centrado en  $p_q$  [42].

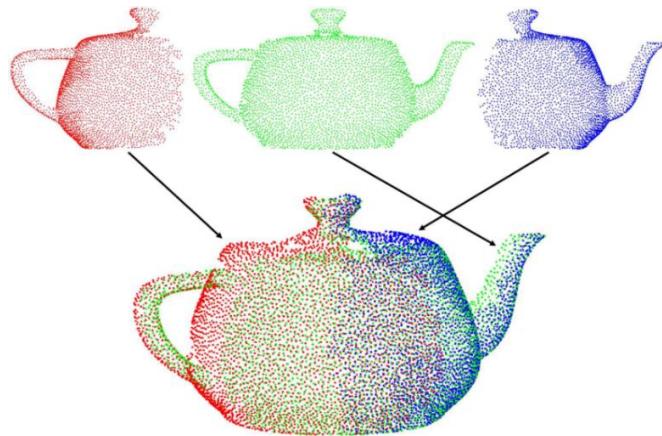
Por lo tanto, para un punto de consulta dado  $p_q$ , el algoritmo primero estima sus valores de SPFH creando pares entre sí mismo y sus vecinos (ilustrados con líneas rojas). Esto se repite para todos los puntos en el conjunto de datos, seguido por una nueva ponderación de los valores SPFH de  $p_q$  utilizando los valores SPFH de su  $p_k$  vecinos, creando así el FPFH para  $p_q$ . Las conexiones FPFH adicionales, resultantes debido al esquema de ponderación adicional, se muestran con líneas negras. Como muestra la figura 2.12, algunos de los pares de valores se contarán dos veces (marcados con líneas más gruesas en la figura).

Es importante resaltar las diferencias entre el método original y esta optimización, en primer lugar, existen regiones exteriores utilizadas para la estimación de características debido a la realización de un proceso simplificado de procesamiento para cada vecino, además se presenta la sobre inclusión de ciertas interconexiones por este mismo fenómeno, el descriptor final contiene 11 contenedores por cada ángulo concatenados en un descriptor final de 33 contenedores.

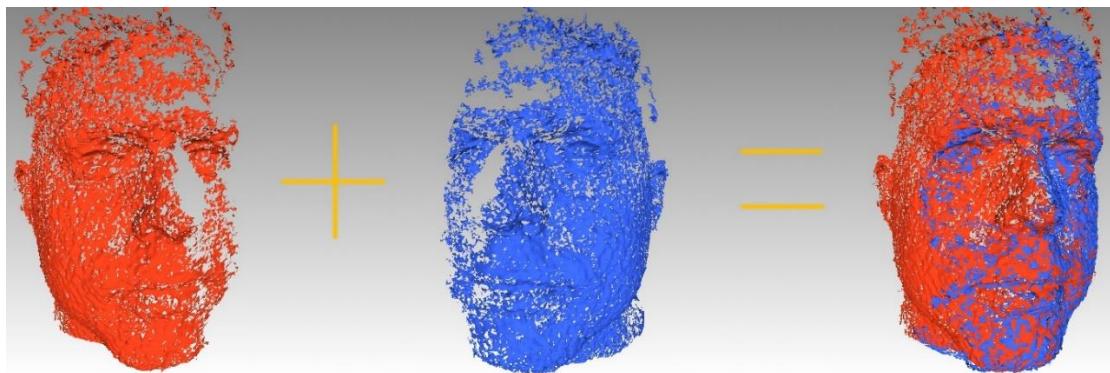
### 2.7. REGISTRO DE NUBES DE PUNTOS 3D

El sensor Microsoft Kinect captura una secuencia de imágenes de profundidad (conjunto de puntos en el espacio tridimensional) del área dentro del marco de la cámara. Al mover Kinect alrededor del objeto estacionario o girar el objeto sobre una plataforma giratoria frente a Kinect, permite capturar toda la

superficie del objeto. Para reconstruir toda la superficie, las vistas individuales deben alinearse. Cada vista del objeto se traslada y rota (transforma) ligeramente en comparación con la anterior. Para alinear dos nubes de puntos, se necesita encontrar esa transformación relativa entre dos vistas. Una vez que se encuentra la transformación, una de las nubes puntos se pueden alinear con respecto a la otra con la finalidad que queden en el mismo marco referencial. El proceso de encontrar una transformación adecuada se llama registro de nube de puntos. Repetir este proceso para todos los puntos de vista permite alinear todas las nubes de puntos y así recrear toda la superficie del objeto [8]. En las figuras 2.13 y 2.14 se puede observar que se ha capturado distintas vistas en nubes de puntos de un objeto y un rostro, luego se han registrado entre sí obteniendo una única nube de puntos con información de todas las nubes individuales asociadas al objeto y al rostro.



**Figura 2.13.** Registro de 3 nubes de puntos de un objeto (tetera) [43].



**Figura 2.14.** Registro de 2 nubes de puntos de un rostro humano [44].

Uno de los algoritmos más utilizados en los antecedentes para el registro de nubes de puntos 3D es el ICP (Iterative Closest Point) [45], que encuentra el registro minimizando la distancia entre las correspondencias de puntos, conocidas como el punto más cercano. Sin embargo, este solo tiene un buen desempeño cuando efectivamente las nubes de puntos de diferentes vistas presentan vecindad o un solapamiento considerable, de lo contrario este algoritmo tiene un desempeño bastante pobre. Por lo que soluciones que operan con flujo de video (~30 FPS) como lo es KinectFusion [1][2][3] o aplicaciones que aseguren esta condición de vecindad eliminan la necesidad de utilizar un registro grueso inicial antes de usar el algoritmo ICP.

Los antecedentes [15] y [5] proponen el uso del algoritmo RANSAC (Random Sample Consensus) para mitigar esta problemática de vecindad entre nubes de puntos en una etapa a la que denominan registro grueso. Con la información RGB y de profundidad, se procede a usar el algoritmo RANSAC para realizar una alineación gruesa. En el procedimiento de alineación gruesa, las nubes de puntos de dos vistas adyacentes se registran aproximadamente en base a las características SIFT en las imágenes RGB, dando como resultado una primera aproximación de transformación entre ambas nubes.

La segunda etapa del registro es el registro fino mediante el algoritmo ICP la cual toma como punto de partida la aproximación de la transformación que arroja la etapa de registro grueso. El algoritmo estándar ICP alinea dos nubes de puntos mediante la asociación iterativa de puntos a través de una búsqueda de vecindad más cercana, y la estimación de los parámetros de transformación utilizando una función de costo cuadrado medio [45] [46].

Se asegurará la vecindad entre nubes de puntos, ya que se tomarán las imágenes al objeto con diferencias angulares controladas bajo una plataforma rotativa, se podría pensar de prescindir de una primera etapa de registro grueso mediante RANSAC y únicamente usar una etapa de registro mediante ICP. La razón de que este trabajo tenga 2 etapas de registro y no solo una en la que se use únicamente ICP, es debido a la problemática de fallo constante presente en los algoritmos de registro que utilizan técnicas de optimización de distancia entre nubes como lo es ICP. En donde la técnica de optimización de dichos algoritmos no siempre alcanza a dar el mejor resultado de registro debido a que queda atrapada un mínimo local en vez de un mínimo global y en muchas ocasiones el resultado es pobre y no es suficiente al tratarse de nubes de objetos pequeños [34].

Este trabajo propone implementar un registro de dos etapas. La primera etapa, corresponde a un registro grueso en donde se implementará el algoritmo SAC-IA (Sample Consensus Initial Alignment) visto en [34] que utiliza histogramas de características de punto rápido (FPFH) para realizar una alineación inicial que asegura un registro en el espacio mínimo global correcto. La siguiente etapa, de registro de fino se utilizará el algoritmo ICP que partirá desde el resultado arrojado por la etapa de registro grueso y que tendrá un punto de parada más estricto que el algoritmo en el registro grueso.

Pero antes de analizar el algoritmo SAC- IA y ICP se introducirá el concepto de transformación 3D para lograr una buena comprensión de los algoritmos.

### 2.7.1. Transformación 3D

Una transformación espacial 3D hace referencia al proceso de trasladar, rotar, cambiar de escala o deformar un conjunto de puntos en un espacio tridimensional. En este trabajo asumiremos que los objetos que se escanean no serán deformados en el proceso, por lo cual se trabajará solo con transformaciones rígidas ya que estas no contemplan la acción de deformación ni cambio de escala.

Lo primero para abordar el tema, es definir un punto en el espacio tridimensional como se puede ver en la ecuación 2.9.

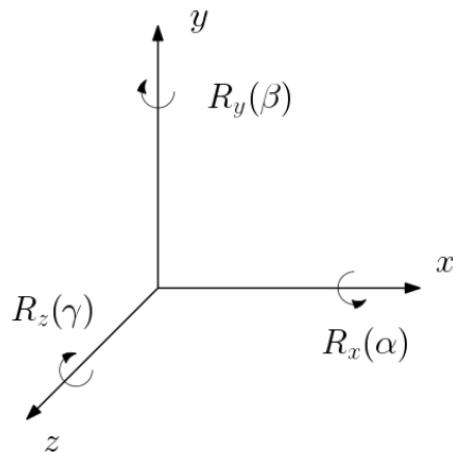
$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \quad (2.9)$$

Un conjunto de puntos relacionados se llama nube de puntos y están descritos por la ecuación 2.10. En donde N es la cantidad de puntos en la nube.

$$P = \{p_i \mid i = 1, \dots, N\} \quad (2.10)$$

#### 2.7.1.1. Rotación

La primera componente que analizaremos de la transformación rígida es la rotación. La rotación de una nube de puntos se puede realizar de manera independiente con respecto a los 3 ejes de un espacio tridimensional como se puede observar en la figura 2.15.



**Figura 2.15.** Rotación alrededor del eje x( $\alpha$ ), eje y( $\beta$ ) y el eje z( $\gamma$ ).

Como se mencionó, la rotación puede expresarse mediante tres rotaciones independientes alrededor del eje fijo utilizando matrices de rotación como se puede ver en la ecuación 2.11, donde  $\alpha, \beta$  y  $\gamma$  son los ángulos de rotación respecto al eje X, Y y Z respectivamente.

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix},$$

**(2.11)**

$$R_y(\beta) = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix},$$

$$R_z(\gamma) = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Para adquirir la matriz de rotación completa, las rotaciones individuales se deben aplicar una a la vez. Dado que los ángulos dependen del orden de aplicación de las rotaciones, el orden debe ser fijo. En este libro utilizaremos la siguiente notación para una matriz de rotación 3D sobre los 3 ejes, descrita en la ecuación 2.12.

$$R = R(\alpha, \beta, \gamma) = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

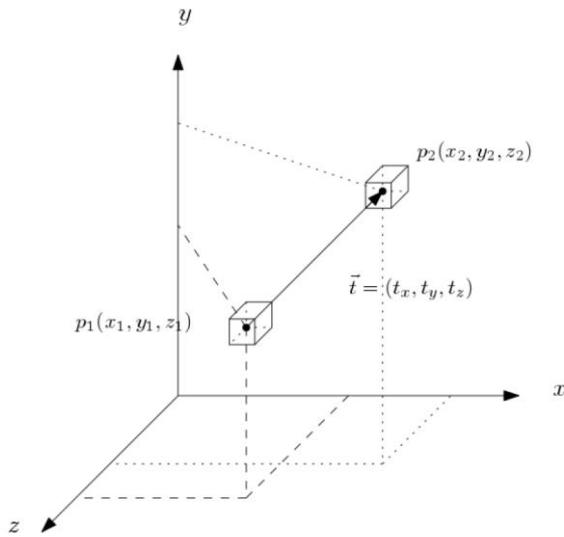
$$R_{(\alpha, \beta, \gamma)} = \begin{pmatrix} \cos \beta \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\ \cos \beta \sin \gamma & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma - \sin \alpha \cos \gamma \\ -\sin \beta & \sin \alpha \cos \beta & \cos \alpha \cos \beta \end{pmatrix} \quad \text{(2.12)}$$

Para ángulos pequeños, se puede hacer una aproximación usando  $\cos \theta \approx 1$ ,  $\sin^2 \theta \approx 0$  y  $\sin \theta \approx \theta$ . Por lo tanto, la matriz de rotación quedaría como lo muestra la ecuación 2.13.

$$R_{(\alpha, \beta, \gamma)} = \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix} \quad \text{(2.13)}$$

### 2.7.1.2. Translación

La segunda y última componente que analizaremos de la transformación rígida es la translación. La translación de un punto o una nube de puntos se puede definir como movimientos directos en el espacio tridimensional sin ningún cambio en la orientación como se puede ver en la figura 2.16.



**Figura 2.16.** Translación de  $p_1$  a  $p_2$ .

La translación en el espacio tridimensional se puede expresar como el vector descrito en la ecuación 2.14.

$$\vec{t} = (t_x, t_y, t_z) \quad \text{(2.14)}$$

Donde  $t_x, t_y, t_z$  muestra el movimiento a través de cada uno de los 3 ejes. Entonces, las coordenadas de un punto después de la translación se escriben según la ecuación 2.15 donde  $p_1$  es el punto a trasladar,  $p_2$  el punto trasladado y  $\vec{t}$  la translación aplicada.

$$p_2 = p_1 + \vec{t} \quad \text{(2.15)}$$

### 2.7.1.3. Transformación rígida

Como se mencionó con anterioridad, si todos los puntos en el objeto tienen la misma distancia entre ellos antes y después de la transformación ósea no hay deformación en la nube de puntos, entonces la transformación se llama rígida. En este libro solo se consideran transformaciones rígidas las consistentes en rotación y la translación como en la ecuación 2.16. Donde,  $R$  es la rotación y  $t^\rightarrow$  la translación aplicada al punto  $p_i$  que da como resultante el punto  $m_i$ .

$$m_i = Rp_i + t^\rightarrow \quad (2.16)$$

### 2.7.1.4. Coordenadas homogéneas y matriz de transformación

Con el objetivo de expresar ambos movimientos de translación y rotación en una misma matriz se proponen las coordenadas homogéneas. Las coordenadas homogéneas pueden expresar las coordenadas de un punto  $p_i$  en un espacio tridimensional como se puede observar en la ecuación 2.17.

$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad (2.17)$$

Como objetivo principal las coordenadas homogéneas permiten expresar una matriz aumentada de transformación que contiene rotaciones y translaciones como lo indica la ecuación 2.18. En donde  $R_{i,j}$  son parámetros correspondientes a cambios de escala y rotación, mientras que los parámetros  $t_x$ ,  $t_y$  y  $t_z$  son parámetros correspondientes a traslación.

$$T = \begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_x \\ R_{2,1} & R_{2,2} & R_{2,3} & t_y \\ R_{3,1} & R_{3,2} & R_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.18)$$

Usando la misma notación, la ecuación 2.16. se puede llegar a la simplificación que se observa en la ecuación 2.19 que hace referencia a la transformación rígida aplicada al punto  $p_i$  mediante la matriz de transformación aumentada.

$$m_i = Tp_i \quad (2.19)$$

Como último concepto para entrar a discutir los algoritmos de registro, introduciremos el concepto de composición de transformadas. Dadas 3 nubes de puntos  $C_1$ ,  $C_2$  y  $C_3$ , la transformación aumentada resultante de registrar  $C_1$  y  $C_2$  llamada  $T_1$ , la transformación aumentada resultante de registrar  $C_2$  y  $C_3$  llamada  $T_2$  se desea encontrar la transformación aumentada que registre las nubes de puntos  $C_1$ ,  $C_3$ . Para solucionar este problema, se deben componer las transformadas  $T_1$  y  $T_2$  como se puede observar en la ecuación 2.20.

$$T_1 = \begin{pmatrix} R_{1,1(1)} & R_{1,2(1)} & R_{1,3(1)} & t_{x1} \\ R_{2,1(1)} & R_{2,2(1)} & R_{2,3(1)} & t_{y1} \\ R_{3,1(1)} & R_{3,2(1)} & R_{3,3(1)} & t_{z1} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} R_{1,1(2)} & R_{1,2(2)} & R_{1,3(2)} & t_{x2} \\ R_{2,1(2)} & R_{2,2(2)} & R_{2,3(2)} & t_{y2} \\ R_{3,1(2)} & R_{3,2(2)} & R_{3,3(2)} & t_{z2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.20)$$

$$T_3 = T_1 \otimes T_2$$

$$T_3 =$$

$$\begin{pmatrix} R_{1,1(1)}R_{1,1(2)} + R_{1,2(1)}R_{2,1(2)} + R_{1,3(1)}R_{3,1(2)} & R_{1,1(1)}R_{1,2(2)} + R_{1,2(1)}R_{2,2(2)} + R_{1,3(1)}R_{3,2(2)} & R_{1,1(1)}R_{1,3(2)} + R_{1,2(1)}R_{2,3(2)} + R_{1,3(1)}R_{3,3(2)} & t_{x1} + t_{x2} \\ R_{2,1(1)}R_{1,1(2)} + R_{2,2(1)}R_{2,1(2)} + R_{2,3(1)}R_{3,1(2)} & R_{2,1(1)}R_{1,2(2)} + R_{2,2(1)}R_{2,2(2)} + R_{2,3(1)}R_{3,2(2)} & R_{1,1(1)}R_{1,3(2)} + R_{1,2(1)}R_{2,3(2)} + R_{1,3(1)}R_{3,3(2)} & t_{y1} + t_{y2} \\ R_{3,1(1)}R_{1,1(2)} + R_{3,2(1)}R_{2,1(2)} + R_{3,3(1)}R_{3,1(2)} & R_{3,1(1)}R_{1,2(2)} + R_{3,2(1)}R_{2,2(2)} + R_{3,3(1)}R_{3,2(2)} & R_{1,1(1)}R_{1,3(2)} + R_{1,2(1)}R_{2,3(2)} + R_{1,3(1)}R_{3,3(2)} & t_{z1} + t_{z2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ahora que conocemos el concepto básico de transformación rígida, explicaremos cómo funciona el algoritmo ICP que se usará en ambas etapas de registro.

### 2.7.2. Algoritmo ICP

En un algoritmo de registro por lo general, los puntos a los que se aplica la transformación se llaman puntos de datos y la otra nube de puntos se llama puntos de modelo. A continuación, los puntos de datos se

denotan por  $P = \{pi \mid i = 1, \dots, N\}$  y los puntos modelo por  $M = \{mi \mid i = 1, \dots, N\}$ . El objetivo del algoritmo puede describirse matemáticamente como la minimización de una métrica  $e$  descrita en la ecuación 2.21, en donde  $M$  y  $P$  son las nubes a registrar y  $T$  la transformación que alinea ambas nubes.

$$e = f(M, TP) \quad (2.21)$$

La métrica que utiliza el algoritmo ICP que se utilizará en este trabajo es la suma de cuadrados. Dada la correspondencia de puntos entre  $pi$  y  $mi$ , y se define en la ecuación 2.22.

$$e = \frac{1}{N} \sum_{i=1}^N \|m_i - Tp_i\|^2 \quad (2.22)$$

En dónde  $mi$  hace referencia a un punto modelo,  $pi$  un punto de datos,  $N$  la cantidad de puntos si se trata de nube de puntos y  $T$  la transformación aplicada.

Debido a que, en la configuración actual, la forma del objeto no cambia, se considera únicamente la rotación y la translación entre puntos en el algoritmo de registro. Esto permite expresar la métrica de minimización por rotación y translación como lo muestra la ecuación 2.24.

$$e = \frac{1}{N} \sum_{i=1}^N \|m_i - Rp_i - t^\rightarrow\|^2 \quad (2.23)$$

En donde  $R$  y  $t^\rightarrow$  hacen referencia a la rotación y translación aplicada al punto de datos  $pi$  respectivamente.

Si trabajamos la rotación mediante matrices nos resultan tres parámetros a modificar y si adicionamos la translación tiene nos resultan otros tres, en total hay seis parámetros para estimar. Dado al menos tres puntos correspondientes, los parámetros pueden ser estimados. Los pasos que sigue el algoritmo ICP para resolver el problema de minimización propuesto en la ecuación 2.18 son [45]:

1. Los puntos de datos  $P$  y los puntos de modelo  $M$  son dados.
2. Inicialice la iteración configurando  $P_0 = P$ ,  $R = I$ ,  $t^\rightarrow = (0,0,0)$  y  $k = 0$ . Si hay alguna estimación inicial para la rotación y de la translación de alguna otra fuente, utilícelos para la inicialización. Los pasos 3-7 se repiten hasta la convergencia dentro de cierta tolerancia  $\tau$ .
3. Calcule los puntos más cercanos  $Y_k$  para  $P_k$  en  $M$ .
4. Encuentre la transformación óptima  $T_k$  entre  $P_0$  y  $Y_k$ .
5. Aplique la transformación  $P_{k+1} = T_k P_0$ .
6. Encuentre el error  $e_k$  entre  $Y_k$  y  $M$ . Termine el proceso si el error cae por debajo del umbral,  $e_k - e_{k+1} < \tau$ .

Ahora que conocemos el funcionamiento del algoritmo ICP explicaremos como trabaja el algoritmo que se usará en la etapa de registro grueso denominado SAC-IA (Sample Consensus Initial Alignment).

### 2.7.3. Algoritmo SAC-IA

La tarea de un algoritmo de registro es encontrar la transformación aumentada  $T$  que permita alinear dos nubes de puntos 3D con el fin de llevarlas a un mismo marco referencial con el mínimo error posible. En este caso, el algoritmo SAC-IA encuentra una transformación inicial entre dos nubes de puntos a partir de sus características geométricas extraídas con anterioridad, mediante el algoritmo FPFH. La metodología que sigue el algoritmo para resolver el problema de la estimación inicial de registro es la siguiente [34]:

1. Seleccione los puntos de muestra  $S$  de una nube de puntos 3D  $P$  mientras se asegura que la distancia entre ellos sea al menos mayor que una distancia mínima definida por el usuario  $dmin$ .
2. Para cada uno de los puntos de muestra  $S$ , encuentre una lista de puntos en  $Q$  cuyos histogramas sean similares al histograma de los puntos de muestra  $S$ . De estos, seleccione uno al azar que se considerará la correspondencia a los puntos de muestra  $S$ .
3. Calcule la transformación rígida definida por los puntos de muestra y sus correspondencias mediante el algoritmo ICP, al cual se le ha definido una nueva métrica definida por la ecuación 2.24 en lugar de la métrica convencional descrita en la ecuación 2.23. Luego calcule una métrica de error para evaluar la transformación encontrada resultante del registro del par de nubes de puntos 3D.

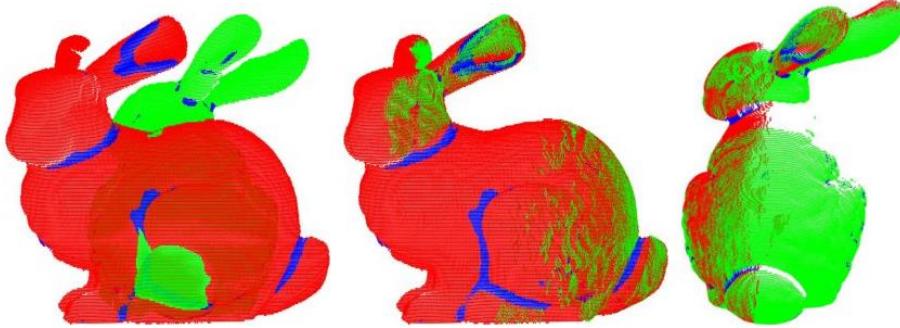
$$e = \frac{1}{N} \sum_{i=1}^N \| \langle (pi - mi), n_{mi} \rangle \|^2 \quad (2.24)$$

En donde  $mi$  hace referencia a un punto modelo,  $pi$  un punto de datos,  $N$  la cantidad de puntos si se trata de nube de puntos y  $n_{mi}$  la normal de superficie del punto modelo  $mi$ .

La métrica de error usada en el tercer paso se determina usando una medida de penalización de Huber  $L_h$  descrita en la ecuación 2.25.

$$L_h(e_i) = \begin{cases} \frac{1}{2} e_i^2 & \|e_i\| \leq t_e \\ \frac{1}{2} t_e(2\|e_i\| - t_e) & \|e_i\| > t_e \end{cases} \quad (2.25)$$

Siendo  $t_e$  el límite lineal/cuadrático y  $e_i$  la diferencia entre ambas nubes ya transformadas. Estos tres pasos se repiten y la transformación que presenta el menor error  $L_h$  es guardada y usada como la transformación inicial resultante entre ambas nubes. La figura 2.17 presenta los resultados obtenidos después del registro con SAC-IA en dos vistas parciales del modelo de conejito de Stanford.

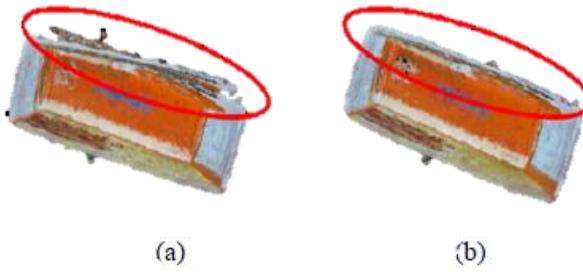


**Figura 2.17.** Resultados obtenidos después del registro con SAC-IA en dos vistas parciales del modelo de conejito de Stanford [47].

## 2.8. ALINEACIÓN GLOBAL

En el proceso de registro, los errores de registro podrían acumularse a medida que se agreguen más nubes de puntos al modelo. La figura 2.18 (a) extraída de [5] ilustra los efectos de esta acumulación de errores. Una ligera inexactitud en cada paso de registro conduce a una desalineación significativa entre el último y el primer cuadro.

La metodología que se empleará para mitigar esta problemática es la siguiente: se implementará una alineación global que consiste en detectar la última trama de un cierre de bucle, registrar este marco con su trama anterior y la primera trama, y distribuir el error uniformemente entre todos marcos como se detalla en [48]. A continuación, se ampliarán los detalles correspondientes a la metodología de la alineación global extraída de [5].



**Figura 2.18.** En (a), (b) se aprecia el resultado del registro sin y con alineación global.

El algoritmo primeramente debe detectar cuando ocurre el cerrado de bucle. Técnicas anteriores han utilizado la distancia euclídea entre cada marco y todos los marcos anteriores, junto con un umbral de número mínimo de escaneos intermedios [48] [49] para detectar cierres de bucles. Este trabajo empleará la de detección de cierre de bucle basada en el supuesto de que la cámara captura las vistas del objeto en sentido horario o anti horario. Cuando se capturen los datos, se asegura que entre nubes de puntos existe una diferencia angular conocida debido a que se controla su rotación mediante una plataforma lo que hace que mecánicamente se asegure que se cierran los bucles y se pueda detectar el cierre del bucle con relativa facilidad.

### 2.8.1. Algoritmo utilizado

Cuando se detecta un cierre de bucle, se formula un bucle gráfico con cada vértice que representa una vista de la nube de puntos 3D. El borde entre los marcos sucesivos es la transformación de postura entre tramas sucesivas. Primero se usa ICP para alinear el primer y último marco con cierre de bucle para obtener la transformación  $\Delta Y$  que debería ser la composición de todas las transformadas entre marcos anteriores. Donde resulta la ecuación 2.26.

$$T_1 \otimes T_2 \otimes T_3 \otimes \dots T_n = \Delta Y \quad (2.26)$$

Donde  $T_i$  es la transformación de postura que se encuentra en el proceso de registro. Si el registro no tiene errores,  $\Delta Y$  debe ser una matriz de identidad. La no identidad de  $\Delta Y$  se debe a los errores de registro.

Se denotan la inversa de  $\Delta Y$  como  $\Delta X$ , entonces  $\Delta X \otimes T_1 \otimes T_2 \dots T_n = I$ , donde  $I$  es la matriz de identidad.  $\Delta X$  se puede distribuir entre todas las posturas en el bucle para minimizar la falta de coincidencia del cierre del bucle. Para lograr un mapa coherente, se necesita calcular los pesos para los vértices que especifican la fracción de  $\Delta X$  por la cual la transformación necesita ser cambiada. Los pesos se calculan como se observa en la ecuación 2.27.

$$W_i = \frac{d(v_i, v_{i-1})}{d(v_s, v_e)} \quad (2.27)$$

En donde  $v_s$  es el primer vértice en el bucle y  $v_e$  el ultimo. Además,  $d(v_l, v_k)$  es la suma de los pesos de los bordes en la trayectoria  $v_l$  a  $v_k$ . La ponderación resultante  $W_i \in [0,1]$  se expresa en la ecuación 2.28.

$$\Delta X = \sum_i (W_i * \Delta X) \quad (2.28)$$

La transformación final modificada para cada marco  $i$  después de la optimización se observa en la ecuación 2.29.

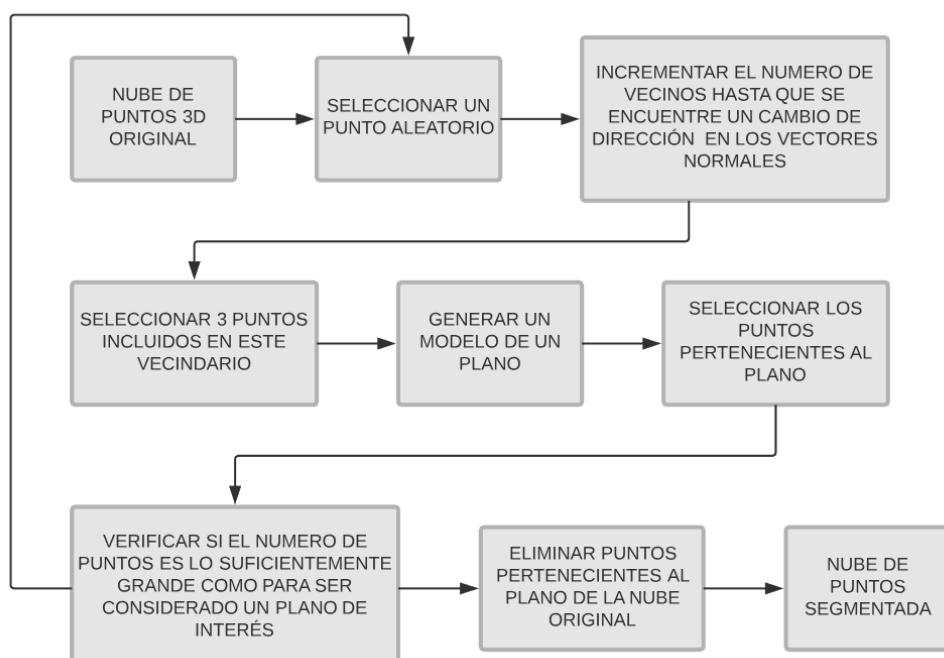
$$T'_i = W_i * \Delta X * T_i \quad (2.29)$$

En donde  $T'_i$  es la transformación modificada [49]. El proceso conjunto se repite varias veces hasta que la convergencia se haya logrado ya sea por un error mínimo especificado  $\Delta Y$  o cuando se alcance la igualdad  $\Delta Y = I$ .

## 2.9. SEGMENTACIÓN DE PLANO TIERRA

Con la finalidad de obtener un modelo 3D único que contenga información del objeto escaneado se debe segmentar los puntos que corresponde al objeto de interés de los demás puntos que hagan al fondo de la escena. Para la identificación y posterior segmentación de los diferentes planos que componen a la escena escaneada se utilizará la técnica denominada: Segmentación de información de rango usando normales de superficie (SIRUNS) [50]. La SIRUNS permite descomponer una nube en clústeres más pequeños e incluso detectar modelos geométricos primarios. A nosotros solo nos importa descomponer la nube 3D en dos grandes clústeres: el objeto de interés y el plano donde reposa.

En la figura 2.19 se puede apreciar un diagrama de flujo sobre la metodología que rige al algoritmo SIRUNS para la eliminación del plano tierra de una nube de puntos 3D.



**Figura 2.19.** Vectores normales a una superficie 3D.

Como primer instancia se selecciona un punto aleatorio  $p$  de toda la nube 3D y se agrupa en una vecindad de puntos de dimensión  $k_i$  definida por el usuario. Luego se analiza todas las normales asociadas a los

puntos de la vecindad buscando cambios significativos de dirección. Si en la vecindad no presentan cambios en la dirección las normales, se hace mayor la dimensión de la vecindad y se vuelven a analizar haciendo repetitivo este proceso si es necesario. Luego de encontrar cambios significativos en las normales en la vecindad se seleccionan 3 puntos incluidos en el vecindario para generar un modelo de un plano como se puede observar en la ecuación 2.30.

$$Ax + By + Cz + D = 0 \quad (2.30)$$

Ahora se analizan todos los puntos de la nube para reportar cuantos son los que pertenecen al plano y decidir si son suficientes como para ser considerado el plano tierra de la nube. Si no son suficientes se repite todo el proceso hasta la selección de un punto aleatorio  $P$ , de lo contrario se remueven los puntos pertenecientes al plano de la nube original dando como resultado la nube segmentada.

## 2.10. REDUCCIÓN DE RUIDO

Después del registro y la alineación global, el modelo 3D se considera bien registrado. Sin embargo, sigue siendo muy ruidoso debido a la adquisición de profundidad imprecisa del dispositivo Microsoft Kinect. Por lo tanto, la eliminación de ruido en la nube de puntos 3D es necesaria para refinar el modelo de objetos 3D.

La técnica utilizada para esta labor es Moving Least Square (MLS) [51]. MLS utiliza la suposición de que en una superficie 3D, el conjunto de puntos 3D define un colector. El propósito de MLS es encontrar el colector que define el conjunto de puntos de entrada 3D. Basado en la anterior suposición anteriormente mencionada, se puede aproximar la superficie MLS por una función. En la siguiente sección se explicará todo el argumento matemático que soporta al algoritmo MLS

### 2.10.1. Reducción de ruido mediante el algoritmo MLS

Sea  $S_i \in R^3, i \in \{1, \dots, M\}$  puntos en la ruidosa nube de puntos 3D. El objetivo del algoritmo es proyectar los puntos en una superficie bidimensional que se aproxime a  $S_i$ . Para cada pequeño vecindario local en la nube de puntos, se ajustan los puntos por un plano tangente local  $H$ . Se denota la altura de  $S_i$  sobre  $H$  como  $r_i$ , y  $h$  el origen del dominio de referencia definido por  $H$ . De tal manera se puede estimar una aproximación polinómica  $g$  tal que el error de cuadrados ponderados queda como lo muestra la ecuación 2.31.

$$\sum_{i=1}^M (g(x_i, y_i) - r_i)^2 \theta(\|s_i - h\|) \quad (2.31)$$

Donde  $\theta$  es una función monótona decreciente lisa, que es positiva en todo el espacio (por ejemplo una aproximación gaussiana que se realizará en esta implementación).  $(x_i, y_i)$  es la coordenada de la proyección de  $S_i$ 's en  $H$ . La función de error anterior se minimiza para luego calcular coeficientes de la función polinómica  $g$ . El proceso de aproximación de la proyección se ejecuta para cada punto del conjunto de puntos 3D y se ha demostrado que puede preservar muy bien la propiedad del colector [51]. En la figura 2.20 se puede apreciar el resultado de esta técnica extraída del antecedente [5].



**Figura 2.20.** En (a) y (b), se aprecia el modelo antes y después de realizar la reducción de ruido mediante la técnica MLS respectivamente.

## 2.11. GENERACIÓN DE SUPERFICIE

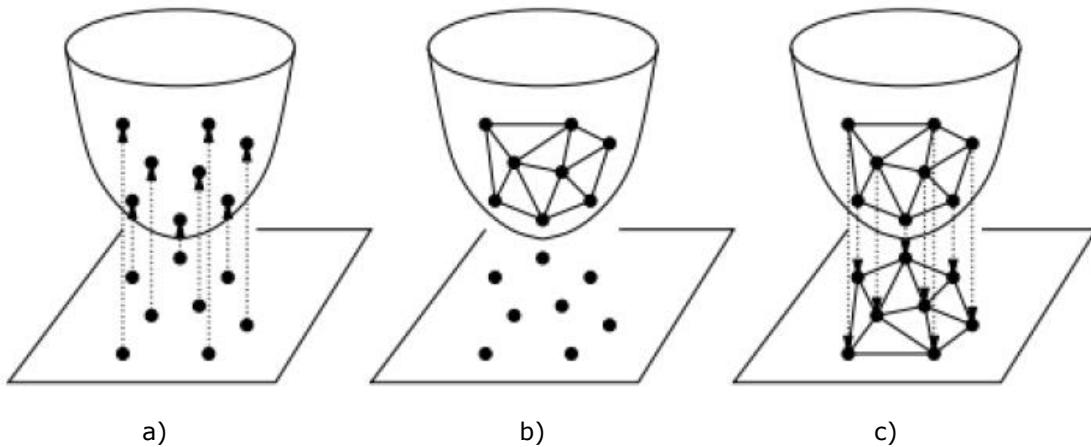
El modelo 3D inicial está representado en nubes de puntos. La obtención de una superficie sobre esta nube de puntos hace que el modelo se asemeje al objeto de la vida real. Para este propósito se recurrirá a la técnica de triangulación de nubes de puntos 3D mediante el método Delaunay [52], la cual es la más utilizada y documentada entre los antecedentes analizados. Una triangulación de Delaunay para un conjunto  $P$  de puntos en un plano se define como una triangulación  $DT(P)$  tal que ningún punto en  $P$  está dentro de la circunferencia de cualquier triángulo en  $DT(P)$ . Podemos interpolar el espacio dentro de los triángulos para obtener una superficie localmente suavizada. Primeramente, se discutirá sobre la

metodología que plantea la triangulación de Delaunay en dos dimensiones que es la base que se utiliza para realizar esta misma técnica en dimensiones más elevadas.

### 2.11.1. Triangulación de Delaunay 2D

Basado en el trabajo de Delaunay (1934), y comúnmente definido en 2D, el principio de Triangulación de Delaunay establece que para un conjunto  $P$  de puntos 2D, existe una triangulación  $DT(P)$  tal que ningún punto en  $P$  está dentro de una circunferencia circunscrita por ningún triángulo en  $DT(P)$ . Las principales aplicaciones de la Triangulación de Delaunay están relacionadas con las búsquedas de vecinos más cercanos, la generación de superficies sobre nubes de puntos y la planificación de rutas [53].

Gallier [54] describió una relación clara y directa entre las triangulaciones de Delaunay y los cascos convexos, encontrados por primera vez por Edelsbrunner y Seidel [55]. Esta relación establece que, dados un conjunto de puntos  $P$  en el espacio Euclídeo  $R^n$ , dichos puntos pueden elevarse a un paraboloide definido en  $R^{n+1}$ . Entonces, la Triangulación de Delaunay de  $P$  es la retroproyección a  $R^n$  de las caras más bajas del casco convexo del conjunto de puntos elevados, como se muestra en la Figura 2.21.



**Figura 2.21.** a) Relación entre la triangulación de Delaunay y un casco convexo 3D. Puntos proyectados a un paraboloide. b) Cálculo del casco convexo en el paraboloide. c) El casco convexo proyectado en el plano [54].

Considere un conjunto de puntos  $P = p, q, r$  en el plano  $R^2$ , siendo  $P' = p', q', r'$  sus respectivas proyecciones sobre el paraboloide  $z = x^2 + y^2$ . Como mencionamos antes, el triángulo definido por los puntos que pertenecen a  $R$  está contenido en la Triangulación de Delaunay si y solo ninguno punto  $s$  no está en su interior.

Extrapolando a  $R^3$ , su proyección,  $s'$ , no debe estar en el lado inferior del plano que pasa a través de los puntos en  $P'$ . Esto se llama Delaunay Lemma.

Para probar el lemma, se toma un plano arbitrario no vertical, que es tangente al paraboloide sobre algún punto  $(a, b)$  en el plano. La ecuación de dicho plano se puede tomar de las derivadas parciales del paraboloide como lo muestra la ecuación 2.32.

$$\frac{\partial z}{\partial x} = 2x, \quad \frac{\partial z}{\partial y} = 2y \quad (2.32)$$

En el punto considerado  $(a, b)$ , las derivadas parciales se evalúan a  $2a$  y  $2b$ , lo que significa que el plano tiene como ecuación la dentada en la ecuación 2.33.

$$z = 2ax + 2by + \gamma \quad (2.33)$$

Para resolver  $\gamma$  hay que considerar que el plano pasa por  $(a, b, a^2 + b^2)$ , lo que implica lo propuesto en la ecuación 2.34.

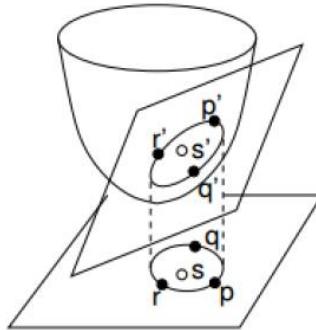
$$a^2 + b^2 = 2a \cdot a + 2b \cdot b + \gamma \Leftrightarrow \gamma = -(a^2 + b^2) \Rightarrow z = 2ax + 2by - (a^2 + b^2) \quad (2.34)$$

La intersección del plano con el paraboloide se puede obtener desplazando el plano una cantidad positiva  $r^2$  y tomando la ecuación paraboloide para reemplazar  $z$  como lo muestra la ecuación 2.35.

$$x^2 + y^2 = 2ax + 2by - (a^2 + b^2) \Rightarrow (x - a)^2 + (y - b)^2 = r^2 \quad (2.35)$$

La ecuación 2.28 es claramente el modelo matemático de un círculo. Entonces, se demuestra que la intersección de un plano arbitrario, que pasa por los puntos elevados, con el paraboloide es una elipse, cuya proyección en  $R^2$  es un circunferencia circunscrita en  $p, q$  y  $r$ , centrada en  $(a, b)$ , como se muestra en

la Figura 2.22. Además, el punto  $s$  se encuentra dentro de este círculo, si y solo si su proyección  $s'$  se encuentra sobre el paraboloide dentro de la mitad inferior del plano que pasa por  $p', q'$  y  $r'$  [56].



**Figura 2.22.** Un punto específico se encuentra dentro de la circunferencia circunscrita por  $p, q$  y  $r$ , si solo si su proyección del espacio 3D está por debajo del plano que contiene las proyecciones  $p', q'$  y  $r'$  [56].

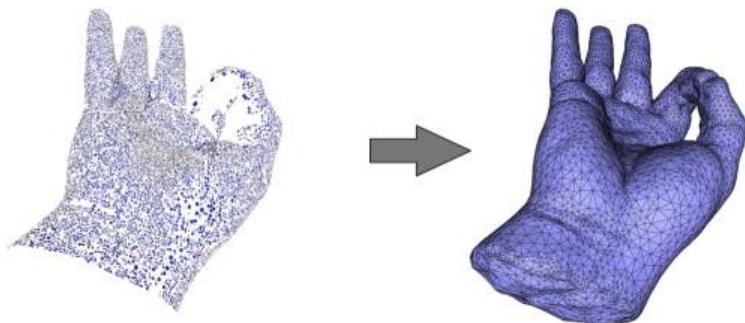
Para concluir acerca de la posición relativa de  $s$  y la circunferencia , se deben calcular dos determinantes,  $\Delta$  y  $\Gamma$ . Si tienen signos opuestos,  $s$  está dentro del círculo. Los determinantes se calculan como se muestra en la ecuación 2.36.

$$\Delta = \begin{bmatrix} 1 & p & p' & p^2 + p'^2 \\ 1 & q & q' & q^2 + q'^2 \\ 1 & r & r' & r^2 + r'^2 \\ 1 & s & s' & s^2 + s'^2 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} 1 & p & p' \\ 1 & q & q' \\ 1 & r & r' \end{bmatrix} \quad (2.36)$$

Los algoritmos de Triangulación de Delaunay intentan definir los triángulos entre puntos adyacentes que no tienen ningún otro punto en su interior, es decir, que garantizan que los determinantes antes mencionados tienen el mismo signo.

### 2.11.2. Triangulación de Delaunay 3D

La definición presentada de Triangulación de Delaunay se puede extrapolar para dimensiones más altas. Especialmente para conjuntos de puntos en  $R^3$ , se deben elevar hacia una representación similar a un paraboloide en  $R^4$ , cuya re-proyección en la dimensión original crea un conjunto de tetraedros adyacentes, cuyas esferas circunscritas tienen interiores vacíos. La intersección de cada tetraedro es un conjunto vacío o una cara o borde común [57]. Al aplicar la Triangulación de Delaunay, se obtiene un conjunto de tetraedros casi regulares. Las caras triangulares obtenidas son casi equiláteras, ya que desde el ángulo más pequeño se maximizan dichos triángulos. En la figura 2.23 se puede apreciar el resultado obtenido al aplicar la triangulación 3D de Delaunay a una nube de puntos 3D que corresponde a una mano humana.



**Figura 2.23.** Resultado obtenido al realizar el algoritmo de triangulación 3D de Delaunay a una nube de puntos 3D correspondientes a una mano humana

## 2.12. PROCESO UNIFICADO RACIONAL (RUP)

El proceso racional unificado o RUP es una metodología de desarrollo de software creada por la empresa Rational Software, actualmente propiedad de IBM [58]. Conjunto a lenguaje unificado de modelado (UML) forman la metodología más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos [58]. Esta metodología es implementada en el desarrollo de este trabajo. En la siguiente sección se amplía los principios de desarrollo que rigen esta metodología.

### 2.12.1. Principios de desarrollo

Los principios de desarrollo que plantea la metodología RUP son [58]:

*Adaptar el proceso:* El proceso debe adaptarse a las necesidades impuestas del problema.

*Equilibrar prioridades:* Debe poder encontrarse un equilibrio que satisfaga los requisitos de todos los participantes del proyecto, con el objetivo de minimizar los desacuerdos que puedan surgir entre estos.

*Demostrar valor iterativamente:* Deben realizarse aportes internos sobre el avance del proyecto con la finalidad de refinar la dirección de este.

*Colaboración entre equipos:* Debe existir un canal de comunicación entre los equipos implicados en el proyecto para poder coordinar requisitos, desarrollo, evaluaciones, planes, resultados, entre otros aspectos.

*Enfocarse en la calidad:* El control de calidad debe realizarse en todos los aspectos de la producción del proyecto.

*Elevar el nivel de abstracción:* Se debe motivar el uso de conceptos reutilizables tales como patrones de diseño de software o esquemas.

### 2.12.2. Fases del RUP

La metodología RUP divide el proceso de desarrollo en 4 fases, dentro de las cuales se realizan pocas pero grandes iteraciones dependiendo de la complejidad del proyecto [58]. En la figura 2.24 se muestra el esfuerzo asociado a las actividades según la fase de la metodología RUP que se encuentre el proyecto.



**Figura 2.24.** Esfuerzo en actividades según la fase del proyecto siguiendo la metodología RUP [58].

En la fase de iniciación se encamina hacia el planteamiento del problema, la delimitación del alcance de proyecto y al establecimiento del punto de partida. El propósito de la fase de elaboración es seleccionar los casos de usos que definen la arquitectura base del sistema y dan una solución preliminar. En fase de construcción se completa la funcionalidad del sistema para que en la fase de transición se asegure que el proyecto está libre de errores y listo para usuario final.

## 2.13. OBSERVACIONES FINALES

Este capítulo se ha dedicado a enmarcar el sistema de modelado 3D automático de objetos DoEverything3D respecto a diferentes plataformas diseñadas con el mismo fin, analizando las características de estos sistemas ya implementados tanto en el área académica como en la comercial. Este análisis permitió definir una metodología clara para la implementación del sistema de modelado 3D automático de objetos, ya que se pudo desglosar el objetivo principal del sistema en diversos algoritmos y módulos que trabajan de manera secuencial para llevar a cabo esta tarea.

Además, se hace un repaso y análisis sobre diferentes conceptos y técnicas fundamentales de visión artificial que permiten al lector comprender el funcionamiento de los algoritmos que conforman el sistema de modelado 3D automático de objetos DoEverything3D.

Resumiendo, del estudio de los antecedentes que también manejan la temática de generación de modelos 3D se definió la metodología secuencial de algoritmos para la implementación en el sistema DoEverything3D, dicha metodología hace uso de los siguientes algoritmos:

1. *Conversión de imágenes de profundidad en nubes de puntos 3D:* convierte las imágenes de profundidad adquiridas por el sensor Microsoft Kinect a nubes de puntos 3D con métricas reales (Ver sección 2.4)

2. *Segmentación del plano tierra en nubes de puntos 3D*: elimina información que no pertenezca al objeto de interés de una nube de puntos (Ver sección 2.9).
3. *Registro fino de nubes puntos 3D mediante algoritmo SAC-IA y descriptores FPFH*: encuentra una transformación que ubica un par de nubes de puntos 3D en un mismo espacio global (Ver sección 2.7.2).
4. *Registro grueso de nubes puntos 3D mediante algoritmo ICP*: parte de la transformación resultante del algoritmo de registro fino y encuentra una transformación más óptima una que ubica de mejor manera el par de nubes de puntos 3D en un mismo espacio global (Ver sección 2.7.3).
5. *Alineación global de nubes de puntos 3D*: si al realizar la etapa de registro a múltiples nubes la primera nube no logra coincidir con la última debido a los errores acumulados en la etapa de registro, el algoritmo reparte dichos errores de transformación en todas las nubes de manera ponderada minimizando la problemática (Ver sección 2.8).
6. *Reducción de ruido en nubes de puntos 3D mediante el algoritmo MLS*: refina la nube de puntos del modelo 3D (Ver sección 2.10).
7. *Generación de superficie en nubes de puntos 3D mediante la triangulación de Delaunay*: genera una superficie sobre la nube de puntos del modelo 3D haciendo que el modelo se asemeje de mayor manera al objeto en la vida real (Ver sección 2.11).

Dichos algoritmos se implementarán en el sistema DoEverything3D para que se ejecuten de manera secuencial y conjuntamente con la finalidad de generar un modelo 3D de un objeto de interés.

# CAPÍTULO 3

## 3. DESARROLLO DE 360-PLATFORM-ASSISTANT (360PA)

- 
- 3.1 Introducción
  - 3.2. Requerimientos de diseño
    - 3.2.1. Requerimientos funcionales y no funcionales
    - 3.2.2. Diagrama conceptual
  - 3.3. Componentes de la plataforma móvil
    - 3.3.1. Modulo PC
    - 3.3.2. Modulo microcontrolador
    - 3.3.3. Modulo plataforma
  - 3.4. Conclusiones
- 

### 3.1. INTRODUCCIÓN



Este capítulo tiene como finalidad exponer el diseño y desarrollo del dispositivo 360-PLATFORM-ASSISTANT (360PA) o módulo de rotación de DoEverything3D. Se discutirá el diseño y la implementación tanto de sus componentes hardware como de su software. Cabe aclarar que el desarrollo software del sistema se llevó a cabo mediante la metodología RUP, metodología que presenta las etapas que tuvo el desarrollo de 360PA tomando como fase de inicio los requerimientos funcionales y no funcionales, casos de uso real y diagrama conceptual.

Posteriormente se describe detalladamente el proceso de elaboración y desarrollo de cada uno de los módulos del sistema rigiéndose por el cumplimiento de los requerimientos funcionales y tomando en consideración los requerimientos no funcionales.

### 3.2. REQUERIMIENTOS DE DISEÑO

El desarrollo de 360PA se realizó a través del proceso RUP para desarrollo software y cumpliendo con su estructura, donde se presentan los requerimientos funcionales y no funcionales y diagrama conceptual.

En la carpeta Anexo\_DocumentaciónSoftwareRUP360PA se encontrará la documentación restante: casos de uso real y diagramas de secuencia de la aplicación.

#### 3.2.1. Requerimientos funcionales y no funcionales

Los requerimientos funcionales necesarios para el desarrollo del software de 360PA se muestran en la tabla 3.1. La tabla de requerimientos funcionales presenta tres columnas: Ref.# que enumera cada uno de los requerimientos funcionales, Funciones que describe de forma concisa el requerimiento funcional brevemente y categoría, la cual puede ser O (opcional) o E (esencial).

**Tabla 3.1:** Requerimientos funcionales de diseño para 360PA.

Ref. #	Funciones	Categoría
1.0	Permitir establecer una comunicación con el aplicativo DoEverything3D mediante USB.	E
2.0	Permitir recibir órdenes de rotación desde el aplicativo DoEverything3D mediante el puerto USB.	E
3.0	Permitir realizar rotaciones controladas de un objeto sobre sus 360 grados y visualizar el estado de la magnitud angular de la rotación.	E

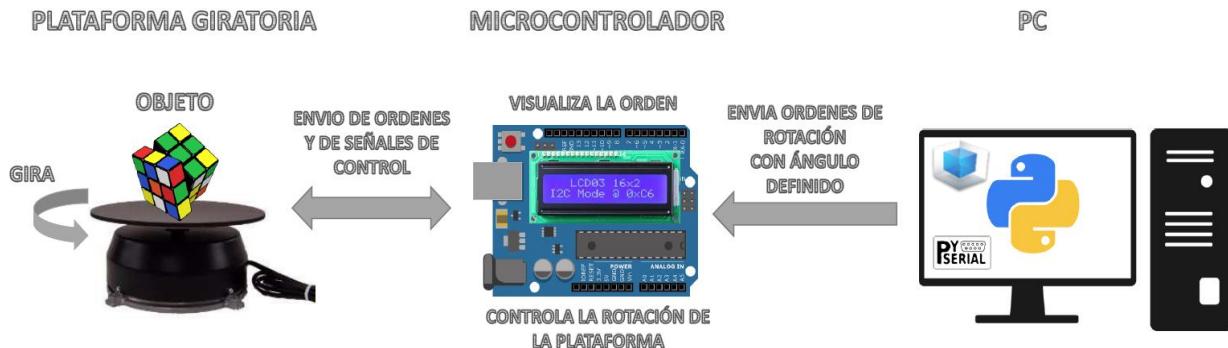
Por otro lado, los requerimientos no funcionales presentan las consideraciones necesarias para el desarrollo del software de 360PA pero que no intervienen en el funcionamiento del mismo, estos requerimientos se muestran en la tabla 3.2. La tabla de requerimientos no funcionales presenta tres columnas: Ref.# que enumera cada uno de los requerimientos no funcionales, Descripción que describe el requerimiento no funcional brevemente y categoría, la cual puede ser O (opcional) o E (esencial).

**Tabla 3.2:** Requerimientos no funcionales de diseño para 360PA.

Ref. #	Descripción	Categoría
1.0	Se requiere un puerto USB 2.0 disponible.	E
2.0	Se requiere tener instalado Python 3.x.	E
3.0	Se requiere tener instalada la API pySerial para Python 3.	E
4.0	Se requiere una alimentación externa de 5V a 1 A.	E
5.0	Se requiere un equipo que cuente con al menos un procesador single Core a 1 GHz y 1 Gb de memoria RAM.	E

### 3.2.2. Diagrama conceptual

El sistema 360-PLATFORM-ASSISTANT (360PA) rige su funcionamiento según lo representado en el diagrama conceptual de la figura 3.1.



**Figura 3.1.** Diagrama conceptual del sistema 360-PLATFORM-ASSISTANT (360PA).

La figura 3.1. es un acercamiento conceptual al sistema 360PA en donde se puede observar el papel que ejerce cada módulo y la asociación existente entre ellos. A continuación, se explica brevemente el sistema desarrollado:

Se implementó un sistema el cual permite realizar rotaciones controladas de un objeto sobre sus 360 grados y a su vez permite visualizar la rotación angular con la que se está girando el objeto. 360PA consta principalmente de 3 grandes módulos:

- ✓ **Módulo PC:** El modulo PC compuesto por un ordenador el cual lleva instalado la API pySerial para Python 3 y el aplicativo DoEverything3D. Este módulo tiene como papel principal identificar el puerto USB en donde esté conectado 360PA y enviar órdenes de rotación hacia el modulo microcontrolador, especificando la cantidad de posiciones a girar siendo cada una de estas posiciones, un equivalente a 18°.
- ✓ **Módulo microcontrolador:** El modulo microcontrolador lo conforma una placa de desarrollo Arduino UNO bajo el microcontrolador ATMEL 328p y un módulo de visualización LCD. Este módulo tiene como objetivo principal recibir la orden del módulo PC y hacer que se ejecute, enviando órdenes y controlando el estado del módulo Plataforma. Además, el modulo permite visualizar la rotación angular con la que se está ejecutando la orden de rotación recibida del módulo PC.
- ✓ **Modulo Plataforma:** El modulo plataforma es conformado por una estructura giratoria motorizada y con rotación monitorizada mediante un encoder. El modulo recibe órdenes de movimiento desde el modulo microcontrolador y envía su señal de control del encoder al módulo microcontrolador.

### 3.3. MODULOS DE 360-PLATFORM-ASSISTANT (360PA)

En esta sección se amplían las características y se explica el funcionamiento de cada uno de los módulos que conforman al sistema 360-PLATFORM-ASSISTANT.

#### 3.3.1. Modulo PC

El modulo PC está compuesto por un ordenador que contiene la API pySerial para Python 3 y el aplicativo DoEverything3D. La API pySerial cumple la función de realizar la comunicación vía puerto USB entre el modulo PC y microcontrolador de manera sencilla, sin necesidad de conocer el funcionamiento del protocolo USB. Sin embargo, para realizar dicha comunicación es necesario conocer el nombre del puerto USB en donde se encuentra conectado el modulo microcontrolador. Por lo tanto, antes de establecer la comunicación, el usuario del aplicativo DoEverything3D selecciona el nombre del puerto de una lista generada por la API pySerial mediante el comando `serial.tools.list_ports.comports()` incluido en la misma API. Luego, se establece la comunicación con dicho puerto seleccionado (`COMx`) mediante el comando `serial.Serial('COMx', baudRate)` siendo el *baudRate* la cantidad de símbolos por segundo (normalmente se usa 9600 con placas de desarrollo Arduino).

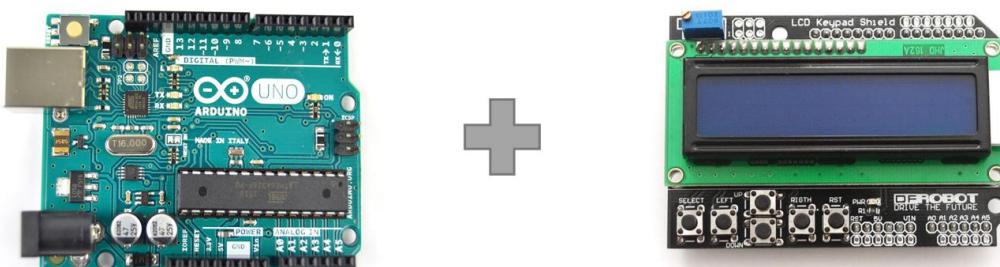
Después de establecer la comunicación entre el modulo PC y microcontrolador, el modulo PC está listo para enviar ordenes de rotación especificando la cantidad de posiciones la cual debe girar la plataforma, siendo cada posición equivalente a 18°. En la figura 3.2, se expone un ejemplo funcional en donde se envía una orden de rotación de una posición (18°) cada 2 segundos al módulo microcontrolador el cual hace uso del puerto USB "COM4".

```
1 #Se importa la libreria pySerial
2 import serial
3 #Se importa la libreria time que permite el uso de esperas
4 import time
5 #Se especifica el puerto USB a utilizar y su baudrate (Comunmente 9600)
6 ser = serial.Serial('COM4',9600)
7 #Ciclo infinito
8 while 1:
9     #Se especifica la rotación equivalente a una posición
10    rotacion = "1"
11    #Se envia la orden al modulo microcontrolador
12    ser.write(rotacion.encode())
13    #Se realiza la espera de 2 segundos
14    time.sleep(2)
```

**Figura 3.2.** Ejemplo de uso de la librería pySerial para Python 3 en donde se envían ordenes de rotaciones indefinidas de una posición (18°) cada 2 segundos.

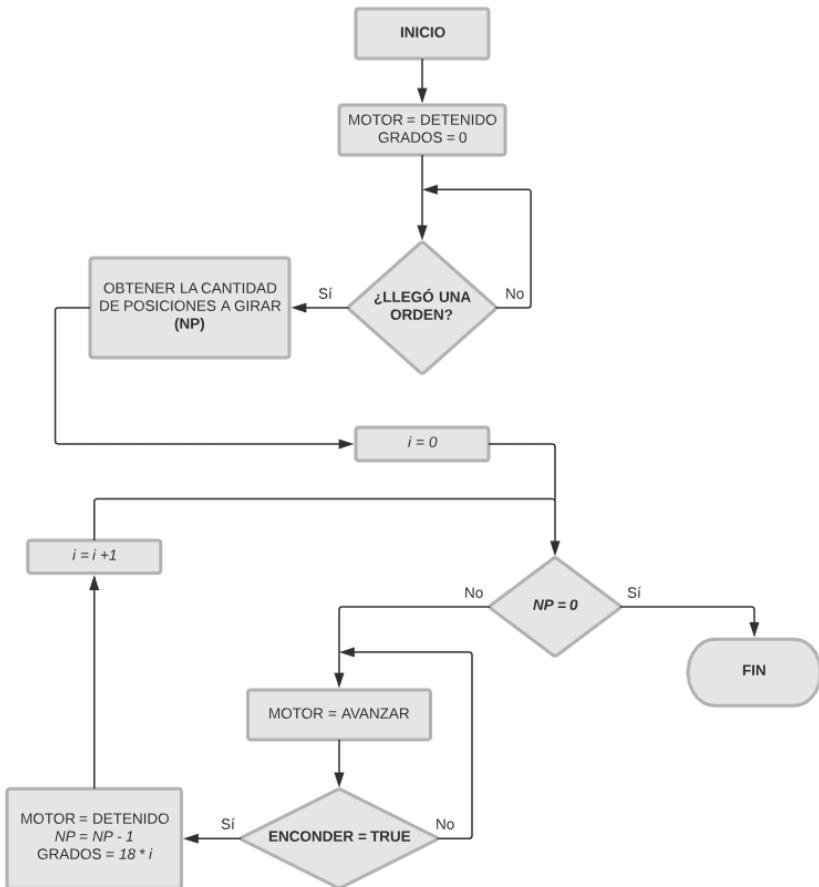
#### 3.3.2. Modulo microcontrolador

El modulo microcontrolador cumple el papel de analizar y vigilar que la orden recibida del módulo PC se ejecute en el módulo plataforma. El modulo microcontrolador está conformado principalmente por una placa de desarrollo Arduino UNO y una extensión LCD DFROBOT compatible con dicha placa como se puede observar en la figura 3.3.



**Figura 3.3.** Componentes principales del módulo microcontrolador. Placa de desarrollo Arduino UNO (izquierda) y extensión LCD DFROBOT compatible con Arduino UNO (derecha).

En la figura 3.4 se explica mediante un diagrama de flujo la metodología que rige a el modulo microcontrolador para analizar y vigilar que una orden recibida del módulo PC se ejecute adecuadamente en el módulo plataforma.

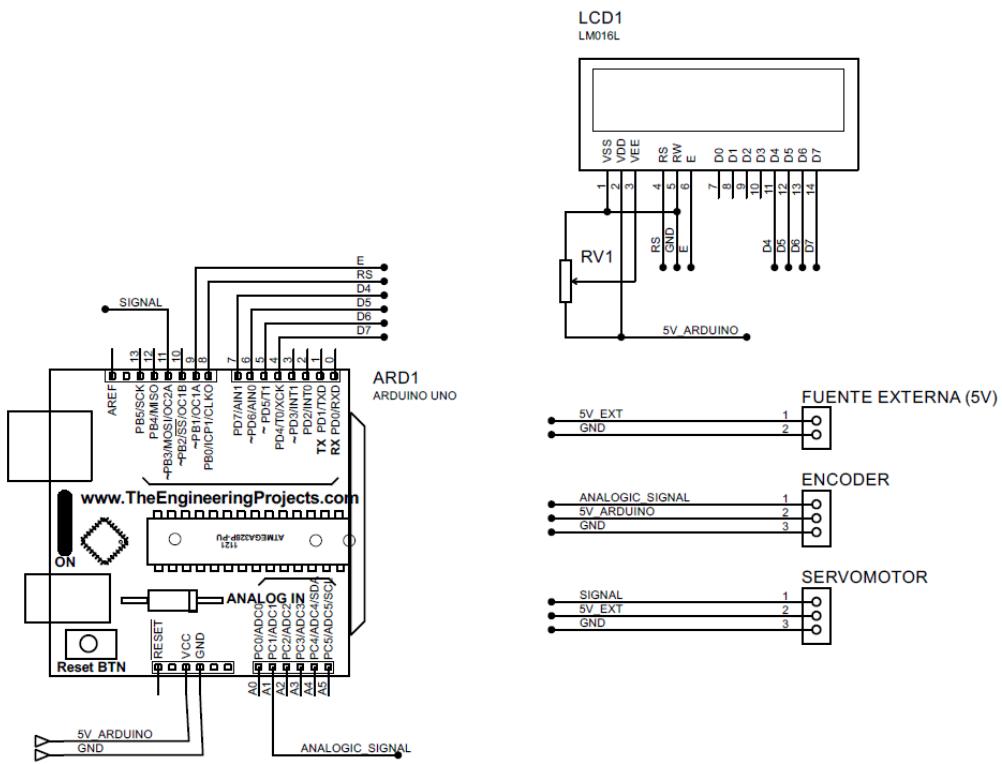


**Figura 3.4.** Diagrama de flujo sobre la metodología que rige a el modulo microcontrolador para analizar y vigilar que una orden recibida del módulo PC se ejecute adecuadamente en el módulo plataforma.

Como primera instancia con el objetivo de inicializar el sistema 360PA, se envía una orden hacia el módulo plataforma de detener el motor y se visualiza en el LCD 0 grados rotados. En el momento que el módulo microcontrolador recibe una orden de rotación desde el módulo PC, se procede a extraer de la orden de rotación la cantidad de posiciones a girar ( $NP$ ). Si la cantidad de posiciones a girar es diferente de cero, se envía una orden al módulo plataforma de encender el motor, el motor arrancará y se permanecerá encendido mientras el encoder ubicado en el mismo módulo no regrese ninguna señal de control. Cabe aclarar que la señal que controla el motor desde el modulo microcontrolador es una señal PWM (Pulse Width Modulation), señal que presenta una frecuencia aproximada de 50 Hz, un ciclo de trabajo variable que se sitúa en su valor mínimo (13.3%) al arrancar el motor y a medida que el motor permanece encendido aumenta hasta llegar a un 20% (rango del ciclo de trabajo 13.3% a 20%) con la finalidad de evitar movimientos bruscos en el motor.

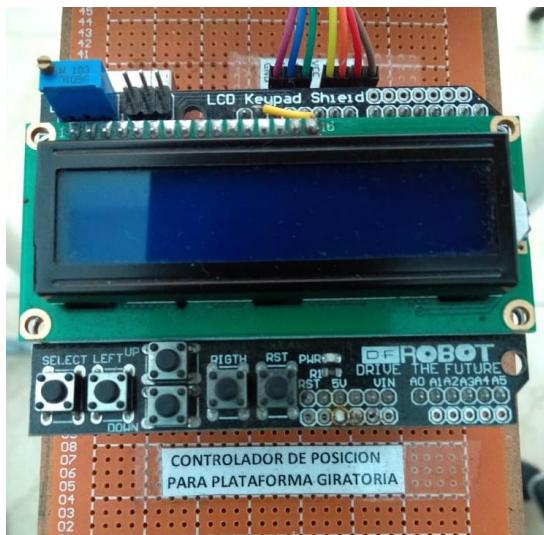
En el momento que el encoder envía su señal de control, el módulo microcontrolador envía una orden de parada hacia el motor ubicado en el módulo plataforma, actualiza en el LCD los grados rotados sumando  $18^\circ$  a la lectura anterior y le resta una unidad a la orden de rotación  $NP$  para volver a analizar si esta orden  $NP$  ya es cero. En el momento que la orden de rotación  $NP$  es equivalente a cero el algoritmo finaliza y el motor a rotado ( $18 * NP$ ) grados.

En la figura 3.5 se visualiza el diagrama esquemático del módulo microcontrolador en donde se pueden apreciar los componentes que lo conforman y las conexiones existentes entre ellos.



**Figura 3.5.** Diagrama esquemático del módulo microcontrolador.

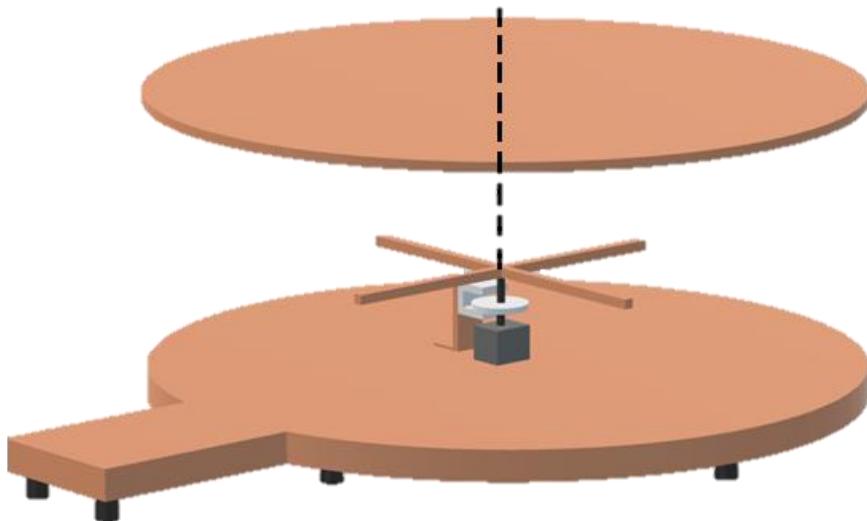
Se puede apreciar que el motor se encuentra alimentado externamente desde el modulo microcontrolador, por razones de la alta demanda de corriente que este presenta con carga ( $\sim 1.5$  A). En la figura 3.6 se aprecia el montaje que se realizó para la implementación del módulo microcontrolador.



**Figura 3.6.** Montaje del módulo microcontrolador.

### 3.3.3. Módulo Plataforma

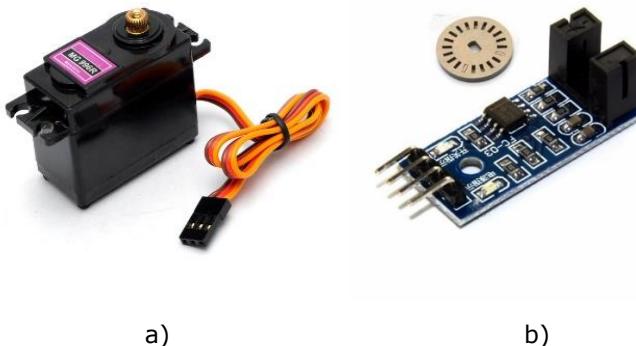
El módulo plataforma es conformado por 3 partes esenciales: la base con estructura giratoria, el motor y el encoder. La base con estructura giratoria permite la rotación de un plato de 80 cm de diámetro sobre un eje en donde irá posicionado el encoder y el motor. En la figura 3.7 se puede apreciar un modelo conceptual de la estructura giratoria (En la carpeta Anexo\_DocumentacionSoftwareRUP360PA se anexará planos detallados de la estructura giratoria). Cabe recalcar, que debido a las dimensiones del plato móvil de la estructura giratoria es recomendable el uso de objetos que no sobresalgan de él y que no presenten un peso excesivo (Max. 5 kg).



**Figura 3.7.** Modelo conceptual de la estructura giratoria

Debido a que los objetos a escanear presentan un bajo peso (Max. 5 kg) y no es necesario la extrema precisión que presenta un motor paso a paso en sus giros (en [5] se realizan rotaciones de 15°), se ha elegido para la rotación de la estructura giratoria el Servomotor Tower Pro MG996R, el cual permite rotar su eje sobre los 360 grados continuamente y presentar un par de 15 kg/cm, suficiente para el peso de los objetos dichos. En la figura 3.8.a se aprecia dicho motor.

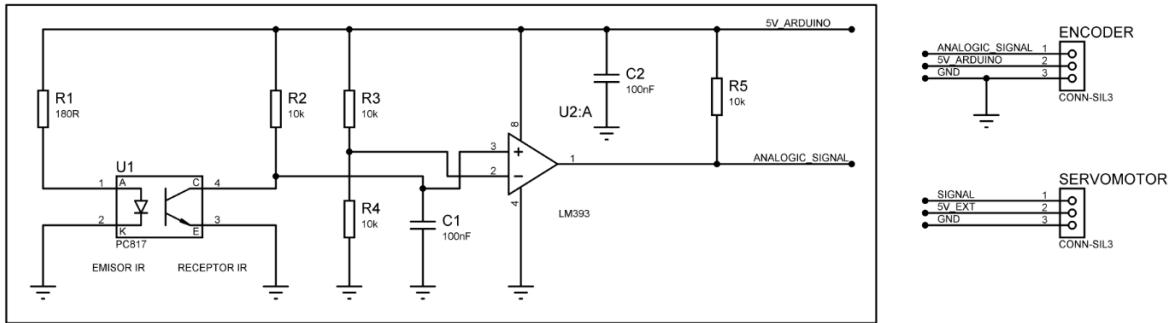
Dado que los servomotores de rotación continua tienen control mediante PWM de velocidad mas no de rotación, se ha debido implantar un encoder en el eje de movimiento del motor. El encoder seleccionado puede captar 20 pasos durante una vuelta del servomotor, lo que implica que puede controlar rotaciones de cada 18° en el eje del servomotor. En la figura 3.8.b se puede apreciar el kit encoder ADAFRUIT FC-03 que consta básicamente de un disco ranurado que permite u obstruye el paso entre emisor y un receptor infrarrojo, los cuales mediante un circuito de acondicionamiento indica si se encuentra o no alineado con una ranura del disco.



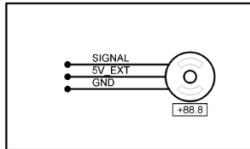
**Figura 3.8.** a) Servomotor Tower Pro MG996R,  
b) Kit módulo de encoder y disco de 20 ranuras ADAFRUIT FC-03

En la figura 3.9 podemos observar el diagrama esquemático presente en el módulo plataforma, en donde se puede observar los componentes que la conforman y la conexión existente entre ellos.

## ENCODER



## SERVOMOTOR



**Figura 3.9.** Módulo de encoder y disco de 20 ranuras ADAFRUIT.

El encoder como ya se ha mencionado cuenta con un circuito de acondicionamiento para la señal que se extrae del emisor-receptor infrarrojo como se muestra en la figura 3.9. Este circuito de acondicionamiento básicamente consta de un amplificador operacional en modo comparador, la comparación sucede entre una referencia de voltaje generada por el divisor de voltaje presente en R3 y R4 de 2.5V y el voltaje colector receptor infrarrojo. Por lo tanto, cuando no exista un flujo de corriente en la rama del receptor infrarrojo (disco ranurado obstruyendo el paso de la luz del emisor) la salida ANALOGIC\_SIGNAL se establecerá en el voltaje de saturación positivo del amplificador (5V), de lo contrario cuando exista flujo de corriente en la rama del receptor infrarrojo (disco ranurado permitiendo el paso de la luz del emisor) la salida ANALOGIC\_SIGNAL se establecerá en el voltaje de saturación negativo del amplificador (0V). De esta manera el encoder comunica mediante su señal de control ANALOGIC\_SIGNAL al módulo microcontrolador que ha ocurrido un cambio de posición en el eje del motor. En la figura 3.10 se aprecia el montaje que se realizó para la implementación del módulo plataforma conjunto al módulo microcontrolador.



**Figura 3.10.** Montaje del módulo plataforma conjunto al módulo microcontrolador.

### **3.4. CONCLUSIONES**

En este capítulo se expuso todo el diseño e implementación que tuvo el sistema 360PA, presentando cada uno de sus módulos diseñados y programados (si es el caso) para cumplir a cabalidad todos los requerimientos funcionales y no funcionales planteados. El desarrollo del módulo en el PC tuvo como lenguaje de programación Python con ayuda de la API *pySerial* mientras que el modulo microcontrolador hizo uso del lenguaje Arduino.

La API *PySerial* proporciono un correcto y fácil manejo del protocolo USB, permitiendo así cumplir los requerimientos funcionales de establecer un canal de comunicaciones entre el aplicativo DoEverything3D y el sistema 360PA. La API *PySerial* también, permitió recibir órdenes de rotación desde el aplicativo DoEverything3D mediante el puerto USB. Todas estas funcionalidades se realizaron con facilidad para hacer uso del protocolo serial, ya que se utilizó la API *PySerial* que realiza todas estas acciones de manejo de manera oculta para el usuario y el desarrollador.

Por otro lado, la lectura y escritura de señales digitales, y el manejo de señales PWM que permite la placa de desarrollo Arduino mediante su lenguaje de programación propio, dio lugar a el cumplimiento del tercer requerimiento funcional, esto es: permitir realizar rotaciones controladas de un objeto sobre sus 360 grados y visualizar el estado de la magnitud angular de la rotación.

Finalmente, en el capítulo se expone los componentes hardware utilizados para la implementación del sistema 360PA, las conexiones que se realizaron entre ellos, diagramas esquemáticos y las imágenes reales la implementación del sistema.

# CAPÍTULO 4

## 4. DESARROLLO DEL APLICATIVO DOEVERYTHING3D

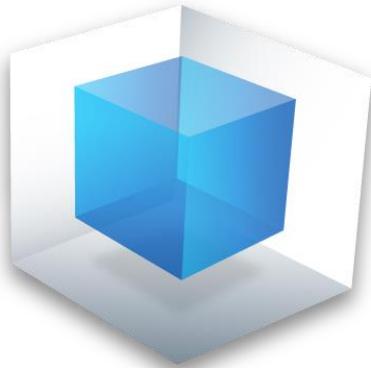
---

- 4.1. Introducción
- 4.2. Requerimientos de diseño
  - 4.2.1. Requerimientos funcionales y no funcionales
    - 4.2.1.1. Requerimientos funcionales
    - 4.2.1.2. Requerimientos no funcionales
  - 4.2.2. Diagrama conceptual
  - 4.2.3. Diagrama de clases
  - 4.2.4. Modelo de información
- 4.5. Interfaz gráfica de DoEverything3D
  - 4.5.1. Módulo de usuarios
  - 4.5.2. Módulo de configuración
  - 4.5.3. Módulo de calibración
  - 4.5.4. Módulo de escaneo
  - 4.5.5. Módulo de historial de trabajos

---

### 4.6. Conclusiones

#### 4.1. INTRODUCCIÓN



Como se vio en la sección 1.4, el sistema DoEverything3D se compone de 4 módulos principales: Modulo de adquisición, módulo de rotación, módulo de persistencia y módulo de aplicativo. Este último, siendo el cerebro del sistema, ya que es el modulo que contiene toda la lógica del sistema y tiene los permisos de controlar o usar todos los demás módulos.

En este capítulo, se enfocará principalmente en el diseño y desarrollo del módulo aplicativo perteneciente al sistema DoEverything3D. Se discutirá su desarrollo en 5 sub-módulos principales: sub-módulo de usuarios, sub-módulo de configuración, sub-módulo de calibración, sub-módulo de escaneo y sub-módulo de historial de trabajos. Cabe aclarar que el desarrollo del aplicativo DoEverything3D se llevó a cabo a través de la metodología RUP, metodología que presenta las etapas que tuvo el desarrollo del aplicativo DoEverything3D tomando como

fase de inicio los requerimientos funcionales y no funcionales, casos de uso real, diagrama conceptual y diagrama de clases. En este capítulo y el capítulo siguiente se reportarán los siguientes componentes de dicha metodología: requerimientos funcionales y no funcionales, diagrama conceptual, diagrama de clases y pruebas de integración. Mientras que en la carpeta Anexo1\_DocumentaciónSoftwareRUPDoEverything3D se encontrará la documentación RUP restante: casos de uso real, diagramas de secuencia de la aplicación.

Posteriormente, se describirá de forma detallada y puntual la manera en que se dio el cumplimiento de los requerimientos funcionales y no funcionales en cada uno de los 5 módulos que conforman el aplicativo.

#### 4.2. REQUERIMIENTOS DE DISEÑO

El desarrollo de DoEverything3D se realizó a través del proceso RUP para desarrollo software y cumpliendo con su estructura, se presentan los requerimientos funcionales y no funcionales, diagrama conceptual,

diagrama de clases y además se explicará el modulo persistencia para el buen entendimiento del funcionamiento de los 5 sub-módulos del módulo aplicativo.

En la carpeta Anexo\_DocumentaciónSoftwareRUPDoEverything3D se encontrará la documentación RUP restante: casos de uso real, diagramas de secuencia de la aplicación.

#### 4.2.1. Requerimientos funcionales y no funcionales

Los requerimientos funcionales necesarios para el desarrollo del aplicativo de DoEverything3D se muestran agrupados según a el sub-modulo que correspondan en la tabla 4.1. La tabla de requerimientos funcionales presenta tres columnas: Ref.# que enumera cada uno de los requerimientos funcionales, Funciones que describe de forma concisa el requerimiento funcional brevemente y categoría, la cual puede ser O (opcional) o E (esencial).

**Tabla 4.1.** Requerimientos funcionales de diseño para DoEverything3D.

Ref. #	Funciones	Categoría
<b>Modulo usuarios</b>		
1.0	Permitir crear un usuario.	E
2.0	Permitir acceder a una sesión de usuario.	E
<b>Modulo configuración</b>		
3.0	Permitir seleccionar y almacenar en una base de datos la configuración de operación del aplicativo por usuario.	E
<b>Modulo calibración</b>		
4.0	Permitir calibrar la cámara de profundidad del Microsoft Kinect.	E
<b>Modulo escaneo</b>		
5.0	Permitir adquirir de profundidad de un objeto mediante el Microsoft Kinect.	E
6.0	Permitir controlar el giro del objeto a modelar.	E
7.0	Permitir la conversión de las imágenes de profundidad capturadas a nubes de puntos 3D.	E
8.0	Permitir segmentar las nubes de puntos 3D a partir de la detección del plano tierra.	E
9.0	Permitir ejecutar el registro de nubes de puntos 3D segmentadas.	E
10.0	Permitir ejecutar una reducción de ruido de la nube de puntos 3D registrada.	E
11.0	Permitir generar una superficie sobre la nube de puntos 3D con ruido reducido.	E
12.0	Permitir visualizar el modelo 3D del objeto terminado.	E
13.0	Permitir almacenar en una base de datos los modelos 3D resultantes de cada usuario.	E
<b>Modulo historial de trabajos</b>		
14.0	Permitir visualizar el historial de modelos almacenados del usuario en la base de datos.	E
15.0	Permitir visualizar un modelo 3D almacenado en el historial de trabajos.	E
16.0	Permitir exportar un modelo almacenado en la base de datos del usuario bajo un nombre y en una ruta específica.	E

Por otro lado, los requerimientos no funcionales presentan las consideraciones necesarias para el desarrollo del software de DoEverything3D pero que no intervienen en el funcionamiento del mismo, estos requerimientos se muestran en la Tabla 4.2. La Tabla 4.2 presenta tres columnas: Ref.# que enumera cada uno de los requerimientos no funcionales, Descripción que describe el requerimiento no funcional brevemente y categoría, la cual puede ser O (opcional) o E (esencial).

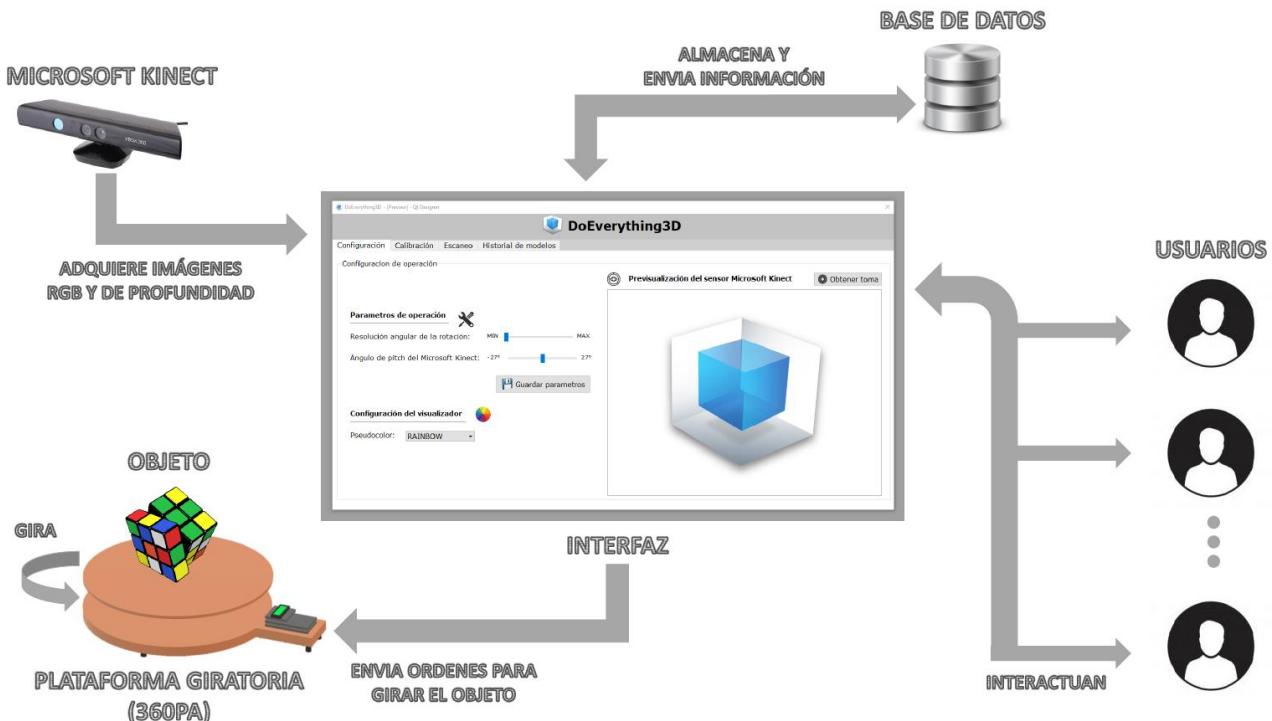
**Tabla 4.2.** Requerimientos no funcionales de diseño para DoEverything3D.

Ref. #	Descripción	Categoría
<b>SOFTWARE</b>		
1.0	Se utilizará el SDK para Kinect v1.8.	E
2.0	Se utilizarán las API's PCL v1.8.1, Open3D v0.1, libFreenect v0.57, pySerial v3.2.1, PyQt 5.10, PyMesh v0.1, numpy v1.14, mysqlConnector v2.0.1 y pyqtgraph v0.10.0.	E

3.0	Se utilizará Python 3.6.2.	E
<b>HARDWARE</b>		
4.0	Es necesario poseer el dispositivo Microsoft Kinect v1.0.	E
5.0	Es necesario poseer una plataforma móvil con rotación controlada.	E
6.0	Es necesario tener como mínimo 2GB de memoria RAM disponibles, un procesador Dual-core 2.66-GHz o superior y 2 o más puertos USB 2.0 libres para ejecutar el aplicativo.	E

#### 4.2.2. Diagrama Conceptual

La figura 4.1 es un acercamiento conceptual al sistema DoEverything3D en donde se puede observar el papel que ejerce cada uno de los 4 módulos que conforman al sistema y la asociación existente entre ellos.



**Figura 4.1.** Diagrama conceptual del sistema DoEverything3D.

Como se discutía anteriormente el sistema DoEverything3D consta principalmente de 4 grandes módulos:

- ✓ **Módulo de adquisición:** el módulo es conformado únicamente por el sensor Microsoft Kinect, encargado de realizar la adquisición de imágenes de profundidad sobre el objetivo a escanear. La orden de capturar estas imágenes de profundidad es enviada desde el modulo del aplicativo para luego proceder a enviar a este mismo modulo las imágenes capturadas.
- ✓ **Módulo de rotación (360PA - 360 PLATFORM ASSITANT):** el módulo lo conforma una plataforma giratoria basada en un microcontrolador, capaz de monitorizar y controlar la resolución angular de sus movimientos. El papel principal de esta plataforma es realizar la rotación controlada del objeto a modelar, con el objetivo de obtener información de profundidad sobre todas las caras del objeto mediante el sensor Microsoft Kinect. Las órdenes que indican cuando la plataforma debe girar y con qué resolución angular son enviadas desde el modulo del aplicativo.
- ✓ **Módulo de persistencia:** el módulo es desarrollado usando SQL, las tablas del modelo relacional se encargan de almacenar información asociada a cada usuario como son los parámetros de configuración y sus trabajos realizados. El módulo del aplicativo tiene permisos para pedir y modificar información asociada a un usuario específico.
- ✓ **Módulo de aplicativo:** el módulo del aplicativo es el “cerebro” del sistema y nos encargaremos principalmente de discutir todo su funcionamiento y los algoritmos que lo rigen en este capítulo. Este módulo se desarrolló bajo el lenguaje de programación Python y es la integración de múltiples APIs.

(PCL, Open3D, Pymesh, libFreenect, PyQt, mysqlConnector, pySerial, numpy y pyqtgraph) en conjunto con una interfaz gráfica intuitiva compuesta de 5 sub-módulos. Los cuales trabajan de manera asociada para la generación de modelos 3D a partir de las imágenes de profundidad recibidas del módulo de adquisición, y configuraciones asociadas al usuario almacenadas en el módulo persistencia. Cabe aclarar que el modulo aplicativo hace uso de los tres módulos restantes que conforman al sistema DoEverything3D (adquisición, rotación y persistencia) y además contiene toda la lógica necesaria del sistema DoEverything3D. Por lo tanto, el modulo tiene permisos para acceder y modificar la información asociada a cada usuario, alojada en el módulo de persistencia y obtener capturas de imágenes de profundidad de un objeto mediante el uso de los módulos adquisición y rotación. A continuación, se resumirá el papel que juega cada uno de los 5 sub-módulos que conforman a el módulo aplicativo:

- ✓ **Sub-Módulo de usuarios:** Este sub-módulo de usuarios tiene como objetivo permitir acceder al aplicativo a un usuario registrado bajo una sesión configurada anteriormente por el mismo y también, permitir registrar usuarios nuevos al aplicativo.
- ✓ **Sub-Módulo configuración:** Este sub-módulo tiene como función principal permitir que el usuario configure y deje establecidos parámetros de funcionamiento del aplicativo. La idea es que el usuario no se vea abrumado por la cantidad de parámetros a configurar, únicamente se dejaron a disposición de él, la resolución angular con la que rotará la plataforma del módulo rotación y el ángulo de pitch que tendrá el sensor Microsoft Kinect del módulo adquisición.
- ✓ **Sub-Módulo calibración:** Este sub-módulo tiene como función parametrizar el sensor Microsoft Kinect del módulo de adquisición. Esta parametrización se realiza con el fin obtener una nube 3D fiable a las coordenadas métricas reales a partir de las imágenes de profundidad adquiridas desde el modulo adquisición. Este sub-modulo aparte de permitir la parametrización del sensor, permite almacenar en la base de datos los resultados de la parametrización y asociarlos al usuario que la realizó.
- ✓ **Sub-Módulo escaneo:** Este sub-módulo cumple la importantísima labor de generar un modelo 3D a partir de las imágenes de profundidad capturadas de manera sincrónica por el módulo adquisición y rotación a un objeto de interés. Este módulo a su vez permite almacenar el módulo 3D generado y asociarlo a una lista de trabajos del usuario que lo ha generado.
- ✓ **Sub-Módulo historial de trabajos:** Este sub-módulo permite la visualización de los modelos 3D que tiene asociado el usuario en su lista de trabajos y además permite exportar dichos modelos en una ubicación específica del ordenador.

#### 4.2.3. Diagrama de Clases

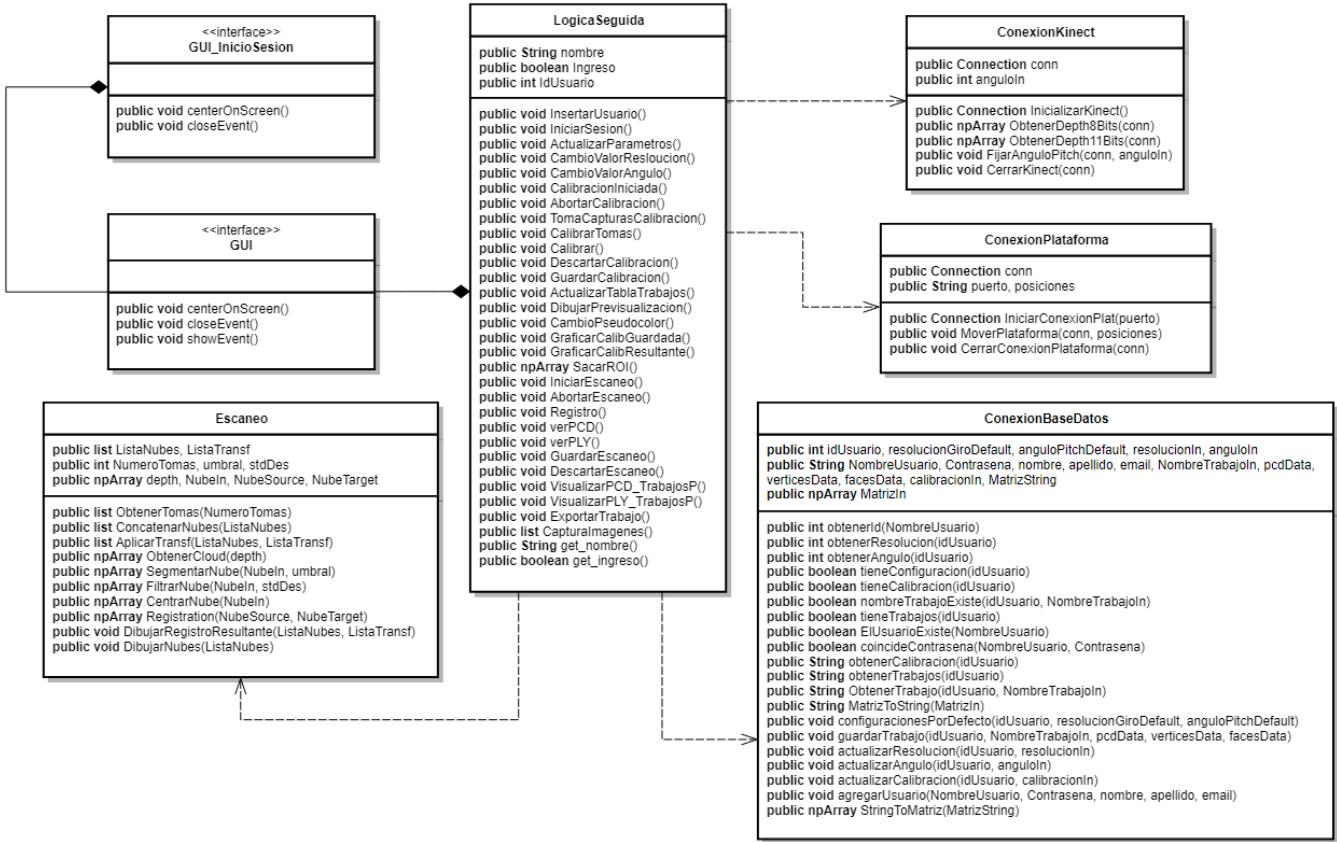
El diagrama de clases de la figura 4.2 permite apreciar la forma en la que se encuentra estructurado el software del módulo aplicativo. Se puede apreciar en el diagrama de clase dos interfaces graficas: *GUI\_InicioSesion* y *GUI*. La primera interfaz gráfica (*GUI\_InicioSesion*) se encarga de ser la parte donde el usuario interactúa para iniciar su sesión al aplicativo, o para registrarse como nuevo usuario. La segunda interfaz gráfica (*GUI*) se encarga ser la parte donde el usuario interactúa para hacer uso de los sub-módulos configuración, calibración, escaneo e historial de trabajos y se considera la interfaz gráfica principal. Cabe aclarar que ninguna de estas interfaces contiene ninguna lógica y son únicamente el medio a comunicarse con el usuario, por la tanto la interfaz principal (*GUI*) instancia la clase *LogicaSeguida* que cumple la función de atender todos los llamados de los componentes presentes en las interfaces gráficas, y comunicar las respuestas (si las hay) de los eventos generados por los usuarios hacia la interfaz gráfica correspondiente. Cabe aclarar que la clase *LogicaSeguida* hace uso de las clases *Escaneo*, *ConexionKinect*, *ConexionPlataforma* y *ConexionBaseDatos*.

La clase *ConexionKinect* tiene adjunto todos los métodos necesarios para generar una conexión y una desconexión con el sensor Microsoft Kinect, y de esa manera obtener imágenes de profundidad y también fijar el ángulo de pitch que presenta dicho sensor.

La clase *Escaneo* contiene todos los métodos necesarios para generar un modelo 3D a partir de imágenes de profundidad adquiridas por el sensor Microsoft Kinect hacia un objeto de interés.

La clase *ConexionPlataforma* contiene los métodos necesarios para poder generar una conexión, y una desconexión con el sistema 360PA y de esa manera poder enviar ordenes de rotación hacia dicho sistema.

La ultima clase *ConexionBaseDatos* tiene adjunto todos los métodos necesarios para acceder, modificar y actualizar la información asociada al usuario, como lo es: información del usuario, parámetros de operación del aplicativo, calibración asociada al usuario y trabajos asociados al usuario.



**Figura 4.2.** Diagrama de clases del sistema DoEverything3D.

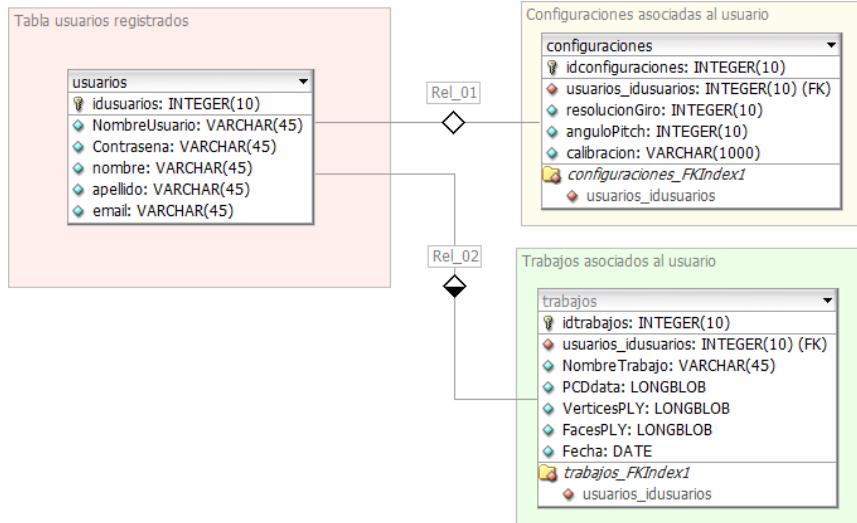
#### 4.2.4. Modelo de información

Antes de empezar a discutir los sub-módulos que conforman al módulo aplicativo de DoEverything3D, discutiremos sobre el módulo de persistencia con la finalidad de entender cuáles son los datos que el modulo aplicativo extrae, manipula o almacena en este módulo. En la figura 4.3 se puede apreciar la estructura que presenta el módulo de persistencia. El modulo persistencia está constituido por 3 tablas, las cuales son: *usuarios*, *configuraciones* y *trabajos*.

La tabla *usuarios* es la encargada de guardar la información asociada a cada usuario registrado en el sistema. Esta información es: identificador de usuario (*idUsuario*), nombre que el usuario tiene en la aplicación (*NombreUsuario*), contraseña del usuario para acceder al aplicativo (*Contrasena*), nombre real del usuario (*nombre*), apellido del usuario (*apellido*) y email del usuario (*email*). Esta tabla *usuarios* tiene una relación “uno a uno” con la tabla *configuraciones* y una relación “uno a muchos” con la tabla *trabajos*.

La tabla *configuraciones* es la encargada de alojar los parámetros de funcionamiento del aplicativo que cada usuario tiene seleccionados y también almacenar la matriz de calibración (*calibración*) que cada usuario tiene del sensor Microsoft Kinect. Los parámetros de funcionamiento que almacena la tabla *configuraciones* son: la resolución angular con la que la plataforma del módulo de rotación girará (*resolucionGiro*) y el ángulo de pitch que se fijará en el sensor Microsoft Kinect (*anguloPitch*).

La ultima tabla *trabajos* se encarga de almacenar los modelos 3D que el usuario decidió almacenar, estos modelos 3D se almacenan bajo un nombre (*NombreTrabajo*) con 3 archivos de datos (*PCDdata*, *VerticesPLY*, *FacesPLY*) y bajo una fecha específica (*Fecha*). Cabe aclarar que debido a dimensiones que presentan los archivos de datos que manejan los modelos 3D (Aprox. 30 MB) es necesario hacer uso del formato *LONGBLOB* que permite archivos máximos de 4 GB.



**Figura 4.3.** Estructura del modelo de información del sistema DoEverything3D.

### 4.3. INTERFAZ GRÁFICA DE DOEVERYTHING3D

Como ya se discutió, el modulo aplicativo del sistema DoEverything3D se encuentra dividido en 5 sub-módulos. A continuación, explicaremos el funcionamiento y la lógica que efectúa cada uno de ellos para hacer cumplimiento de los requerimientos funcionales y no funcionales planteados anteriormente.

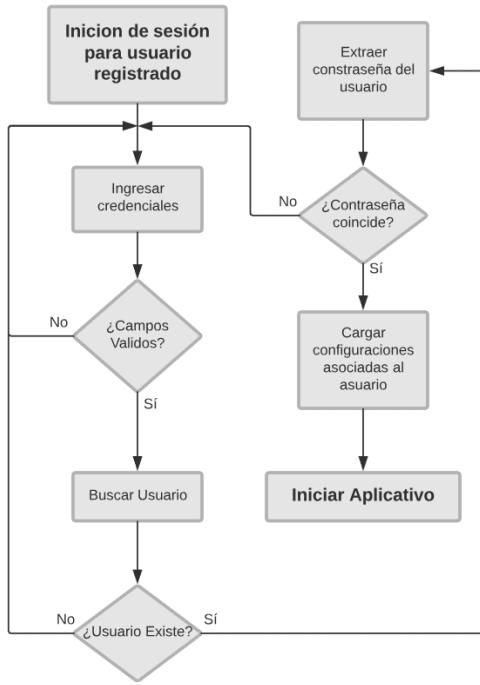
#### 4.3.1. Módulo de usuarios

Este sub-módulo de usuarios tiene como objetivos permitir acceder al aplicativo a un usuario registrado bajo una sesión configurada anteriormente por el mismo y también, permitir registrar usuarios nuevos al aplicativo. A continuación, explicaremos cuales son los algoritmos que rigen a este sub-módulo para cumplir dichos objetivos.

En la figura 4.4 podemos apreciar la interfaz gráfica con la que el usuario interactúa para iniciar sesión o para registrarse en el aplicativo.

**Figura 4.4.** a) Interfaz gráfica pestaña para inicio de sesión de usuario.  
b) Interfaz gráfica pestaña para registro de usuarios.

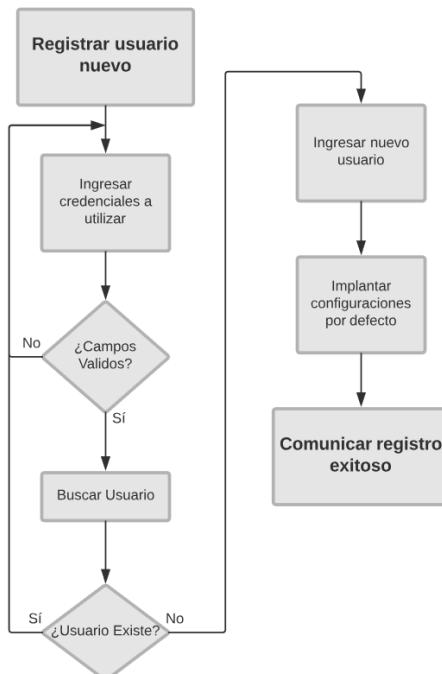
El procedimiento que se realiza para iniciar sesión a un usuario registrado se puede apreciar en la figura 4.5.



**Figura 4.5.** Procedimiento para iniciar sesión a un usuario registrado.

Como primera instancia, para iniciar sesión el usuario ingresa sus credenciales a las cuales se encuentra asociado (el nombre de usuario y contraseña) como se puede ver en la figura 4.4.a y oprime el botón "Ingresar". Luego, el aplicativo verifica que las credenciales ingresadas por el usuario no sean nulas ni tampoco presenten caracteres diferentes a letras y números, para proceder a verificar si el nombre de usuario existe en el módulo de persistencia. Si el nombre de usuario existe, se extrae del módulo de persistencia la contraseña asociada a dicho nombre, con la finalidad de verificar si la contraseña introducida por el usuario coincide con la contraseña alojada en el módulo de persistencia. Por último, si las contraseñas son equivalentes, se extraen las configuraciones asociadas al usuario y desplegar la interfaz gráfica principal del aplicativo con las configuraciones fijadas en ella.

Ahora, el procedimiento de registrar nuevos usuarios se expone en la figura 4.6.



**Figura 4.6.** Procedimiento para registrar nuevos usuarios al aplicativo.

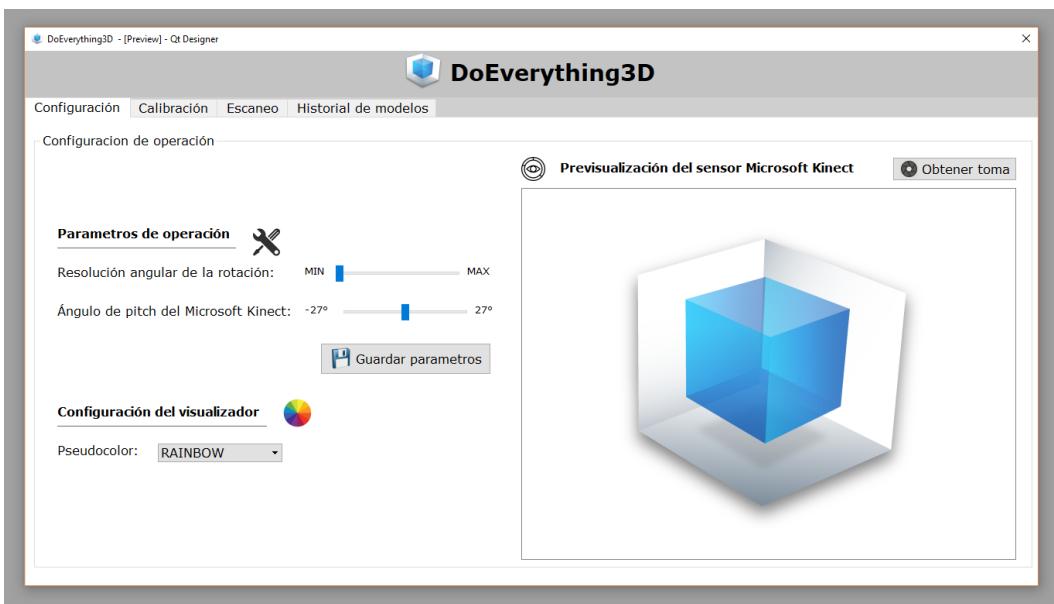
Como primer paso, para registrar un usuario nuevo se procede a ingresar las credenciales necesarias (nombre de usuario y contraseña) y se desea también las optionales (nombre, apellido, email) como se puede apreciar en la figura 4.4.b. Luego, el aplicativo verifica que las credenciales ingresadas por el usuario

no sean nulas ni tampoco presenten caracteres diferentes a letras y números, para proceder a verificar si el nombre de usuario existe en el módulo de persistencia. Si el nombre de usuario no existe, se procede a ingresar en el módulo persistencia un usuario nuevo con todas las credenciales e información suministrada. Por último, se le asocian unas configuraciones por defecto al usuario (resolución angular de la plataforma, ángulo pitch sensor Kinect y calibración) y se comunica que el registro fue exitoso.

Ahora que se ha terminado de explicar el algoritmo que rige al módulo de usuarios procedemos a discutir el funcionamiento del módulo configuración.

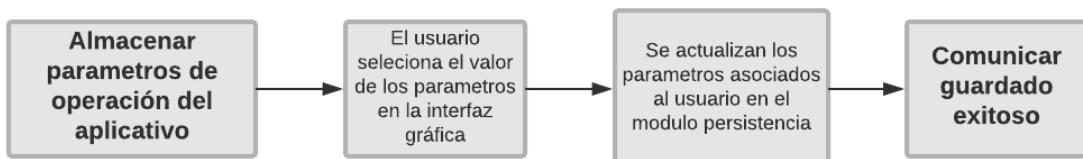
#### 4.3.2. Módulo de Configuración

En la figura 4.7 podemos apreciar la interfaz gráfica con la que el usuario interactúa para establecer y almacenar los parámetros de configuración del aplicativo. Este sub-módulo tiene como función principal permitir que el usuario configure y deje establecidos los parámetros de funcionamiento del aplicativo en el módulo persistencia. La idea es que el usuario no se vea abrumado por la cantidad de parámetros a configurar por lo tanto, únicamente se dejaron a disposición de él, la resolución angular con la que rotará la plataforma del módulo rotación y el ángulo de pitch que tendrá el sensor Microsoft Kinect del módulo adquisición. El modulo también presenta funcionalidades opcionales como es pre visualizar una imagen de profundidad desde el sensor Kinect y cambiar el falso color con la que está se visualiza.



**Figura 4.7.** Interfaz gráfica principal, pestaña configuración.

El procedimiento que se realiza para la fijación y guardado de los parámetros de funcionamiento por usuario en el módulo persistencia se visualiza en la figura 4.8.



**Figura 4.8.** Procedimiento para actualizar y almacenar los parámetros de operación del aplicativo asociado a un usuario.

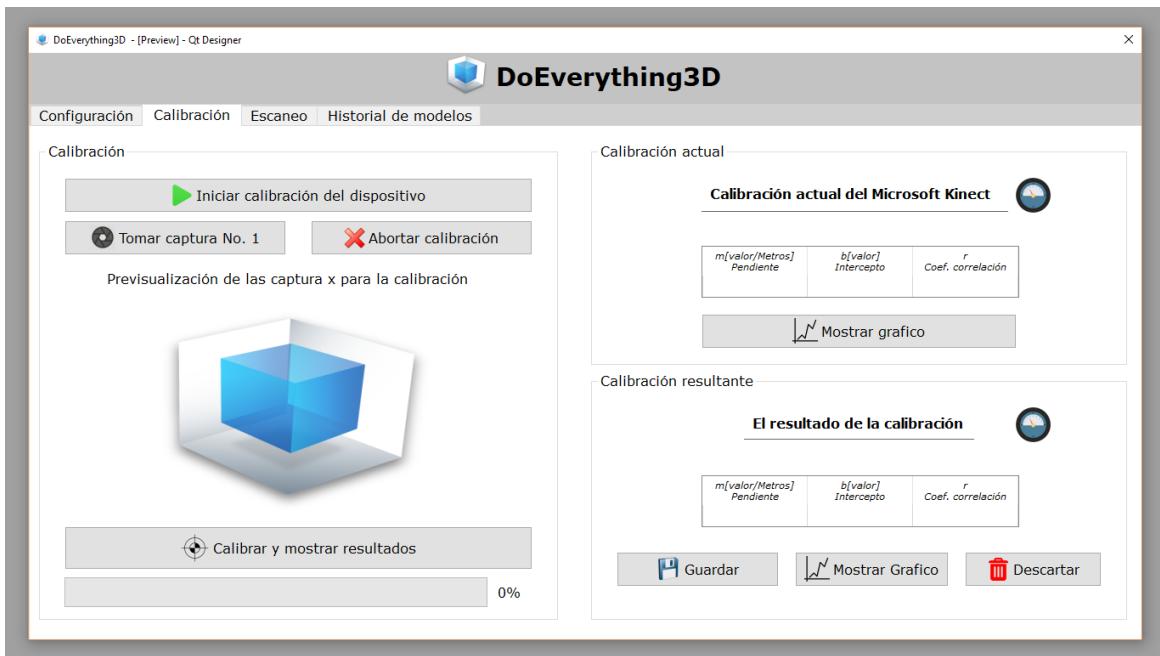
Como primera instancia, para almacenar de manera asociada a un usuario los parámetros de operación del aplicativo, el usuario debe seleccionar mediante los dos deslizadores la resolución angular de rotación de la plataforma y el ángulo de pitch del sensor Microsoft Kinect para luego presionar el botón "Guardar parámetros". Luego, el aplicativo actualiza los parámetros que están asociados el usuario por los nuevos introducidos y comunica el guardado exitoso de dichos parámetros.

Cabe recalcar que la funcionalidad opcional de pre visualización de una imagen de profundidad desde el sensor Kinect fue incluida al aplicativo, para que el usuario tuviera una ayuda a la hora de decidir en qué ángulo de pitch fijar el sensor. Es importante anotar que la funcionalidad de cambio del pseudocolor o falso color, es solo una opción visual para mejorar la apariencia de la pre visualización mencionada anteriormente.

Ahora que se ha terminado de explicar el algoritmo que rige al módulo de configuración procedemos a discutir el funcionamiento del módulo calibración.

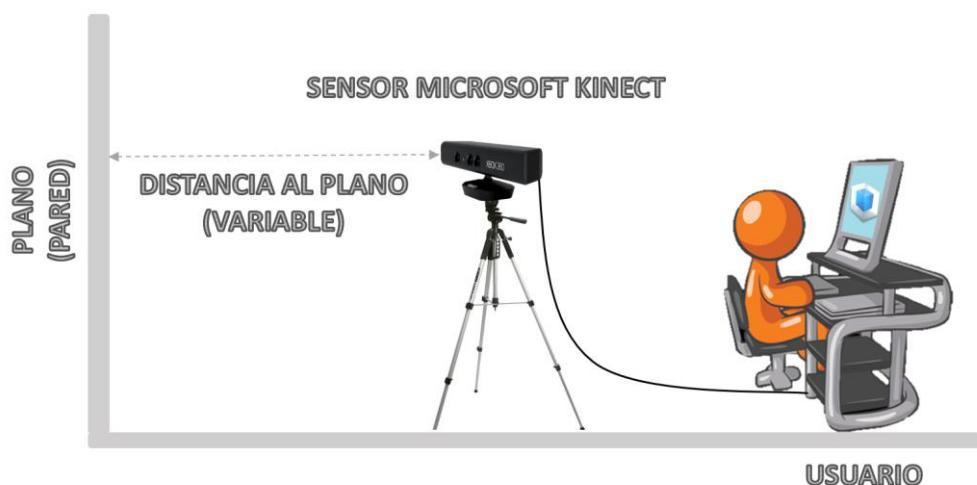
#### 4.3.3. Módulo de Calibración

En la figura 4.9 podemos apreciar la interfaz gráfica con la que el usuario interactúa para calibrar o parametrizar el sensor Microsoft Kinect. Este sub-módulo tiene como función parametrizar el sensor Microsoft Kinect del módulo de adquisición. Esta parametrización se realiza con el fin obtener una nube 3D fiable a las coordenadas métricas reales a partir de las imágenes de profundidad adquiridas desde el módulo adquisición. Este sub-módulo aparte de permitir la parametrización del sensor, permite visualizar y almacenar los resultados de la parametrización de manera asociada al usuario que la realizó. Además, el sub-módulo permite de manera opcional visualizar la calibración que actualmente tiene asociada el usuario.

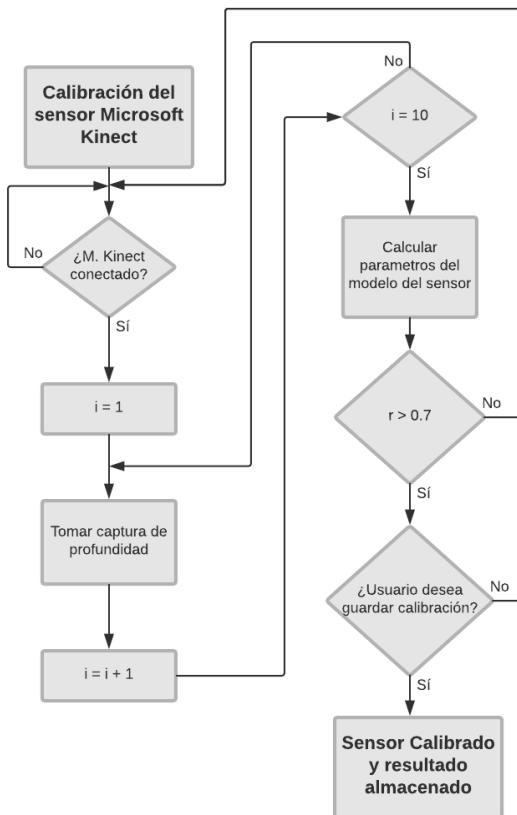


**Figura 4.9.** Interfaz gráfica principal, pestaña calibración.

la figura 4.10. muestra mediante un diagrama conceptual como el Kinect debe estar posicionado para realizar la calibración. Mientras en la figura 4.11, se aprecia mediante un diagrama de flujo el procedimiento que se realiza para la calibración del sensor Microsoft Kinect y almacenado del resultado de dicha calibración de manera asociada a un usuario.



**Figura 4.10.** Descripción gráfica de cómo debe ir posicionada el sensor Microsoft Kinect al realizar la calibración.



**Figura 4.11.** Procedimiento que se realiza para la calibración del sensor Microsoft Kinect y almacenado del resultado de dicha calibración de manera asociada a un usuario.

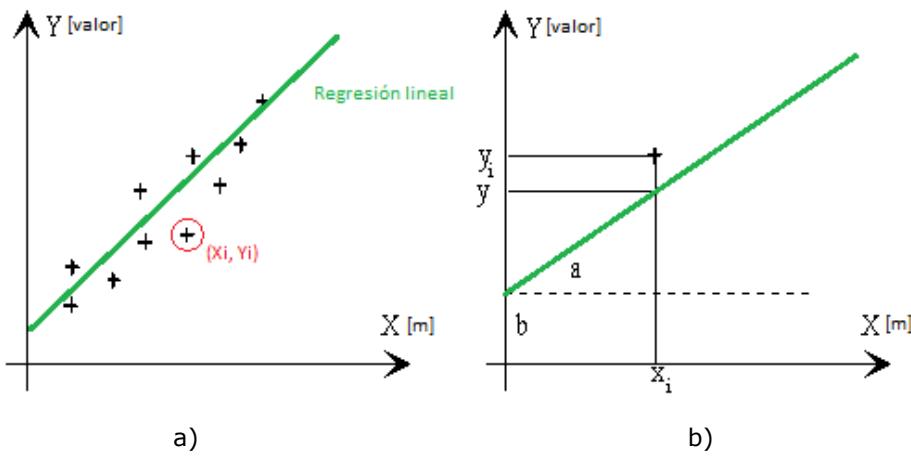
Como primera instancia, para calibrar el sensor Microsoft Kinect y almacenar su calibración de manera asociada al usuario en el módulo persistencia, el usuario debe presionar el botón “Iniciar calibración del dispositivo” en donde el aplicativo verificará si se encuentra conectado el sensor. Luego de verificar la conexión del sensor Microsoft Kinect, el usuario debe realizar una serie de 10 capturas de imágenes de profundidad a un plano ubicado a diferentes distancias del sensor (0.6 m, 0.8 m, 1.0 m, 1.2 m, 1.4 m, 1.6 m, 1.8 m, 2.0 m, 2.2 m y 2.4 m) como lo muestra la figura 4.10. Este plano puede corresponder a una pared, un tablero, etc. El aplicativo ahora procede a calcular los parámetros del sensor con la orden que usuario da pulsando el botón “Calibrar y mostrar resultados”, para lo cual se comienza extrayendo un conjunto de pixeles de cada imagen adquirida (25 pixeles ubicados en el centro de la imagen en donde debería estar capturado el plano) y los promedia, con el fin de asociar un valor de medida arrojado por sensor Microsoft Kinect con un valor de distancia métrica real. Internamente el aplicativo tendría dos vectores de 10 valores, en donde el primer vector  $X$  alojará las distancias a las que fueron capturadas las imágenes y el segundo vector  $Y$  tendrá los valores promediados de la lectura del sensor a las diferentes distancias presentes en  $X$  como se aprecia en la ecuación 4.1.

$$X = [0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4]$$

(4.1)

$$Y = [V_{0.6}, V_{0.8}, V_{1.0}, V_{1.2}, V_{1.4}, V_{1.6}, V_{1.8}, V_{2.0}, V_{2.2}, V_{2.4}]$$

Ahora el aplicativo tomará estos dos vectores presentes en la ecuación 4.1 y los relacionara entre ellos, formando pares de puntos de la forma  $(X_i, Y_i)$  con el fin de generar una regresión lineal entre los 10 pares de puntos tal como se puede apreciar en la figura 4.12.a. Esta regresión lineal regresa la ecuación de una recta que mejor ajusta a los 10 pares de puntos de la forma  $y = aX_i + b$ .



**Figura 4.12.** a) Regresión lineal de un conjunto de puntos.

b) error  $ei$  en la coordenada  $y$  entre la regresión lineal  $y$  y el punto real.

Para calcular dicha regresión, se denomina el error  $ei$  a la diferencia  $Y_i - y$ , siendo  $Y_i$  el valor real del punto, y el valor ajustado por la regresión  $y = ax_i + b$ , tal como se ve en la figura 4.12.b. El criterio de ajuste para los parámetros  $a$  y  $b$  se toma como aquél en el que la desviación cuadrática media sea mínima, es decir, debe de ser mínima la suma presente en la ecuación 4.2.

$$s = \sum_{i=1}^{10} ei^2 = \sum_{i=1}^{10} (Y_i - (ax_i + b))^2 \quad (4.2)$$

Ahora para encontrar los dos parámetros  $a$  y  $b$  se procede a encontrar el máximo y mínimo de la función  $s$  mediante la igualación de la derivada de  $s$  respecto de  $a$  y  $b$  a cero. Lo que da lugar a un sistema de dos ecuaciones con dos incógnitas como se aprecia en la ecuación 4.3 del que se despeja  $a$  y  $b$ .

$$\begin{aligned} \frac{\partial s}{\partial a} &= 0 \rightarrow a = \frac{10 \sum X_i Y_i + \sum X_i \sum Y_i}{10 \sum X_i^2 - (\sum X_i)^2} \\ \frac{\partial s}{\partial b} &= 0 \rightarrow b = \frac{\sum Y_i - a \sum X_i}{10} \end{aligned} \quad (4.3)$$

Luego de encontrar los parámetros  $a$  y  $b$  que definen el comportamiento del sensor Microsoft Kinect se halla el coeficiente de correlación  $r$  que indica la intensidad o el grado de dependencia entre los vectores  $X$  y  $Y$ . Este coeficiente de correlación lo utiliza el aplicativo para determinar si los parámetros de calibración fueron bien calculados, ya que si es muy bajo (menor a 0.7) indica muy poca relación entre los datos de los vectores  $X$  y  $Y$ . Este coeficiente se obtiene de la ecuación 4.4.

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{s_X s_Y}} \quad (4.4)$$

Siendo el numerador de la ecuación 4.4 el producto de las desviaciones de los valores  $X$  y  $Y$  respecto de sus valores medios y en el denominador tenemos el producto las desviaciones cuadráticas medias de  $X$  ( $s_X$ ) y de  $Y$  ( $s_Y$ ).

Si el aplicativo detecta que el coeficiente de correlación es mayor a 0.7, actualizara la tabla de calibración resultante mostrando el resultado de esta. Luego, permitirá al usuario visualizar la regresión lineal realizada mediante el botón "Mostrar gráfico" en el apartado de calibración resultante, y a la vez permitirá al usuario decidir si guardar o no la calibración resultante (parámetros  $a$  y  $b$ ) asociada al usuario. Si el usuario permite guardar dicha calibración, el aplicativo procederá a actualizar la calibración que tenga asociada el usuario por la nueva encontrada y comunicará el guardado exitoso de la calibración.

Además, el sub-módulo permite de manera opcional visualizar la calibración que actualmente tiene asociada el usuario si este oprime el botón "Mostrar gráfico" en el apartado de calibración actual. En donde el aplicativo extraerá los parámetros  $a$  y  $b$  asociados al usuario desde el modulo persistencia y graficará la regresión lineal asociada a dichos parámetros.

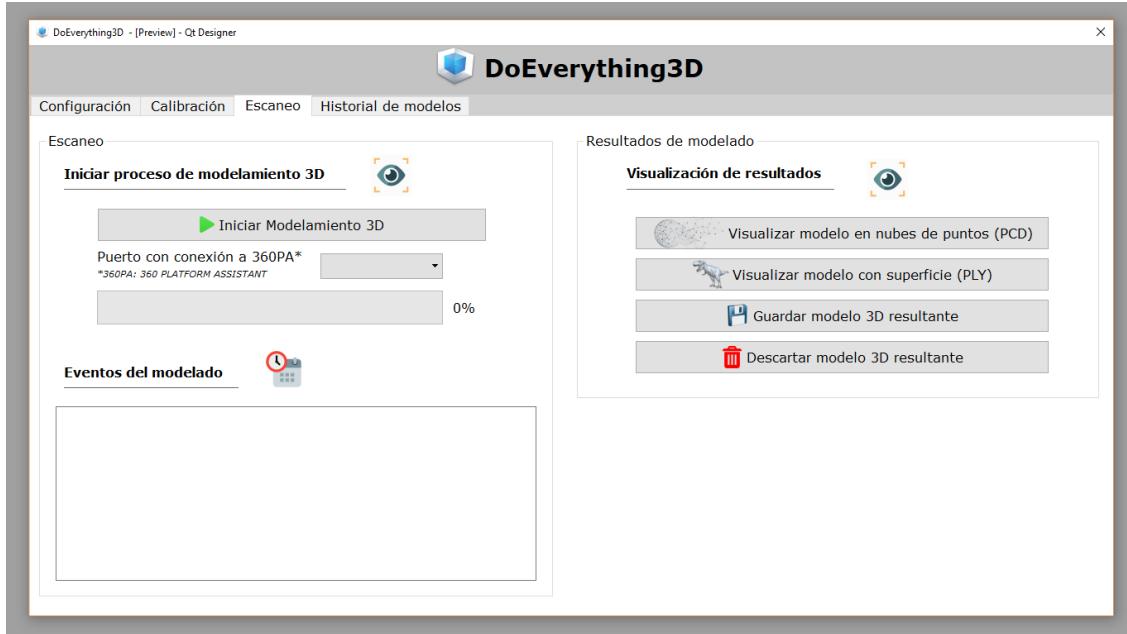
Ahora que se ha terminado de explicar el algoritmo que rige al módulo de calibración procedemos a discutir el funcionamiento del módulo configuración.

#### 4.3.4. Módulo de Escaneo

Este sub-módulo cumple la importantísima labor de generar un modelo 3D a partir de las imágenes de profundidad capturadas de manera sincrónica por el módulo adquisición y rotación a un objeto de interés.

Este módulo a su vez permite visualizar y almacenar el módulo 3D generado de manera asociada al usuario que lo ha generado.

En la figura 4.13 podemos apreciar la interfaz gráfica con la que el usuario interactúa para la generación y almacenado de modelos 3D de forma asociada a un usuario.



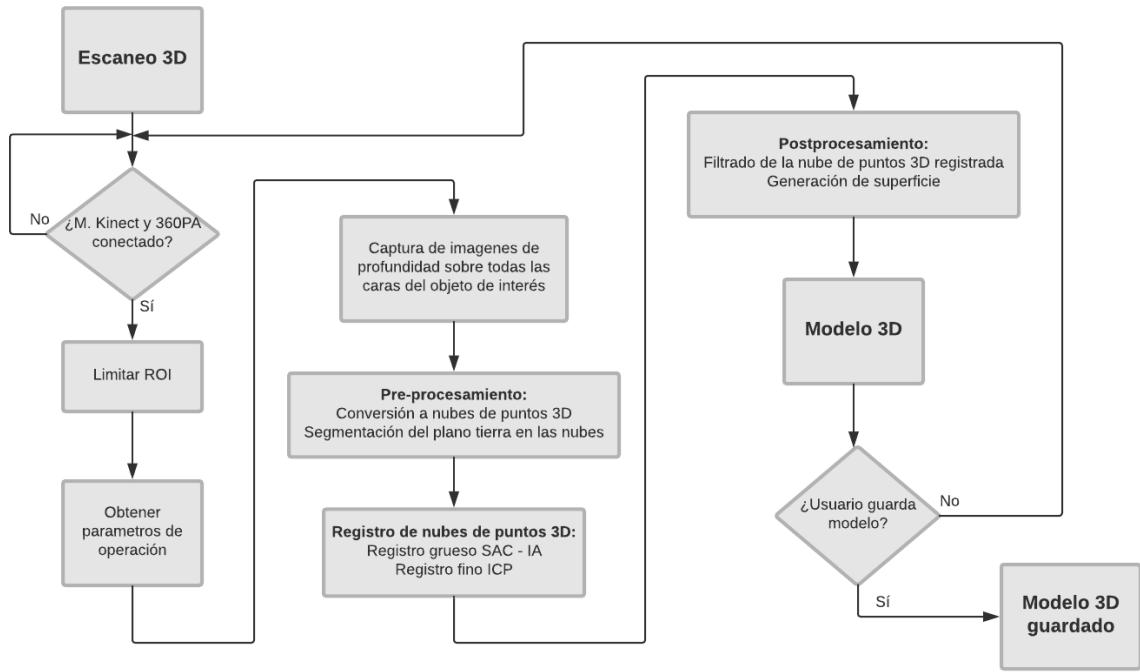
**Figura 4.13.** Interfaz gráfica principal, pestaña escaneo.

El procedimiento que se realiza la generación y almacenado de modelos 3D de forma asociada a un usuario se visualiza en la figura 4.14.

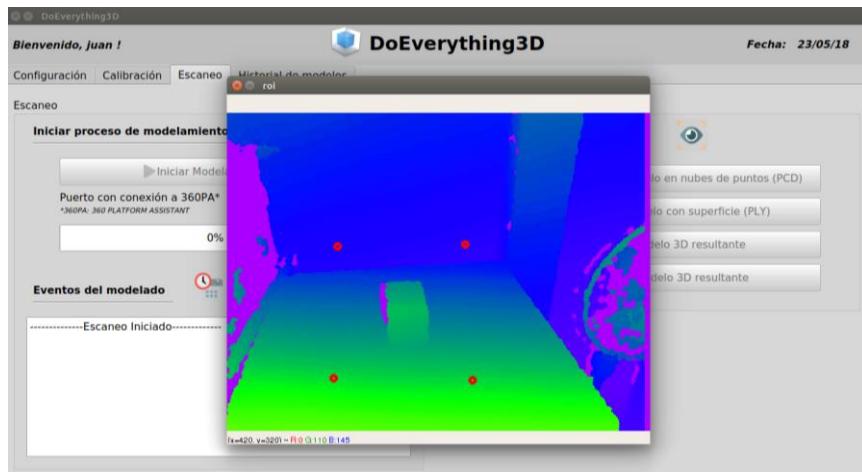
Como primera instancia, para generar un modelo 3D de un objeto de interés a partir de imágenes de profundidad tomadas a éste. El usuario presiona el botón "Iniciar modelamiento 3D" en donde el aplicativo verificará que el dispositivo Microsoft Kinect y el sistema 360PA (360 Platform Assistant) se encuentren debidamente conectados (360PA debe estar conectado al puerto seleccionado por el usuario). Si los dos dispositivos se encuentran debidamente conectados, se desplegará una ventana como en la figura 4.15. En la ventana desplegada, se podrá visualizar la escena que está capturando el sensor Microsoft Kinect en formato de imagen de profundidad, y el usuario deberá limitar la región de interés (ROI) que desea escanear (espacio ocupado por el objeto de interés). Esto el usuario lo realiza manualmente mediante la elección de 4 límites que serán demarcados por círculos rojos. Estos límites, serán usados próximamente para recortar las imágenes de profundidad que se tomarán sobre todas las vistas del objeto, asegurando que cada una de estas imágenes contenga en mayor parte información de profundidad del objeto de interés.

Luego de delimitar la ROI, el aplicativo extrae los parámetros de funcionamiento asociados a la plataforma de rotación y escaneo (resolución angular de la plataforma, ángulo pitch sensor Kinect y calibración) para comenzar el proceso de capturas de imágenes de profundidad sobre el objeto. El aplicativo con dichos parámetros, procede a fijar el ángulo de pitch del Microsoft Kinect y realizar la capturas de

20 imágenes de profundidad sobre el objeto de interés (El numero 20 es debido a resolución angular de la plataforma que un giro completo en 360PA equivale a 20 posiciones). Entre cada imagen capturada el aplicativo enviará una orden de rotación al sistema 360PA, en la orden se especificará que la cantidad de posiciones a rotar será equivalente al valor del parámetro "resolución angular de la plataforma". El aplicativo para culminar esta etapa de captura de datos recortará dichas imágenes de profundidad con 4 limitadores obtenidos de la ROI. De esta manera se conseguirá obtener imágenes de profundidad sobre los 360° principalmente del objeto de interés.



**Figura 4.14.** Procedimiento que se realiza para la calibración del sensor Microsoft Kinect y almacenado del resultado de dicha calibración de manera asociada a un usuario.



**Figura 4.15.** Ventana dedicada a limitar la ROI mediante 4 marcadores rojos (La ROI es el área contenida por los 4 marcadores).

Después de culminar la captura de las imágenes de profundidad sobre el objeto de interés, el aplicativo realiza la conversión de dichas imágenes en nubes de puntos 3D con ayuda del algoritmo visto en la sección 2.4, y el modelo del sensor extraído de los parámetros  $a$  y  $b$  en “calibración”. Para convertir una de las imágenes de profundidad capturadas a nube de puntos 3D se usan las ecuaciones 4.5.

$$Z = \frac{Vp - b}{a}, \quad X = \frac{\bar{x}(Z-f)}{f}, \quad Y = \frac{\bar{y}(Z-f)}{f} \quad (4.5)$$

Donde  $Vp$  es el valor del pixel a analizar de la imagen de profundidad,  $\bar{x}$  y  $\bar{y}$  las posiciones en la imagen de profundidad del pixel analizado,  $f$  la distancia focal del sensor Microsoft Kinect y  $(X, Y, Z)$  las coordenadas en el espacio del punto 3D calculado. Dada una imagen de profundidad de  $n * m$  pixeles se deben encontrar  $n * m$  puntos 3D, por lo que se explicará el proceso para pasar un pixel de la imagen a coordenadas 3D y se entenderá que el proceso se debe realizar en los  $n * m - 1$  pixeles restantes.

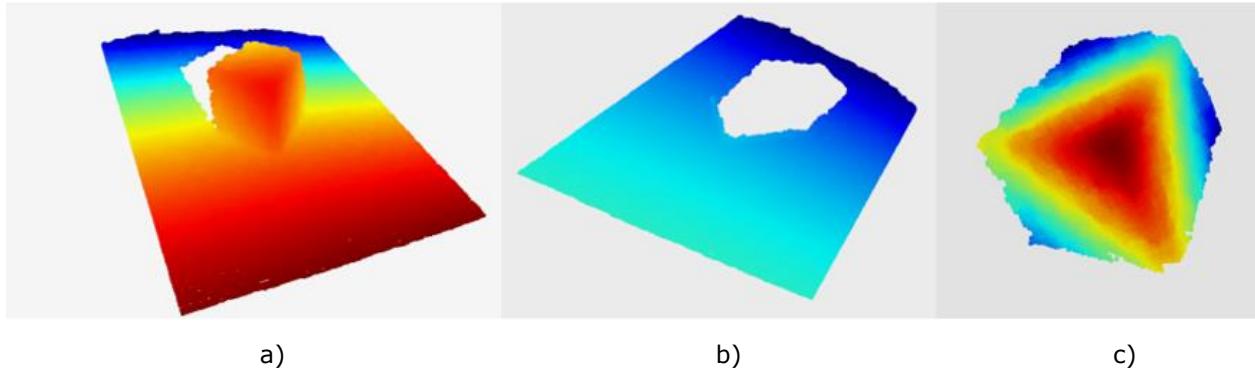
Primeramente, se procede a encontrar la coordenada  $Z$  del punto 3D, dados el valor del pixel a analizar de la imagen de profundidad ( $Vp$ ) y los parámetros  $a$  y  $b$ . Luego obtener la coordenada  $Z$  del punto 3D, se

procede a hallar las coordenadas restantes  $X$  y  $Y$ . Dichas coordenadas  $X$  y  $Y$  se encuentran mediante las posiciones en la imagen de profundidad del pixel analizado  $\bar{x}$  y  $\bar{y}$  y además de la distancia focal del sensor Microsoft Kinect ( $f$ ) que se puede obtener en el SDK del sensor otorgado por Microsoft. De esta manera, se repite el proceso en todos los pixeles faltantes y se obtiene la nube de puntos 3D con métricas reales asociadas a la imagen de profundidad.

Luego de obtener todas las nubes de puntos 3D, el aplicativo procede a eliminar el plano tierra de cada una de las nubes, con el objetivo de tener únicamente información del objeto de interés en cada una de estas nubes. Para dicho proceso se hace uso del algoritmo SIRUNS descrito en la sección 2.9 el cual lo implementa la librería PCL (Point Cloud Library). Repasando el algoritmo SIRUNS de segmentación, se tiene que este funciona mediante la agrupación de puntos que no presenten cambio de dirección en sus normales de superficie en una nube de puntos 3D. Luego, se evalúa cuál es la agrupación de puntos más grande y se extraen 3 puntos al azar de ella. Luego con los 3 puntos extraídos se procede a generar un modelo de un plano como se puede observar en la ecuación 4.6.

$$Ax + By + Cz + D = 0 \quad (4.6)$$

Con el modelo matemático del plano generado, se agrupan todos los puntos que pertenezcan a él en la nube evaluada y se asocian dichos puntos como el plano tierra de la nube. De esta manera el algoritmo logra identificar el plano tierra en la nube para posteriormente eliminarlo. En la figura 4.16 se puede observar el resultado de la identificación y eliminación del plano tierra en una nube 3D mediante el algoritmo SIRUNS donde el objeto de interés es un cubo.



**Figura 4.16.** a) Nube completa con un cubo como objeto de interés.

b) Plano tierra de la nube identificado mediante el algoritmo.

c) Nube con el objeto de interés segmentado del plano de tierra.

Luego de segmentar todas las nubes de puntos 3D el aplicativo ejecuta la etapa de registro con dichas nubes. Cabe aclarar que el registro se realiza entre par de nubes contiguas, por lo que se explicará el proceso entre un par y se entenderá que se realizará repetitivamente con los pares restantes. Por tanto, si son  $n$  nubes a registrar se tendrá como resultado  $n - 1$  transformadas provenientes de los algoritmos de registro.

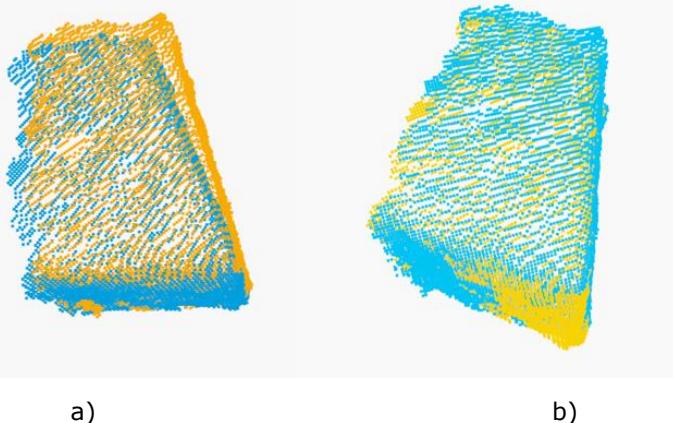
Primeramente, las nubes segmentadas pasarán por la etapa de registro grueso mediante el algoritmo SAC-IA explicado en la sección 2.7.2. Debido a que el algoritmo SAC-IA requiere las 33 características resultantes del descriptor FPFH de cada punto del par de nubes a registrar, es necesario ejecutar el descriptor en dichas nubes antes de registrarlas. Como se ya se comentó en la sección 2.6, no es recomendable ejecutar el descriptor sobre todos los puntos de la nube, ya que muchos puntos en estas contienen información irrelevante y van a perjudicar el desempeño en tiempo de ejecución del descriptor. Para solucionar este inconveniente se reducen la cantidad de puntos del par de nubes a registrar mediante vóxeles como se vio en la sección 2.6.1, para luego si ejecutar el descriptor FPFH en todos los puntos de estas nubes ya reducidas.

Ahora, con el par nubes de puntos reducidas y con sus características FPFH ya calculadas, se procede a estimar la transformación que sitúa a ambas nubes en el espacio mínimo global correcto, mediante el algoritmo SAC-IA. Los criterios de parada establecidos al algoritmo SAC-IA son: 4000000 de iteraciones realizadas o un error  $L_h$  menor a 0.075 (ver sección 2.7.2). Estos valores son establecidos en función de las estimaciones empíricas de [59]. El algoritmo SAC-IA retornará la transformación que sitúa ambas nubes en un mismo espacio global, y que a la vez servirá de punto de partida para la siguiente etapa de registro, a cargo del algoritmo ICP.

El algoritmo ICP que se ejecutará en la etapa de registro fino para disminuir pequeños errores en el registro se ha explicado su funcionamiento en la sección 2.7.3. Este algoritmo partirá de la estimación de rotación

y traslación calculada por el algoritmo SAC-IA y se ejecutará hasta cumplir los siguientes requerimientos de parada: 1000 iteraciones realizadas o un error cuadrático medio entre nubes menor a 0.0001 (ver sección 2.7.3). Estos valores nuevamente son establecidos en función de las estimaciones empíricas de [59]. Este procedimiento se realizará con todos los pares de nubes de puntos 3D contiguas asociadas al objeto de interés obteniendo lo que denominamos transformadas relativas.

El resultado del registro de dos etapas en par de nubes de puntos 3D con un cubo como objeto de interés mediante el uso de las transformadas relativas se puede observar en la figura 4.17.



**Figura 4.17.** a) Par de nubes contiguas de un cubo (Azul y amarillo) antes de registrar.

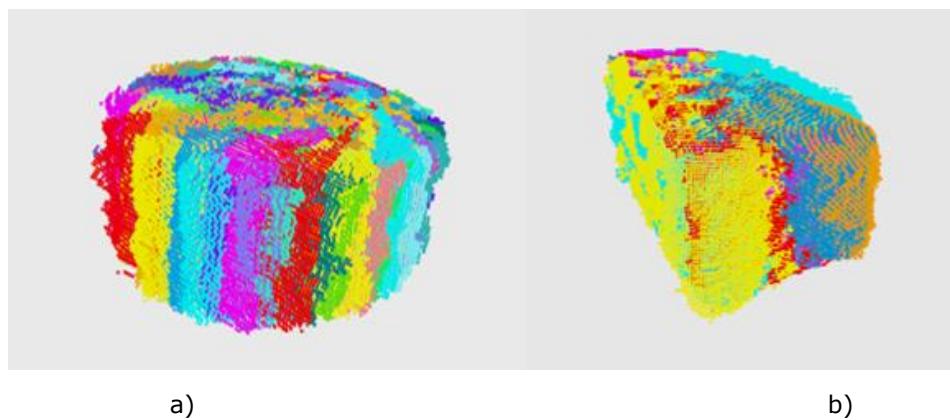
b) Par de nubes contiguas de un cubo (Azul y amarillo) luego de registrar mediante las transformadas relativas.

Las transformadas relativas resultantes del registro de un par de nubes de puntos 3D ubican a la nube desalineada en el espacio global en el que se encuentra su pareja. En nuestro caso al tener que alinear más de dos nubes de puntos no es posible ubicar en el mismo espacio global todas las nubes aplicando únicamente las transformadas relativas. Por esta razón, es necesario componer dichas transformadas relativas a medida que la nube a registrar se aleja de una nube que fijaremos como inicial  $C_0$ , dichas transformadas compuestas denominaremos transformadas globales. Por tanto, si tenemos  $n$  nubes de puntos  $C$ ,  $n-1$  transformadas relativas  $tr_j$ , las  $n-1$  transformadas globales  $tg_j$  se hallaría según la ecuación 4.7.

$$tg_j = tr_j \otimes tr_{j-1} \otimes tr_{j-2} \otimes \dots \otimes tr_0 \quad (4.7)$$

Siendo  $tr_0$  la transformación relativa resultante del registro entre la nube  $C_0$  y la nube  $C_1$  y la transformada relativa  $tr_j$  la resultante del registro entre la nube  $C_j$  y la nube  $C_{j+1}$ . Luego de obtener todas las  $n-1$  transformaciones globales, el aplicativo procede a aplicar las transformadas globales de manera secuencial a las nubes de puntos 3D que se desean registrar, y posteriores a la nube  $C_0$  (ver sección 2.7.1). Por último, el aplicativo consolida todas las  $n$  nubes registradas como una sola.

El resultado del registro de dos etapas a 20 nubes de puntos 3D con un cubo como objeto de interés mediante el uso de las transformadas globales se puede observar en la figura 4.18.

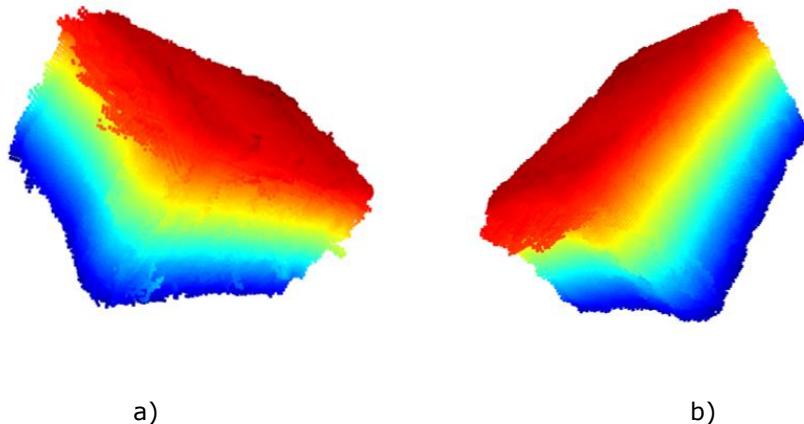


**Figura 4.18.** a) Veinte nubes contiguas de un cubo (diferentes colores) antes de registrar.

b) Veinte nubes contiguas de un cubo (diferentes colores) luego de registrar mediante las transformadas globales.

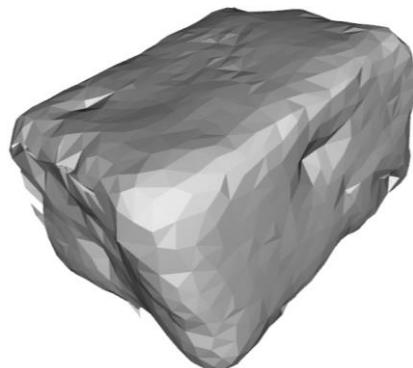
Luego de pasar la etapa de registro, el aplicativo procede a filtrar la nube de puntos 3D registrada con el objetivo de refinarla. Para ello el aplicativo, hace uso del algoritmo MLS visto en la sección 2.10

implementado por la librería PCL. En la figura 4.19 se puede apreciar el efecto de suavizada que se consigue mediante el filtrado a una nube de puntos 3D de una caja.



**Figura 4.19.** a) Nube de puntos 3D de una caja sin filtrar.  
b) Nube de puntos 3D de una caja después del filtrado.

Por último, en la generación del modelo, el aplicativo crea una superficie 3D sobre la nube de puntos filtrada siguiendo el algoritmo de triangulación de Delaunay visto en la sección 2.11 el cual lo implementa la librería Pymesh. En la figura 4.20 se puede apreciar la superficie generada por el aplicativo a la nube de puntos filtrada de la figura 4.19.b.



**Figura 4.20.** Superficie generada por el aplicativo a la nube de puntos filtrada de la figura 4.18.b.

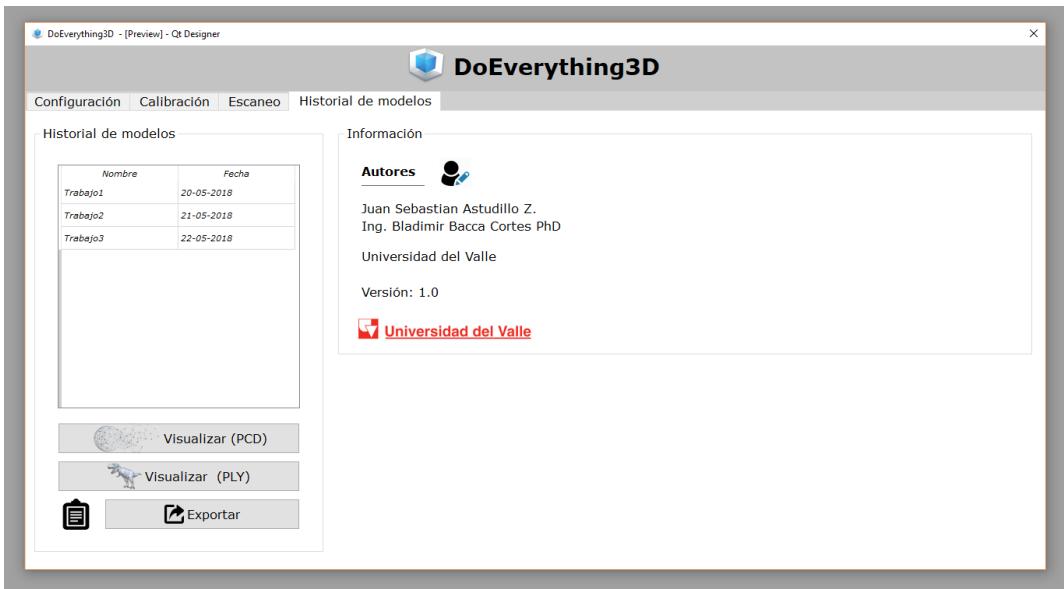
De esta manera el aplicativo, entrega a disposición del usuario un modelo 3D del objeto de interés en 2 formatos: el primer formato en nube de puntos 3D y el segundo formato compuesto por vértices y triángulos (con superficie generada). Ambos formatos del modelo 3D se encuentran disponibles para su visualización mediante los botones “Visualizar modelo en nube de puntos (PCD)” y “Visualizar modelo con superficie PLY”. También, si el usuario lo desea puede almacenar ambos formatos de nubes de puntos (ambos formatos se guardan mediante el botón “Guardar modelo 3D resultante” en el módulo persistencia. Cabe aclarar, que el aplicativo al momento de almacenar pedirá que el usuario ingrese un nombre con el que se guardará el modelo 3D en el módulo persistencia solo si este nombre ingresado es válido (que solo contenga letras y números, no se encuentra repetido en el módulo persistencia y no se nulo).

De esta manera concluimos la explicación del sub-modulo más importante del módulo aplicativo y abrimos la brecha para proceder a comentar el funcionamiento del ultimo sub-modulo, el modulo historial de trabajos.

#### 4.3.5. Módulo de Historial de trabajos

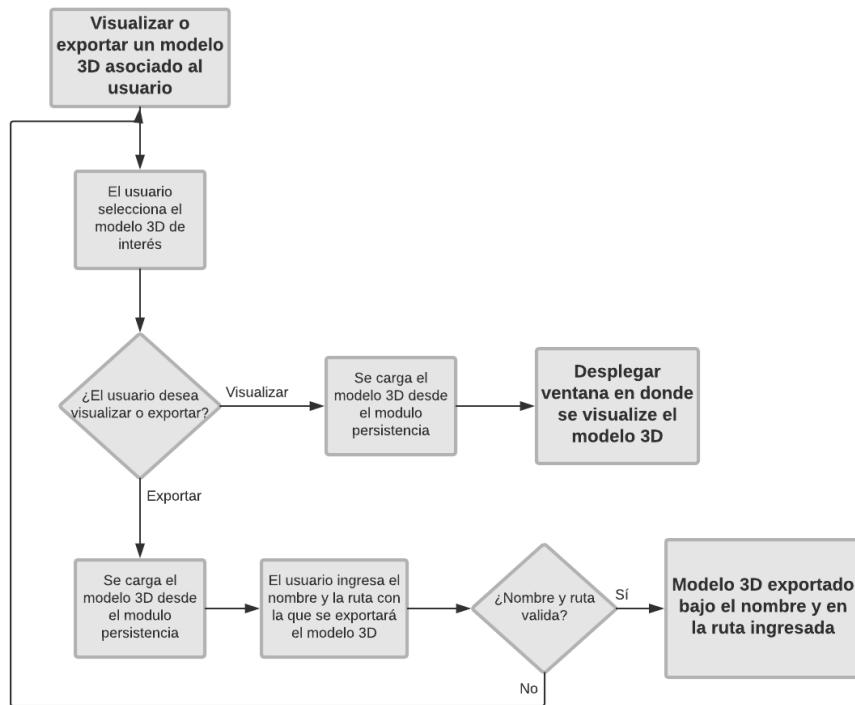
Este sub-módulo permite la visualización de los modelos 3D que tiene asociado el usuario en su lista de trabajos y además permite exportar dichos modelos en una ubicación específica del ordenador.

En la figura 4.21 podemos apreciar la interfaz gráfica con la que el usuario interactúa para la visualización de los modelos 3D asociados al usuario y también para exportar dichos modelos a una ubicación específica en el ordenador.



**Figura 4.21.** Interfaz gráfica principal, pestaña Historial de modelos.

El procedimiento que se realiza para la visualización de los modelos 3D asociados al usuario y también para exportar dichos modelos a una ubicación específica en el ordenador se visualiza en la figura 4.22.



**Figura 4.22.** Procedimiento que se realiza para la visualización de los modelos 3D asociados al usuario y también para exportar dichos modelos a una ubicación específica en el ordenador.

Primeramente, para la visualización un modelo 3D asociado a un usuario, éste deberá seleccionar el modelo 3D de interés que desea visualizar en la tabla de historial de modelos, en donde se encontraran listados todos los modelos 3D que él tenga asociado. A continuación, en función del formato que se desee visualizar del modelo 3D seleccionado, se deberá oprimir ya sea el botón “Visualizar (PCD)” o “Visualizar PLY”. Luego, el aplicativo carga dicho formato del modelo 3D seleccionado desde el modulo persistencia y lo despliega en una ventana para su correcta visualización.

Por otra parte, para realizar la exportación de modelo 3D asociado a un usuario en una ubicación específica del ordenador, el usuario deberá seleccionar el modelo 3D de interés que desea exportar en la tabla de historial de modelos, en donde se encontraran listados todos los modelos 3D que él tenga asociado. A continuación, el usuario deberá oprimir botón “Exportar” y aplicativo desplegará una ventana en donde el

usuario deberá introducir el nombre que se utilizará para exportar el modelo 3D. Luego, el aplicativo desplegará una segunda ventana en donde el usuario seleccionará la ubicación en donde será exportado el modelo 3D. Luego de esto, el aplicativo verificará que la ruta y el nombre seleccionado sean atributos válidos (no pueden ser atributos nulos y la ruta debe existir en el ordenador). Si ambos atributos son válidos, se procede a cargar ambos formatos del modelo 3D desde el módulo persistencia para ser exportados en la ubicación seleccionada y bajo el nombre seleccionado.

#### 4.4. CONCLUSIONES

En este capítulo se expuso todo el diseño e implementación que tuvo el modulo aplicativo DoEverything3D, presentando cada uno de sus sub-módulos que se han diseñados y programados para cumplir a cabalidad todos los requerimientos funcionales y no funcionales planteados. El desarrollo de dichos sub-módulos tuvo como lenguaje de programación Python con ayuda de las APIs *PCL*, *Open3D*, *Pymesh*, *PyQt*, *mysqlConnector*, *libFreenect*, *numpy* y *pySerial*.

Cabe recalcar la función de la API *PyQt*, la cual permitió realizar cada una de las interfaces graficas presentes en el aplicativo DoEverything3D, donde dichas interfaces proporcionan un fácil e intuitivo manejo del aplicativo al usuario.

Por otro lado, la API *mysqlConnector* proporcionó la conexión necesaria para acceder, modificar y actualizar los datos del el módulo persistencia, permitiendo así conjunto a la lógica presente de los sub-módulos usuarios, configuración, escaneo e historial de trabajos cumplir los siguientes requisitos funcionales: permitir crear un usuario, permitir acceder a una sesión de usuario, permitir seleccionar y almacenar en una base de datos la configuración de operación del aplicativo por usuario, permitir almacenar en una base de datos los modelos 3D resultantes de cada usuario, permitir visualizar el historial de modelos almacenados del usuario en la base de datos y permitir exportar un modelo almacenado en la base de datos del usuario bajo un nombre y en una ruta específica.

Por su parte la API *libFreenect* permite el control del ángulo de pitch y de la adquisición (Imágenes RGB, Imágenes de profundidad y señales sonaras) del sensor Microsoft Kinect mediante el puerto USB, la posibilidad de obtener imágenes de profundidad y fijar el ángulo de pitch de dicho sensor e conjunto a la lógica del sub-módulo escaneo dan pie al cumplimiento del requerimiento funcional de permitir adquirir de profundidad de un objeto mediante el Microsoft Kinect.

También, la API matemática *numpy* permitió el manejo de matrices, vectores, la realización de regresiones lineales y demás operaciones y procedimientos matemáticos. Dichos procedimientos dieron lugar conjunto a la lógica del sub-módulo configuraciones y escaneo dar cumplimiento de los requerimientos funcionales de: permitir calibrar la cámara de profundidad del Microsoft Kinect y permitir la conversión de las imágenes de profundidad capturadas a nubes de puntos 3D.

La funcionalidad de conexión mediante USB con el sistema 360PA la provee la API *pySerial*, dicho sistema cumple la función de recibir órdenes de rotación desde el aplicativo DoEverything3D, más en concreto desde el sub-módulo escaneo. Esta funcionalidad prestada por la API *pySerial* en conjunto con la lógica inmersa en el sub-módulo escaneo permite el cumplimiento del requerimiento funcional de permitir controlar el giro del objeto a modelar.

Ahora las APIs *PCL* (Point Cloud Library), *Open3D* y la ya mencionada *numpy*, permitieron el manejo, la realización de operaciones geométricas y espaciales con las nubes de puntos 3D. Dichas APIs designadas a la ejecución en el aplicativo de algoritmos como son: SAC-IA, FPFH, ICP, SIRUNS, MLS. Estos algoritmos implementados por las APIs *PCL*, *Open3D* y *numpy* conjunto a la lógica inmersa en el sub-módulo escaneo dan pie al cumplimiento a los siguientes los requisitos funcionales: permitir segmentar las nubes de puntos 3D a partir de la detección del plano tierra, permitir ejecutar el registro de nubes de puntos 3D segmentadas y permitir ejecutar una reducción de ruido de la nube de puntos 3D registrada.

Se utilizó la funcionalidad de la API *Pymesh* de poder generar superficies sobre nubes de puntos 3D mediante el algoritmo de triangulación de Delaunay, en conjunto con la lógica presente en el sub-módulo escaneo para dar cumplimiento al requisito funcional de permitir generar una superficie sobre la nube de puntos 3D con ruido reducido.

La API *Open3D* también presenta la funcionalidad de visualización tanto nubes de puntos 3D como de superficies generadas sobre nubes 3D. Esta funcionalidad es crucial, ya que conjunto a la lógica que aplica los sub-módulos escaneo e historial de trabajos se da el cumplimiento de los dos últimos requerimientos funcionales: permitir visualizar el modelo 3D del objeto terminado y permitir visualizar un modelo 3D almacenado en el historial de trabajos.

Cabe aclarar que a diferencia de la metodología propuesta en la sección 2.1, la metodología implementada en el módulo aplicativo prescinde de la utilización del algoritmo de alineación global. Esto es debido a que en diferentes modelos 3D generados por la metodología explicada en este capítulo no se evidencia la necesidad de uso de este algoritmo (primera y última nube de puntos bien registradas) y la implementación de este reduciría el rendimiento del aplicativo a la hora de la generación de modelos 3D (hablando de tiempo de computo).

# CAPÍTULO 5

## 5. PRUEBAS Y RESULTADOS

---

### 5.1. Introducción

### 5.2. Pruebas de integración

5.2.1. Registro de usuario, inicio de sesión y configuración de parámetros de funcionamiento del aplicativo

5.2.2. Calibración del sensor Kinect, escaneo 3D de un objeto y visualización del modelo 3D resultante.

5.2.3. Almacenamiento de un modelo 3D, visualización y exportado de un modelo 3D del historial de trabajos.

### 5.3. Pruebas de campo

5.3.1. Análisis de error de modelado

5.3.2. Extracción de modelos de diferentes objetos

### 5.4. Conclusiones

---

#### 5.1. INTRODUCCIÓN



En este capítulo se pretende mostrar el funcionamiento e integración de los sub-módulos que conforman el aplicativo DoEverything3D detallado en el capítulo 4, así como el sistema 360 Platform Assistant (360PA) que se ha detallado en el capítulo 3. Luego, se procederá a evaluar el desempeño de los modelos 3D generados por el sistema DoEverything3D, y además mostrará una muestra de diferentes objetos escaneados mediante el sistema.

A lo largo de este capítulo se describirán las pruebas realizadas y se analizará detalladamente el comportamiento del sistema en cada una de éstas. Esto con el fin de cumplir todos los requerimientos funcionales del aplicativo DoEverything3D (Ver sección 4.2.1) y del sistema 360PA (Ver sección 3.2.1).

Los resultados de las pruebas realizadas consisten en integración y cuantitativas. Las primeras son capturas de pantallas de la aplicación en funcionamiento y capturas fotográficas para evidenciar el comportamiento del sistema. Mientras que las pruebas cuantitativas, principalmente consisten en la obtención del error que contienen los modelos 3D generados por el sistema DoEverything3D y la muestra mediante captura de modelos 3D de distintos objetos generados por el mismo sistema.

#### 5.2. PRUEBAS DE INTEGRACIÓN

Las pruebas de integración tienen el objetivo de evaluar por conjunto las funcionalidades del sistema. Estas pruebas se encuentran separadas según lo expuesto en los requerimientos funcionales. Para la exposición de cada prueba se utiliza una tabla estructurada de acuerdo a lo exigido por la metodología RUP, de la siguiente manera:

- ✓ *Requerimientos*: Nombres de los requerimientos funcionales evaluados en la prueba.
- ✓ *Tipo de prueba*: Tipo de prueba implementada.
- ✓ *Hardware requerido*: Hardware implementado en la prueba.
- ✓ *Software requerido*: Software implementado en la prueba.
- ✓ *Objetivo del requerimiento*: Objetivo de los requerimientos implementados.
- ✓ *Objetivo de la prueba*: Objetivo de la prueba en el sistema.
- ✓ *Datos necesarios para la prueba*: Datos necesarios para la prueba del sistema.
- ✓ *Procedimiento de la prueba*: Procedimiento para la realización de la prueba.

- ✓ *Resultado esperado*: Resultado esperado de la prueba.
- ✓ *Resultado obtenido*: Resultado obtenido de la prueba.
- ✓ *Comentarios*: Aspectos a tener en cuenta.

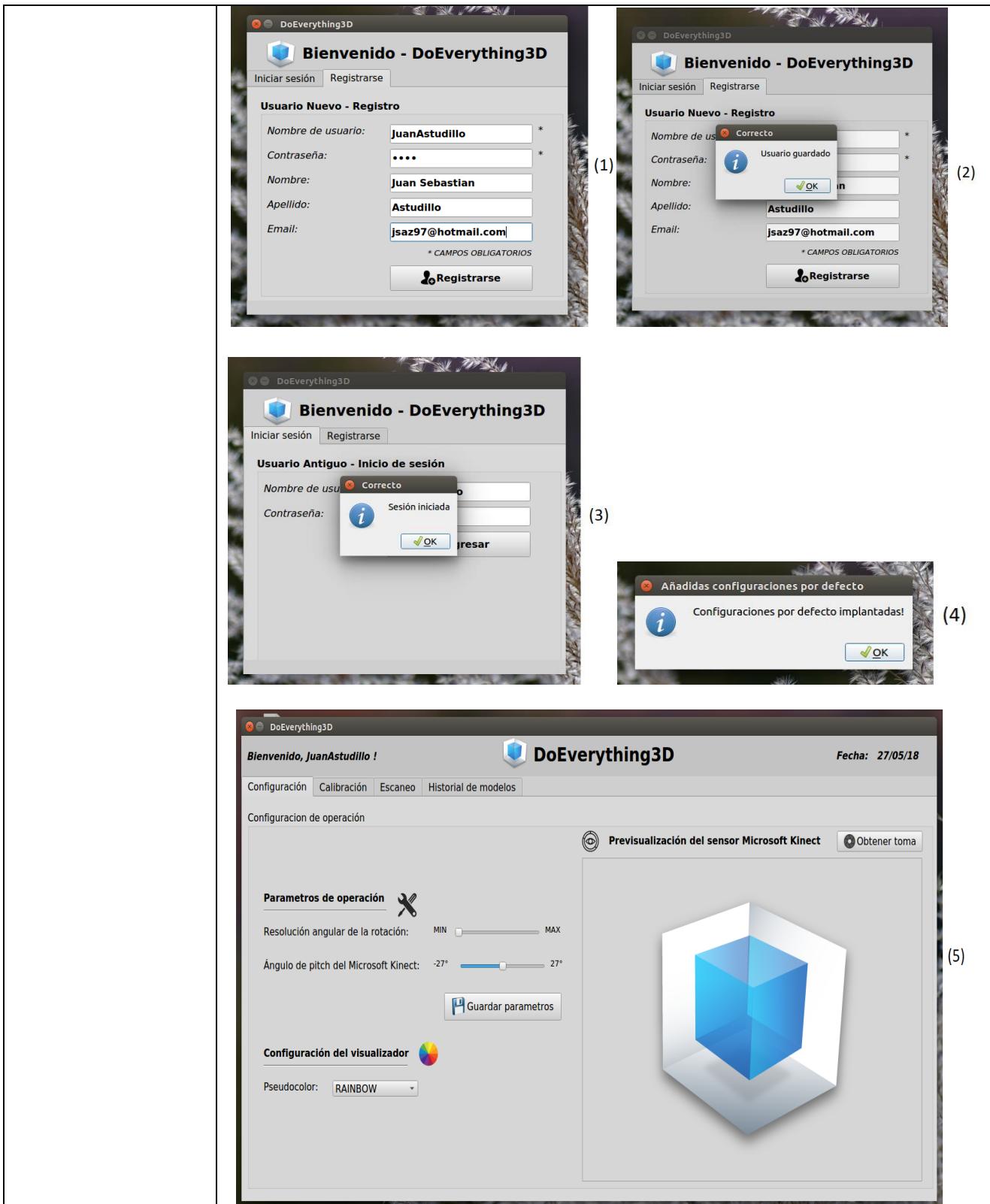
Luego de definir la estructura de las tablas de pruebas funcionales, empezaremos a discutir sobre estas y sus resultados.

### *5.2.1. Registro de usuario, inicio de sesión y configuración de parámetros de funcionamiento del aplicativo*

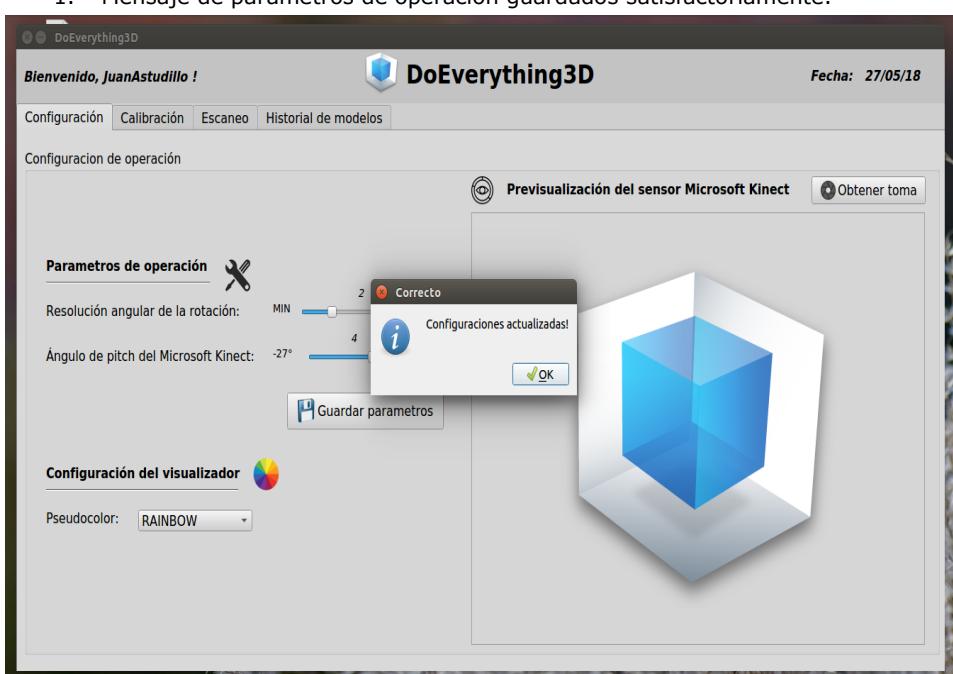
La prueba de integración correspondiente al registro de usuario, inicio de sesión y configuración de parámetros de funcionamiento del aplicativo, está destinada a corroborar que un nuevo usuario pueda registrarse en el sistema, iniciar sesión en este, decidir los parámetros de funcionamiento del mismo y almacenar dichos parámetros de forma asociada a él. Esta prueba junto con los resultados se evidencia en la tabla 5.1.

**Tabla 5.1.** Prueba de integración: Registro de usuario, inicio de sesión y configuración de parámetros de funcionamiento del aplicativo

<b>PRUEBA DE INTEGRACIÓN – REGISTRO DE USUARIO, INICIO DE SESIÓN Y CONFIGURACIÓN DE PARÁMETROS DE FUNCIONAMIENTO DEL APLICATIVO</b>	
<b>Requerimientos</b>	
<b>Tipo de Prueba</b>	Integración.
<b>Hardware Requerido</b>	Computador portátil.
<b>Software Requerido</b>	DoEverything3D (Python 3.5.2, mysqlConnector, PyQt) y XAMPP 7.2.5.
<b>Objetivo del requerimiento</b>	Probar que un nuevo usuario pueda registrarse en el sistema, iniciar sesión en este, decidir los parámetros de funcionamiento del mismo y almacenar dichos parámetros de forma asociada a él.
<b>Registro e inicio de sesión de un usuario en el aplicativo</b>	
<b>Objetivo de la prueba</b>	Verificar que se pueden ingresar usuarios nuevos al sistema y que dichos usuarios puedan ingresar a él.
<b>Datos de la Prueba</b>	Nombre de usuario, Contraseña, Nombre, Apellido y Email.
<b>Procedimiento</b>	<ol style="list-style-type: none"> <li>1. Iniciar el aplicativo DoEverything3D.</li> <li>2. Dirigirse a la pestaña “Registrarse” y llenar los campos solicitados de información del usuario (Nombre de usuario, Contraseña, Nombre, Apellido, Email).</li> <li>3. Presionar el botón “Registrarse”.</li> <li>4. Dirigirse a la pestaña “Iniciar sesión” e ingresar en los campos solicitados sobre la información del usuario (Nombre de usuario y contraseña con la que se ha registrado anteriormente).</li> <li>5. Presionar el botón “Ingresar”.</li> </ol>
<b>Resultado Esperado</b>	<ol style="list-style-type: none"> <li>1. Despliegue de la interfaz gráfica asociada al sub-módulo de usuarios del aplicativo DoEverything3D</li> <li>2. Mensaje de registro exitoso.</li> <li>3. Mensaje de inicio de sesión exitoso.</li> <li>4. Mensaje de configuraciones por defecto implantadas.</li> <li>5. Despliegue de la interfaz principal del aplicativo DoEverything3D.</li> </ol>
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia de resultados obtenidos en las imágenes:</p> <ol style="list-style-type: none"> <li>1. Despliegue e ingreso de datos de la interfaz gráfica asociada al sub-módulo de usuarios del aplicativo DoEverything3D.</li> <li>2. Mensaje de registro exitoso.</li> <li>3. Mensaje de inicio de sesión exitoso</li> <li>4. Mensaje de configuraciones por defecto implantadas.</li> <li>5. Despliegue de la interfaz principal del aplicativo DoEverything3D.</li> </ol>



<b>Comentarios</b>	Para la correcta realización de esta prueba, el usuario al momento de registrarse debe escoger un "Nombre de usuario" que no exista registrado en el aplicativo y al momento de iniciar sesión sus credenciales deben ser iguales a la que uso en el momento de registrarse.
<b>Configuración y almacenado de los parámetros de operación del aplicativo</b>	
<b>Objetivo de la prueba</b>	Verificar que el usuario pueda modificar y almacenar de manera asociada a él, los parámetros de operación del sistema.
<b>Datos de la Prueba</b>	Resolución angular de la rotación y ángulo de pitch del Microsoft Kinect.
<b>Procedimiento</b>	1. Dirigirse a la pestaña de "Configuración" en la interfaz gráfica principal del aplicativo.

	<ol style="list-style-type: none"> <li>2. Modificar los parámetros de operación (Resolución angular de la rotación y ángulo de pitch del Microsoft Kinect) mediante los deslizadores correspondientes.</li> <li>3. Presionar el botón "Guardar Parámetros".</li> </ol>
<b>Resultado Esperado</b>	1. Mensaje de parámetros de operación guardados satisfactoriamente.
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia el resultado obtenido en la imagen:</p> <ol style="list-style-type: none"> <li>1. Mensaje de parámetros de operación guardados satisfactoriamente.</li> </ol> 
<b>Comentarios</b>	No hay comentarios necesarios para la correcta realización de esta prueba.

En la tabla 5.1 se puede visualizar las capturas de pantallas respectivas al procedimiento realizado tanto como para registro e inicio de sesión de un usuario nuevo como para la fijación y almacenado de los parámetros de funcionamiento del aplicativo, y los respectivos resultados obtenidos. En el apartado de registro e inicio de sesión de un usuario nuevo, es expuesta la captura (1) en donde se evidencia el despliegue e ingreso de los datos de un nuevo usuario en la interfaz gráfica asociada al sub-módulo de usuarios del aplicativo DoEverything3D. Si el registro se ha concluido de manera exitosa (Nombre de usuario不存在 en la base de datos) el aplicativo retorna el mensaje visto en la captura (2). Luego es expuesto en la captura (3) el mensaje que arroja el aplicativo cuando se realiza el inicio de sesión del nuevo usuario mediante las credenciales correctas. Luego en (4) se despliega un mensaje comunicando que las configuraciones por defecto fueron implantadas al usuario. En el apartado de fijación y almacenado de los parámetros de funcionamiento del aplicativo, es expuesta la captura (1) que evidencia el mensaje que despliega el aplicativo al momento en el que el usuario guarda sus nuevos parámetros de operación. Este procedimiento evidencia el correcto cumplimiento del requerimiento planteados en dicha prueba.

### 5.2.2. Calibración del sensor Kinect, escaneo 3D de un objeto y visualización del modelo 3D resultante.

La prueba de integración correspondiente a la calibración del sensor Kinect, escaneo 3D de un objeto y visualización del modelo 3D resultante, está destinada a corroborar que el aplicativo permita realizar la calibración del sensor Microsoft Kinect y almacenar su resultado de manera asociada al usuario. Para luego verificar si dicha calibración conjunta a los parámetros de operación y al sistema 360PA permiten generar un modelo 3D de un objeto de interés. Esta prueba junto con los resultados se evidencia en la tabla 5.2.

**Tabla 5.2.** Prueba de integración: Calibración del sensor Kinect, escaneo 3D de un objeto y visualización del modelo 3D resultante.

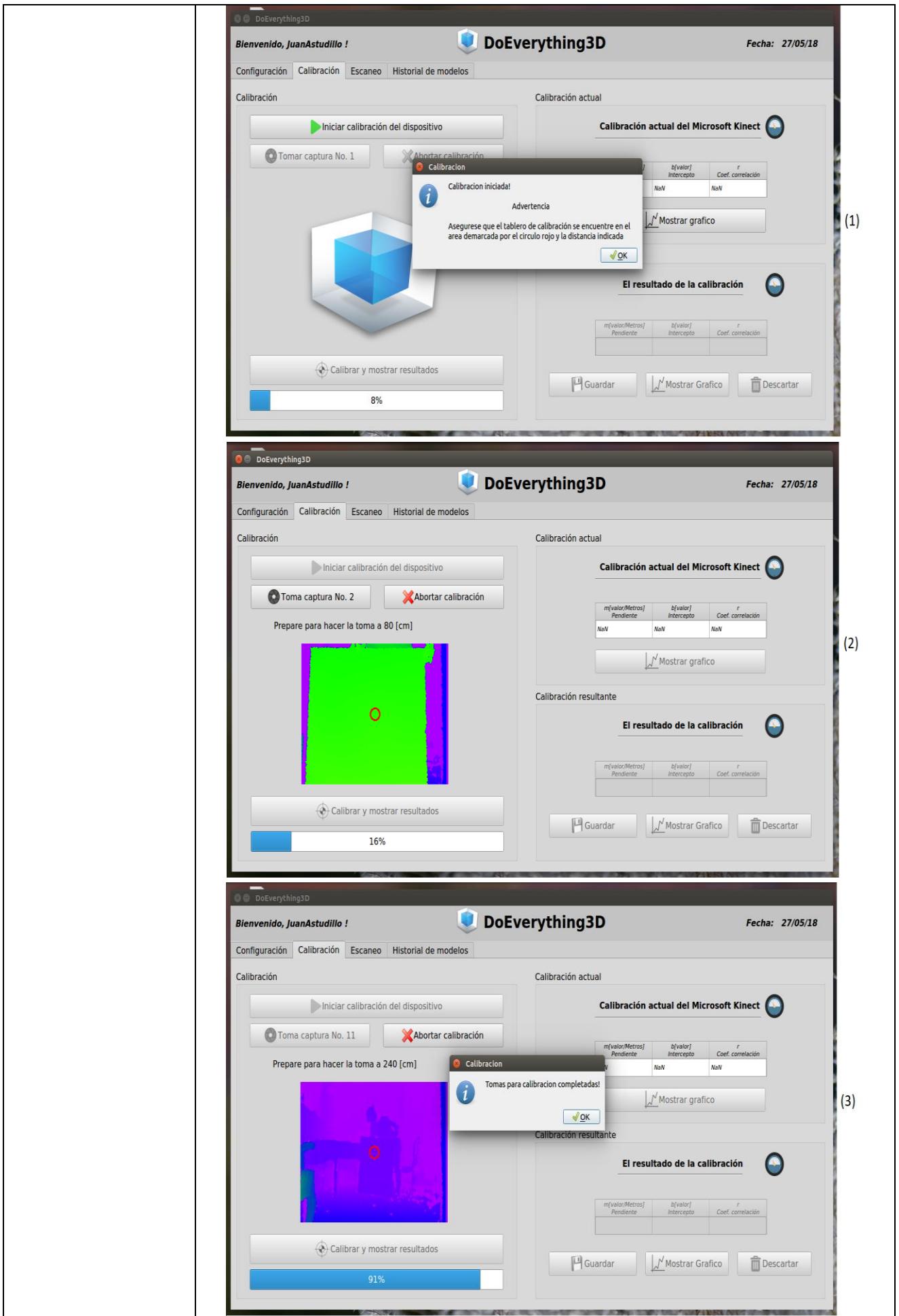
<b>PRUEBA DE INTEGRACIÓN – CALIBRACIÓN DEL SENSOR KINECT, ESCANEO 3D DE UN OBJETO Y VISUALIZACIÓN DEL MODELO 3D RESULTANTE</b>	
<b>Requerimientos DoEverything3D</b>	<ul style="list-style-type: none"> <li>• Permitir calibrar la cámara de profundidad del Microsoft Kinect.</li> </ul>

- Permitir adquirir de profundidad de un objeto mediante el Microsoft Kinect.
- Permitir controlar el giro del objeto a modelar.
- Permitir la conversión de las imágenes de profundidad capturadas a nubes de puntos 3D.
- Permitir segmentar las nubes de puntos 3D a partir de la detección del plano tierra.
- Permitir ejecutar el registro de nubes de puntos 3D segmentadas.
- Permitir ejecutar una reducción de ruido de la nube de puntos 3D registrada.
- Permitir generar una superficie sobre la nube de puntos 3D con ruido reducido.
- Permitir visualizar el modelo 3D del objeto terminado.

#### **Requerimientos 360PA**

- Permitir establecer una comunicación con el aplicativo DoEverything3D mediante USB.
- Permitir recibir órdenes de rotación desde el aplicativo DoEverything3D mediante el puerto USB.
- Permitir realizar rotaciones controladas de un objeto sobre sus 360 grados y visualizar el estado de la magnitud angular de la rotación.

<b>Tipo de Prueba</b>	Integración.
<b>Hardware Requerido</b>	Computador portátil, Sistema 360PA, Microsoft Kinect.
<b>Software Requerido</b>	DoEverything3D (Python 3.5.2, mysqlConnector, PyQt, numpy, PCL, Open3D, Pymesh, pySerial y libFreenect)
<b>Objetivo del requerimiento</b>	Probar que el aplicativo permita realizar la calibración del sensor Microsoft Kinect y almacenar su resultado de manera asociada al usuario. Para luego verificar si dicha calibración conjunta a los parámetros de operación y al sistema 360PA permiten generar un modelo 3D de un objeto de interés.
<b>Calibración del sensor Microsoft Kinect</b>	
<b>Objetivo de la prueba</b>	Verificar que el aplicativo permite la ejecución de la calibración del sensor Microsoft Kinect y también verificar si permite guardar el resultado de dicha calibración de manera asociada al usuario.
<b>Datos de la Prueba</b>	Tomas de profundidad hacia un plano.
<b>Procedimiento</b>	<ol style="list-style-type: none"> <li>1. Dirigirse a la pestaña de "Calibración" en la interfaz gráfica principal del aplicativo.</li> <li>2. Presionar el botón "Iniciar calibración del dispositivo".</li> <li>3. Realizar diez capturas mediante el botón "Tomar captura No. x" (El sensor Kinect debe estar capturando un plano en la mitad de su imagen de profundidad, dicho plano debe estar a una distancia del sensor descrita en la interfaz).</li> <li>4. Presionar el botón "Calibrar y mostrar resultados".</li> <li>5. Presionar el botón "guardar" ubicada en la sección de calibración resultante.</li> </ol>
<b>Resultado Esperado</b>	<ol style="list-style-type: none"> <li>1. Mensaje indicando que se ha iniciado la calibración</li> <li>2. Visualización de la imagen de profundidad tomada y de la distancia a la que se debe separar el sensor Microsoft Kinect del plano (para las 10 tomas) en la pestaña de calibración de la interfaz gráfica principal.</li> <li>3. Mensaje indicando que se ha realizado correctamente la toma de las 10 imágenes.</li> <li>4. Valores resultantes de la calibración desplegados en la tabla de calibración resultante.</li> <li>5. Mensaje informando sobre el correcto almacenamiento de la calibración.</li> <li>6. Actualización de los valores de la tabla de calibración actual.</li> </ol>
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia de resultados obtenidos en las imágenes:</p> <ol style="list-style-type: none"> <li>1. Mensaje indicando que se ha iniciado la calibración con algunas instrucciones de como posicionar el plano enfrente del sensor Microsoft Kinect.</li> <li>2. Visualización de la imagen de profundidad tomada y de la distancia a la que se debe separar el sensor Microsoft Kinect del plano (para las 10 tomas) en la pestaña de calibración de la interfaz gráfica principal.</li> <li>3. Mensaje indicando que se ha realizado correctamente la toma de las 10 imágenes.</li> <li>4. Valores resultantes de la calibración desplegados en la tabla de calibración resultante.</li> <li>5. Actualización de los valores de la tabla de calibración actual por los valores guardados anteriormente.</li> </ol>



**(4)**

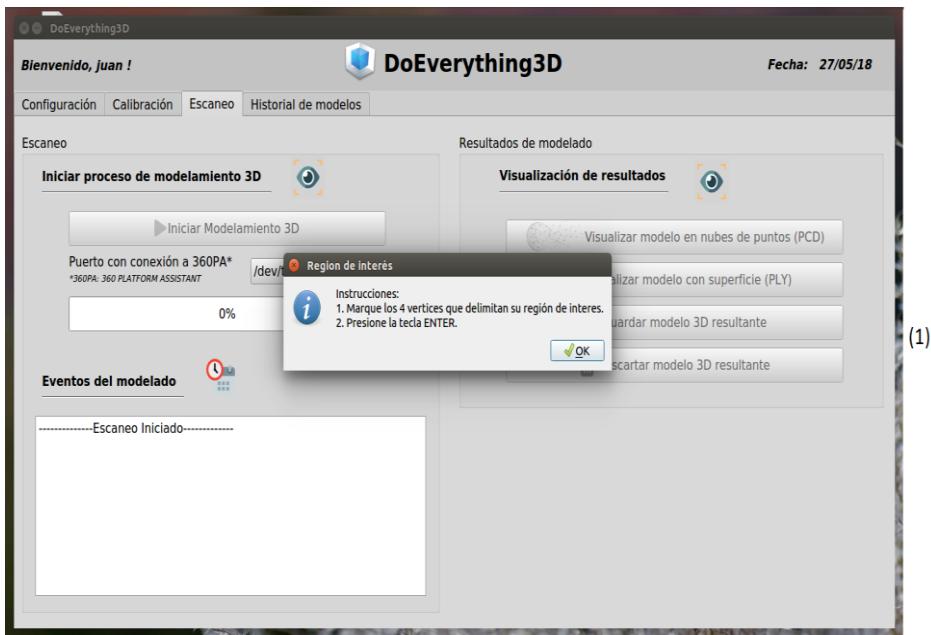
m[valor/Metros] Pendiente	b[valor] Intercepto	Coef. correlación
NaN	NaN	NaN

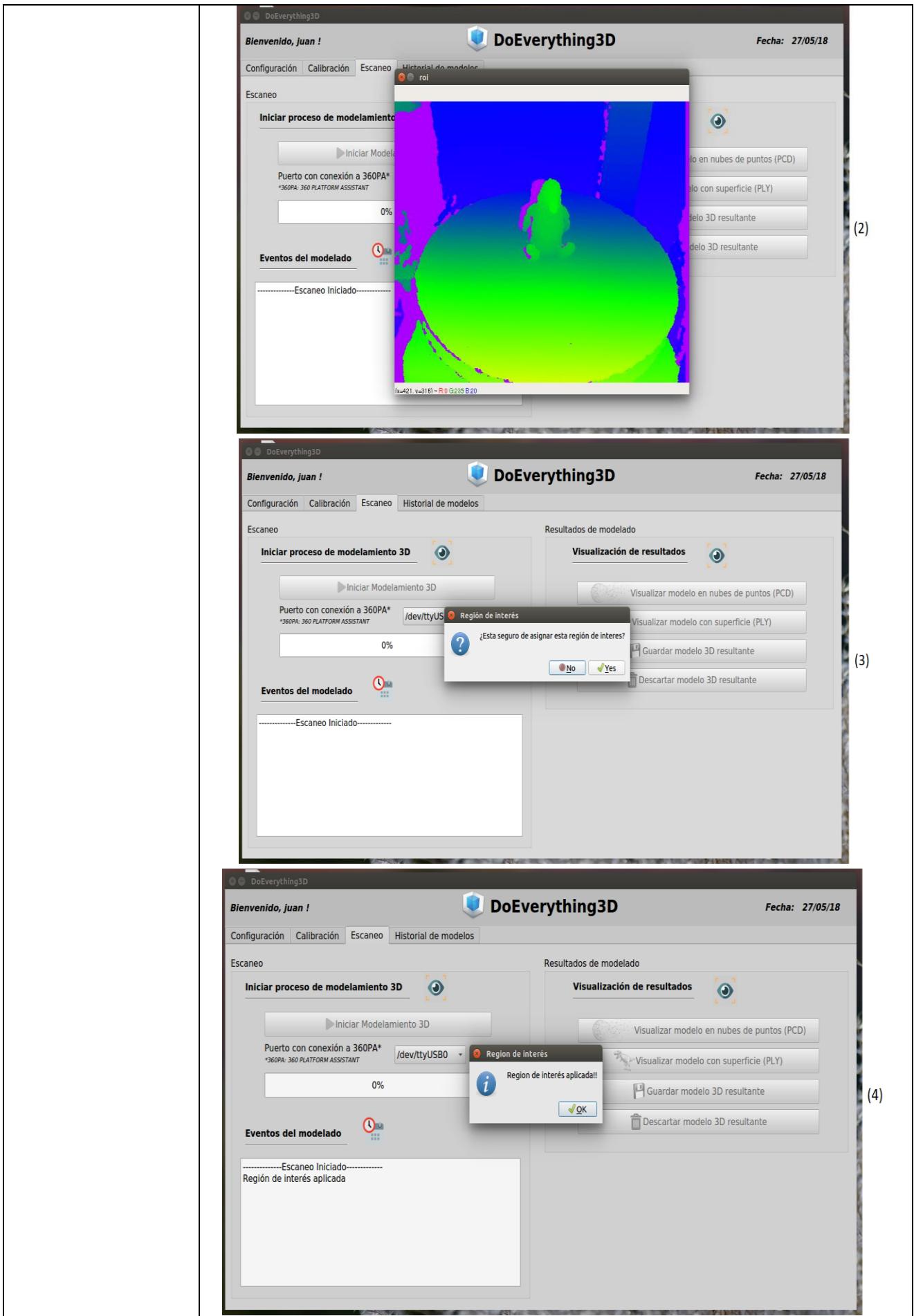
**(5)**

**(6)**

m[valor/Metros] Pendiente	b[valor] Intercepto	Coef. correlación
46.95	136.63	0.94

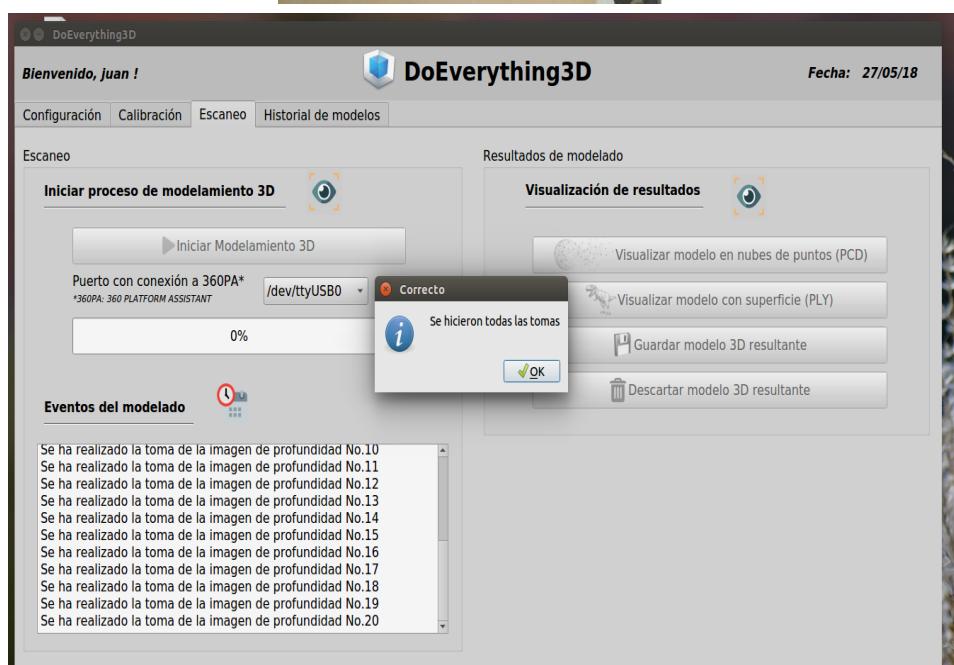
<b>Comentarios</b>	Se debe posicionar el plano a la distancias descritas en la interfaz gráfica para que la calibración sea válida para el aplicativo.
<b>Escaneo 3D de un objeto y visualización del modelo 3D resultante</b>	
<b>Objetivo de la prueba</b>	Verificar si el aplicativo con la calibración del sensor Microsoft Kinect conjunto a los parámetros de operación y al sistema 360PA permiten generar un modelo 3D de un objeto de interés.
<b>Datos de la Prueba</b>	-
<b>Procedimiento</b>	<ol style="list-style-type: none"> <li>1. Dirigirse a la pestaña de "Escaneo" en la interfaz gráfica principal del aplicativo.</li> <li>2. Seleccionar el puerto en donde se encuentra conectado el sistema 360PA en el campo "Puerto con conexión a 360PA".</li> <li>3. Presionar el botón "Iniciar Modelamiento 3D".</li> <li>4. Delimitar la región de interés (ROI) mediante 4 vértices.</li> <li>5. Confirmar que se está de acuerdo con la ROI definida.</li> <li>6. Presionar el botón "Visualizar modelo en nubes de puntos (PCD)"</li> <li>7. Presionar el botón "Visualizar modelo con superficie (PLY)"</li> </ol>
<b>Resultado Esperado</b>	<ol style="list-style-type: none"> <li>1. Mensaje indicando como delimitar la región de interés (ROI).</li> <li>2. Ventana desplegada con la función de limitar la ROI.</li> <li>3. Mensaje preguntando si se está de acuerdo con la ROI definida.</li> <li>4. Objeto rotando y Sensor Kinect realizándole tomas sobre sus 360°.</li> <li>5. Mensaje indicando que la etapa de captura a finalizado.</li> <li>6. Barra de progreso de escaneo en 100%.</li> <li>7. Despliegue de ventana en donde se visualice la nube de puntos del modelo 3D.</li> <li>8. Despliegue de ventana en donde se visualice la superficie del modelo 3D.</li> </ol>
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia de resultados obtenidos en las imágenes:</p> <ol style="list-style-type: none"> <li>1. Mensaje indicando el procedimiento para delimitar la ROI.</li> <li>2. Ventana en donde se delimitará la ROI.</li> <li>3. Mensaje preguntando si se está de acuerdo con la ROI definida.</li> <li>4. Mensaje indicando que se ha fijado la ROI.</li> <li>5. Sistema 360PA rotando el objeto a modelar y sensor Kinect realizando tomas a este.</li> <li>6. Mensaje indicando que se han concluido todas las tomas.</li> <li>7. Barra de progreso de escaneo en 100% indicando que ha finalizado el modelamiento 3D.</li> <li>8. Ventana visualizando el modelo 3D resultante en formato de nubes de puntos.</li> <li>9. Ventana visualizando el modelo 3D resultante en formato de superficie.</li> </ol>



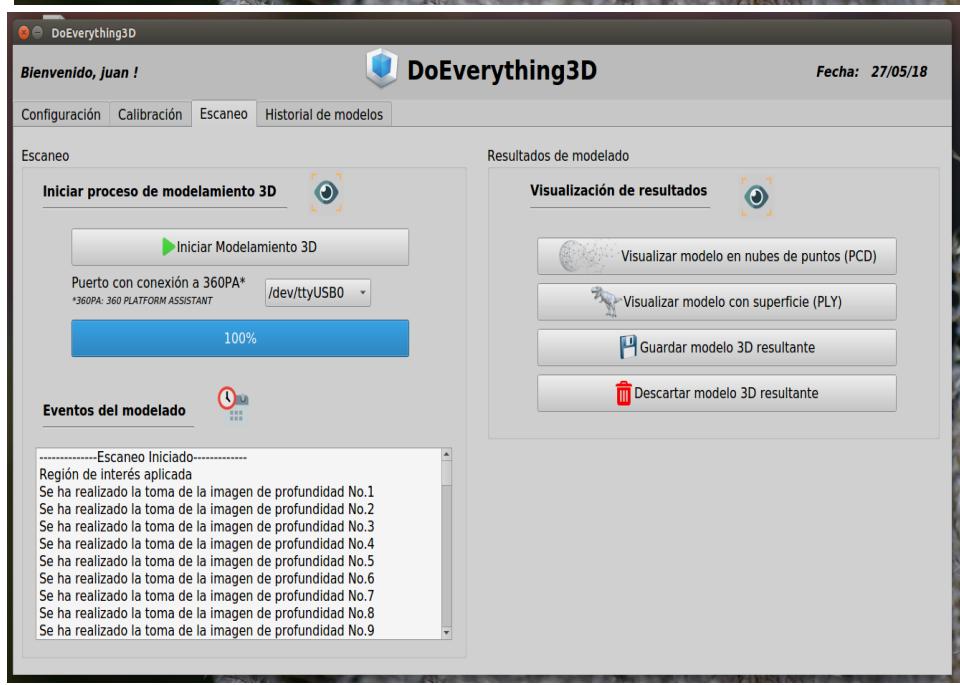




(5)



(6)



(7)

<b>Comentarios</b>	Se debe seleccionar una región de interés correcta para un buen modelamiento 3D del objeto de interés. No se debe obstruir el sensor Microsoft Kinect mientras este se encuentra adquiriendo imágenes.	

En la tabla 5.2 se puede visualizar las capturas de pantallas respectivas al procedimiento realizado tanto como para la ejecución y almacenamiento del resultado de la calibración del sensor Microsoft Kinect como para la generación de modelos 3D. En el apartado de calibración, es expuesta la captura (1) en donde se evidencia en un mensaje arrojado por el aplicativo que informa el inicio de la calibración y además al mensaje se añaden indicaciones de como posicionar el plano enfrente del sensor Microsoft Kinect. Luego el usuario realiza las 10 tomas al plano a diferentes distancias del sensor Microsoft Kinect, dichas tomas son visualizadas como se observa en (2) y además se aprecia que se indica la próxima distancia a la que debe ir posicionado el plano del sensor. Luego, en (3) se puede apreciar el mensaje desplegado por el aplicativo que indica que se han realizado las 10 tomas exitosas. En (4) es expuesta la fijación y actualización de los valores resultantes de la calibración en la tabla de calibración resultante. Al momento de guardar la nueva calibración, la tabla de calibración actualiza sus valores por los nuevos guardados por el usuario como se ve en (5).

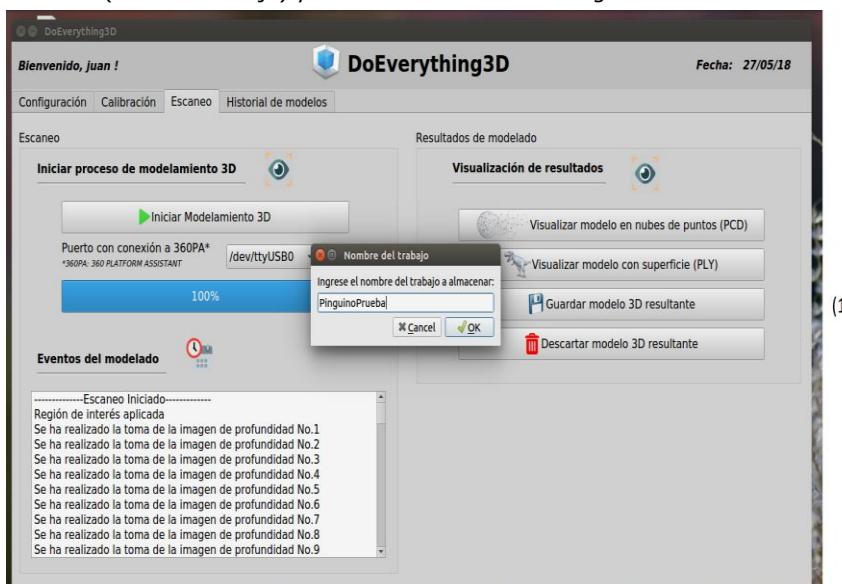
En el apartado de generación de modelos 3D, se expone en (1) un mensaje desplegado por el aplicativo que indica el procedimiento para delimitar la región de interés (ROI). Luego, en (2) se visualiza la ventana arrojada por el aplicativo la cual está destinada a limitar la ROI. En (3) se aprecia un mensaje de confirmación de la ROI definida y si el usuario está de acuerdo con dicha ROI se despliega el mensaje (4), indicando que se ha definido satisfactoriamente la ROI. Luego en (5) se aprecia la disposición del objeto sobre el sistema 360PA, el cual es rota sobre sus 360° mientras el sensor Microsoft Kinect toma imágenes de profundidad sobre dicho objeto. En (6) se aprecia el mensaje desplegado por el aplicativo que informa que el procedimiento de tomas ha concluido. En (7) se aprecia la interfaz gráfica del aplicativo DoEverything3D luego de finalizar el proceso de escaneo o modelamiento. Si el usuario desea ver el modelo 3D generado en formato de nube de puntos se despliega una ventana como en (8), y si lo desea ver en formato de superficie se despliega una ventana como en (9). Estos procedimientos de calibración y escaneo evidencian el correcto cumplimiento de los requerimientos planteados en dicha prueba.

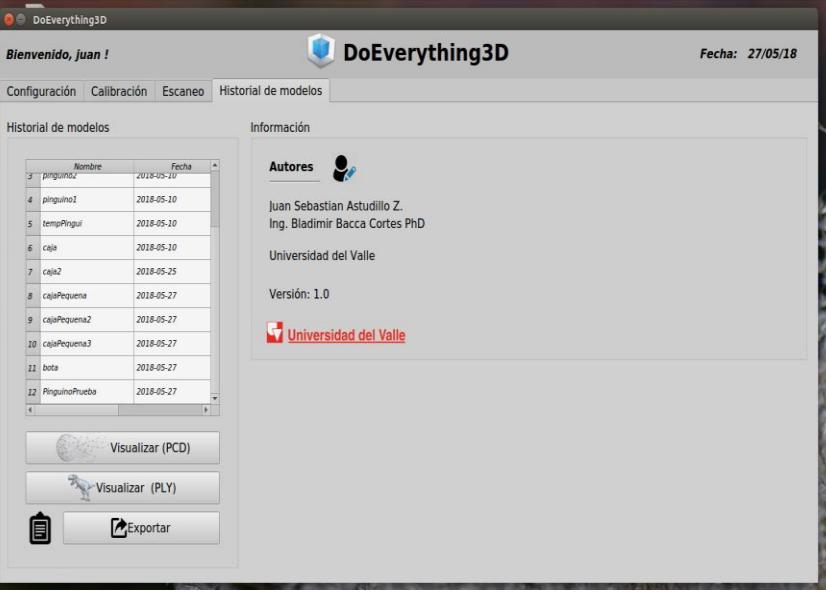
### 5.2.3. Almacenamiento de un modelo 3D, visualización y exportado de un modelo 3D del historial de trabajos.

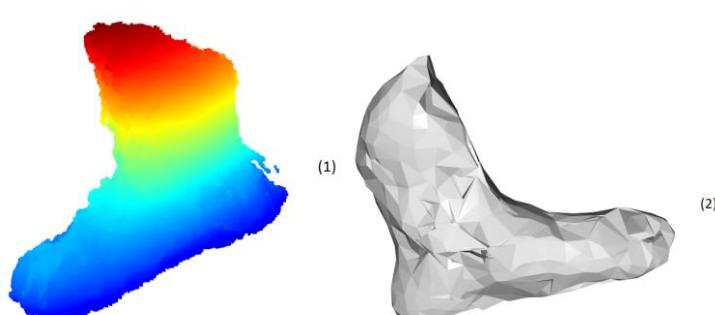
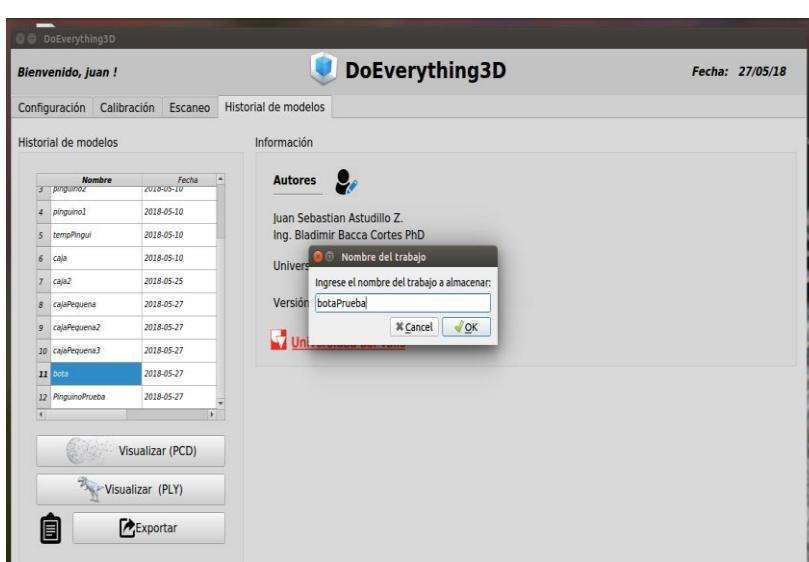
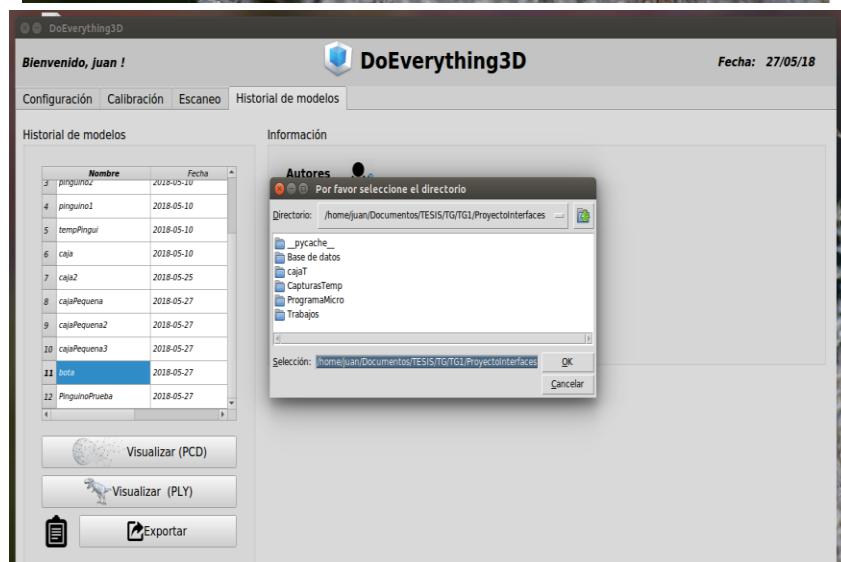
La prueba de integración correspondiente a almacenamiento de un modelo 3D, visualización y exportado de un modelo 3D del historial de trabajos, está destinada a corroborar que el aplicativo permita el almacenamiento de modelos 3D de manera asociada al usuario y también que permita la visualización y la exportación de un modelo 3D del historial de trabajos de un usuario. Esta prueba junto con los resultados se evidencia en la tabla 5.3.

**Tabla 5.3.** Prueba de integración: Almacenamiento de un modelo 3D, visualización y exportado de un modelo 3D del historial de trabajos.

<b>PRUEBA DE INTEGRACIÓN – ALMACENAMIENTO DE UN MODELO 3D, VISUALIZACIÓN Y EXPORTADO DE UN MODELO 3D DEL HISTORIAL DE TRABAJOS</b>	
<b>Requerimientos</b>	<ul style="list-style-type: none"> <li>• Permitir almacenar en una base de datos los modelos 3D resultantes de cada usuario.</li> </ul>

	<ul style="list-style-type: none"> <li>• Permitir visualizar el historial de modelos almacenados del usuario en la base de datos.</li> <li>• Permitir visualizar un modelo 3D almacenado en el historial de trabajos.</li> <li>• Permitir exportar un modelo almacenado en la base de datos del usuario bajo un nombre y en una ruta específica.</li> </ul>
<b>Tipo de Prueba</b>	Integración.
<b>Hardware Requerido</b>	Computador portátil.
<b>Software Requerido</b>	DoEverything3D (Python 3.5.2, mysqlConnector, PyQt, Open3D) y XAMPP 7.2.5.
<b>Objetivo del requerimiento</b>	Verificar que el aplicativo permite el almacenamiento de modelos 3D de manera asociada al usuario y también que permita la visualización y la exportación de un modelo 3D del historial de trabajos de un usuario.
<b>Almacenamiento de un modelo 3D</b>	
<b>Objetivo de la prueba</b>	Corroborar que el aplicativo permite el almacenamiento de modelos 3D de manera asociada al usuario.
<b>Datos de la Prueba</b>	Nombre trabajo, Modelo 3D.
<b>Procedimiento</b>	<ol style="list-style-type: none"> <li>1. Oprimir el botón "Guardar modelo 3D resultante".</li> <li>2. Ingresar el nombre (nombre trabajo) con el que se guardará el modelo 3D bajo sus dos formatos (PCD y PLY).</li> </ol>
<b>Resultado Esperado</b>	<ol style="list-style-type: none"> <li>1. Despliegue de ventana para digitar el nombre con el que se guardará el modelo 3D generado.</li> <li>2. Mensaje indicando que el modelo 3D se ha almacenado satisfactoriamente.</li> <li>3. Inserción del modelo en la tabla de historial de modelos(ubicada en la pestaña de "Historial de modelos" en la interfaz gráfica principal del aplicativo), bajo el nombre indicado (nombre trabajo) y en la fecha en la cual fue generado.</li> </ol>
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia de resultados obtenidos en las imágenes:</p> <ol style="list-style-type: none"> <li>1. Ventana desplegada para introducir el nombre con el que guardará el modelo 3D generado.</li> <li>2. Mensaje indicando que el modelo 3D se ha almacenado con éxito.</li> <li>3. Inserción del modelo en la tabla de historial de modelos (ubicada en la pestaña de "Historial de modelos" en la interfaz gráfica principal del aplicativo), bajo el nombre indicado (nombre trabajo) y en la fecha en la cual fue generado.</li> </ol> 

	 <p>(3)</p>
<b>Comentarios</b>	Se debe escoger un nombre de almacenamiento del modelo 3D que no esté presente en el historial de modelos asociado al usuario.
<b>Visualización del historial de trabajos y exportado de un modelo 3D del historial de trabajos</b>	
<b>Objetivo de la prueba</b>	Corroborar que el aplicativo permite la exportación de un modelo 3D del historial de trabajos de un usuario bajo sus dos formatos (PLY y PCD).
<b>Datos de la Prueba</b>	Nombre trabajo, ruta exportado.
<b>Procedimiento</b>	<ol style="list-style-type: none"> <li>1. Dirigirse a la pestaña de "Historial de modelos" en la interfaz gráfica principal del aplicativo.</li> <li>2. Seleccionar un modelo 3D en la tabla de historial de modelos.</li> <li>3. Oprimir el botón "Visualizar (PCD)".</li> <li>4. Oprimir el botón "Visualizar (PLY)".</li> <li>5. Oprimir el botón "Exportar".</li> <li>6. Ingresar el nombre (nombre trabajo) con el que se exportará el modelo 3D bajo sus dos formatos (PCD y PLY).</li> <li>7. Ingresar la ruta (ubicación en el ordenador) en donde se exportará el modelo 3D bajo sus dos formatos (PCD y PLY) y el nombre introducido (nombre trabajo).</li> </ol>
<b>Resultado Esperado</b>	<ol style="list-style-type: none"> <li>1. Despliegue de ventana en donde se visualice la nube de puntos del modelo 3D seleccionado.</li> <li>2. Despliegue de ventana en donde se visualice la superficie del modelo 3D seleccionado.</li> <li>3. Despliegue de ventana para digitar el nombre con el que se exportará el modelo 3D seleccionado.</li> </ol>

	<p>4. Despliegue de ventana para seleccionar la ruta en donde exportará el modelo 3D seleccionado.</p> <p>5. Mensaje comunicando que la exportación del modelo 3D ha concluido de manera satisfactoria.</p>
<b>Resultado Obtenido</b>	<p><b>Prueba satisfactoria.</b> A continuación, se detallan la secuencia de resultados obtenidos en las imágenes:</p> <ol style="list-style-type: none"> <li>1. Ventana visualizando el modelo 3D seleccionado en formato de nubes de puntos.</li> <li>2. Ventana visualizando el modelo 3D seleccionado en formato de superficie.</li> <li>3. Ventana para digitar el nombre con el que se exportará el modelo 3D seleccionado.</li> <li>4. Ventana para seleccionar la ruta en donde se exportará el modelo 3D seleccionado.</li> <li>5. Mensaje comunicando que la exportación del modelo 3D ha concluido de manera satisfactoria.</li> </ol>  <div style="display: flex; justify-content: space-around;"> <span>(1)</span> <span>(2)</span> </div> <div style="margin-top: 20px;">  <span>(3)</span> </div> <div style="margin-top: 20px;">  <span>(4)</span> </div>

	
<b>Comentarios</b>	Se debe elegir una ruta de exportación en la cual el usuario tenga permisos y un nombre de trabajo no nulo.

En la tabla 5.3 se puede visualizar las capturas de pantallas respectivas al procedimiento realizado tanto a almacenamiento de un modelo 3D como a la visualización y exportado de un modelo 3D del historial de trabajos. En el apartado de almacenamiento de modelos 3D, se expone en (1) la ventana desplegada por el aplicativo en donde se debe ingresar el nombre con el que se guardará el modelo 3D generado. En (2) se visualiza el mensaje arrojado por el aplicativo indicando que el modelo 3D se ha almacenado satisfactoriamente. Por último, en (3) se visualiza la inserción del modelo en la tabla de historial de modelos (ubicada en la pestaña de “Historial de modelos” en la interfaz gráfica principal del aplicativo), bajo el nombre indicado (nombre trabajo) y en la fecha en la cual fue generado.

En el apartado de visualización y exportado de un modelo 3D del historial de trabajos, se expone en (1) la ventana desplegada por el aplicativo visualizando el modelo 3D seleccionado del historial de trabajos en formato de nubes de puntos. En (2) se aprecia la ventana arrojada por el aplicativo visualizando el modelo 3D seleccionado del historial de trabajos en formato de superficie. Luego en (3) se expone la ventana desplegada por el aplicativo para que el usuario digite el nombre con el que se exportará el modelo 3D seleccionado y (4) la ventana para seleccionar la ruta en donde se exportará el modelo 3D seleccionado. Por último, en (5) se evidencia un mensaje comunicando que la exportación del modelo 3D ha concluido de manera satisfactoria.

### 5.3. PRUEBAS CUANTITATIVAS

Ahora, luego de concluir las pruebas de integración, trataremos unas pruebas que evaluarán la calidad del modelo 3D generado y luego mostraremos una muestra de diferentes objetos escaneados y su resultado.

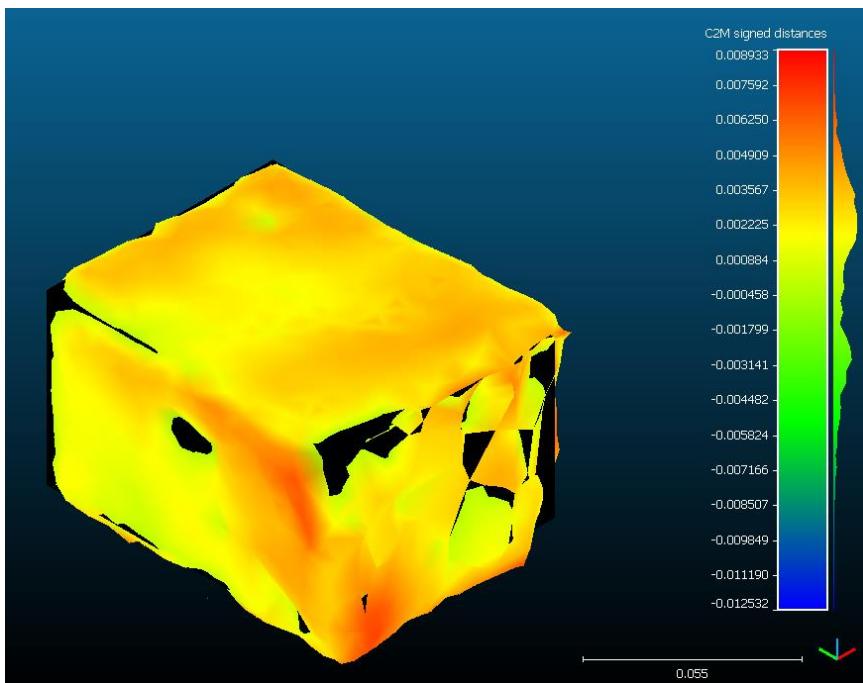
#### 5.3.1. Análisis de Error de Modelado

Con el objetivo de encontrar el error en los modelos 3D que genera DoEverything3D es necesario un sistema de referencia de alta prestaciones que genere el mismo modelo 3D a analizar. Debido a la carencia de dicho sistema de referencia, se ha propuesto escanear objetos a los que se les conoce su modelo 3D real. Para realizar las pruebas que nos indicarán el error presente en los modelos extraídos con DoEverything3D trabajaremos con modelos 3D de cajas. Dichos modelos 3D de cajas pueden ser fácilmente modelables para tomar referencia frente al modelo 3D generado por DoEverything3D.

El error entre el modelo generado por DoEverything3D y el modelo ideal será calculado mediante la distancia Hausdorff [60]. La distancia de Hausdorff entre dos nubes de puntos consiste en calcular para cada punto  $p$  de una nube  $S$  la distancia a su punto más cercano en la otra nube  $S'$  denominado  $p'$ , como lo muestra la ecuación 5.1.

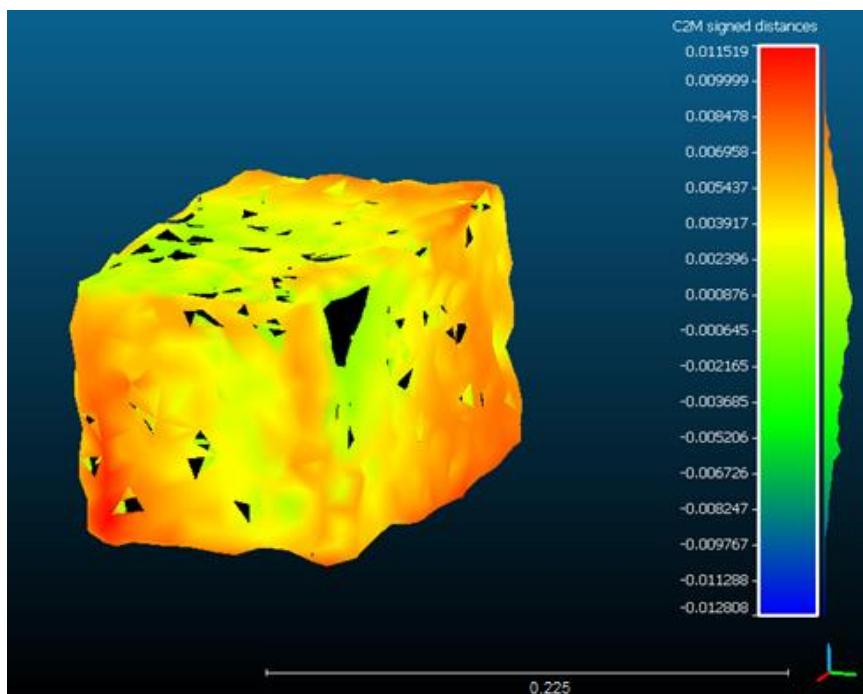
$$d(p, S') = \min_{p' \in S'} \|p - p'\|^2 \quad (5.1)$$

En la figura 5.1 se aprecia el modelo 3D generado por DoEverything3D superpuesto sobre su modelo ideal (Negro). El pseudocolor aplicado al modelo generado por DoEverything3D viene dado en función al error presente respecto al modelo ideal. El Valor del error está dado en metros y el histograma del error se muestra en la columna derecha. Cabe recalcar que la caja modelada presenta unas dimensiones de 10 cm de largo, 7 cm de ancho y 6 cm de alto.



**Figura 5.1.** Modelo 3D generado por el sistema DoEverything3D de una caja de dimensiones 10 cm de largo, 7 cm de ancho y 6 cm de alto superpuesto sobre su modelo ideal (Negro). El pseudocolor aplicado al modelo generado por el sistema DoEverything3D viene dado en función al error presente respecto al modelo ideal (Valor del error en metros e histograma del error en la columna derecha).

Se puede visualizar que el modelo presenta un error medio de -1,799 mm y una media de dispersión o desviación estándar de esta medida (en este trabajo se usa  $5\sigma$ ) de 10,733 mm. En la figura 5.2, se aprecia el error calculado en un segundo modelo 3D generado por DoEverything3D de una caja de dimensiones mayores (17 cm de largo, 13 cm de ancho y 11 cm de alto) el cual se encuentra superpuesto por su modelo 3D ideal (Negro). El error en este segundo modelo 3D generado por DoEverything3D tiene como media -0,645 mm y desviación estándar (en este trabajo se usa  $5\sigma$ ) de 12,164 mm.



**Figura 5.2.** Modelo 3D generado por el sistema DoEverything3D de una caja de dimensiones 17 cm de largo, 13 cm de ancho y 11 cm de alto superpuesto sobre su modelo ideal (Negro). El pseudocolor aplicado al modelo generado por el sistema DoEverything3D viene dado en función al error presente respecto al modelo ideal (Valor del error en metros e histograma del error en la columna derecha).

Con el objetivo de obtener un promedio de error en los modelos 3D generados por el sistema DoEverything3D, se procede a obtener otros diez modelos 3D de cajas de diferentes dimensiones. Esto con

el fin de emplear la mayor parte del rango disponible en la plataforma del sistema 360PA. A dichos modelos se les ha calculado su error como se ha realizado mediante la distancia Hausdorff entre su modelo 3D ideal (Ver Figura 5.1, y 5.2). De esa manera se tienen 12 modelos con su medida media de error respectiva y la desviación estándar media de error (en este trabajo se usará  $5\sigma$ ). Posteriormente, se calcula el promedio de dichas medias de error respectiva y desviaciones estándar (en este trabajo se usará  $5\sigma$ ). Los resultados de los errores y sus promedios con distintas cajas se han condensado en la tabla 5.4 y en la carpeta Anexo\_PruebasErrorDoEverything3D se encontrará la evidencia de dichas pruebas.

**Tabla 5.4.** Rangos de error presentes en diferentes modelos 3D de cajas generados por DoEverything3D.

NUMERO EXPERIMENTO	DIMENSIONES REALES DEL OBJETO			ERROR MEDIO EN MODELO GENERADO POR DOEVERYTHING3D [mm]	DESVIACIÓN ESTÁNDAR EN EL ERROR ( $5\sigma$ ) [mm]
	LARGO [cm]	ANCHO [cm]	ALTO [cm]		
1	$10.0 \pm 0.1$	$7.0 \pm 0.1$	$6.0 \pm 0.1$	$-1,800 \pm 1.000$	$10,733 \pm 1.000$
2	$17.0 \pm 0.1$	$13.0 \pm 0.1$	$11.0 \pm 0.1$	$-0,645 \pm 1.000$	$12,164 \pm 1.000$
3	$8.0 \pm 0.1$	$6.0 \pm 0.1$	$10.0 \pm 0.1$	$1,489 \pm 1.000$	$9,141 \pm 1.000$
4	$8.0 \pm 0.1$	$10.0 \pm 0.1$	$2.5 \pm 0.1$	$0,907 \pm 1.000$	$7,823 \pm 1.000$
5	$24.0 \pm 0.1$	$16.0 \pm 0.1$	$20.0 \pm 0.1$	$-1,571 \pm 1.000$	$12,845 \pm 1.000$
6	$10.0 \pm 0.1$	$6.0 \pm 0.1$	$8.0 \pm 0.1$	$-1,155 \pm 1.000$	$10,504 \pm 1.000$
7	$4.0 \pm 0.1$	$3.0 \pm 0.1$	$8.0 \pm 0.1$	$1,597 \pm 1.000$	$8,072 \pm 1.000$
8	$5.0 \pm 0.1$	$4.0 \pm 0.1$	$5.0 \pm 0.1$	$0,815 \pm 1.000$	$2,999 \pm 1.000$
9	$6.0 \pm 0.1$	$4.0 \pm 0.1$	$3.0 \pm 0.1$	$1,362 \pm 1.000$	$3,564 \pm 1.000$
10	$16 \pm 0.1$	$24.0 \pm 0.1$	$7.0 \pm 0.1$	$4,417 \pm 1.000$	$6,578 \pm 1.000$
11	$25.0 \pm 0.1$	$25.0 \pm 0.1$	$25.0 \pm 0.1$	$2,144 \pm 1.000$	$13,402 \pm 1.000$
12	$15 \pm 0.1$	$11 \pm 0.1$	$9 \pm 0.1$	$-2,703 \pm 1.000$	$8,449 \pm 1.000$
<b>PROMEDIO</b>				<b><math>0.404 \pm 1.000</math></b>	<b><math>8.855 \pm 1.000</math></b>

Posteriormente, con los valores medios del error presente en los modelos generados por DoEverything3D se realiza una primera prueba t de Student [61] con un nivel de confianza del 99.9%. Esta prueba t de Student se realiza con la finalidad de evaluar las siguientes dos hipótesis:

1. **Hipótesis nula:** DoEverything3D genera modelos 3D de objetos con una media de error menor o igual a 0.3988 mm.
2. **Hipótesis 1:** DoEverything3D genera modelos 3D de objetos con una media de error mayor a 0.3988 mm

Para lo cual se tiene una distribución t de dos colas con 11 grados de libertad y con zona de no rechazo de la hipótesis entre  $-3.106 < t < 3.106$ .

Realizando la primera prueba t de Student con la hipótesis 1 se obtiene un valor de distribución de 3.112, valor que se encuentra fuera de la zona de no rechazo. Mientras, si se comprueba la hipótesis nula se obtiene un valor de distribución de 3.103, el cual se encuentra en la zona de no rechazo.

Luego, con los valores de desviación estándar del error ( $5\sigma$ ) presente en los modelos generados por DoEverything3D se realiza una segunda prueba t de Student [61] con un nivel de confianza igual a la prueba anterior de 99.9%. Esta prueba t de Student se realiza con la finalidad de evaluar las siguientes dos hipótesis:

1. **Hipótesis nula:** DoEverything3D genera modelos 3D de objetos con una desviación estándar del error ( $5\sigma$ ) sobre la media de error menor o igual a 8.846 mm.
2. **Hipótesis 1:** DoEverything3D genera modelos 3D de objetos con una desviación estándar del error ( $5\sigma$ ) sobre la media de error mayor a 8.846 mm.

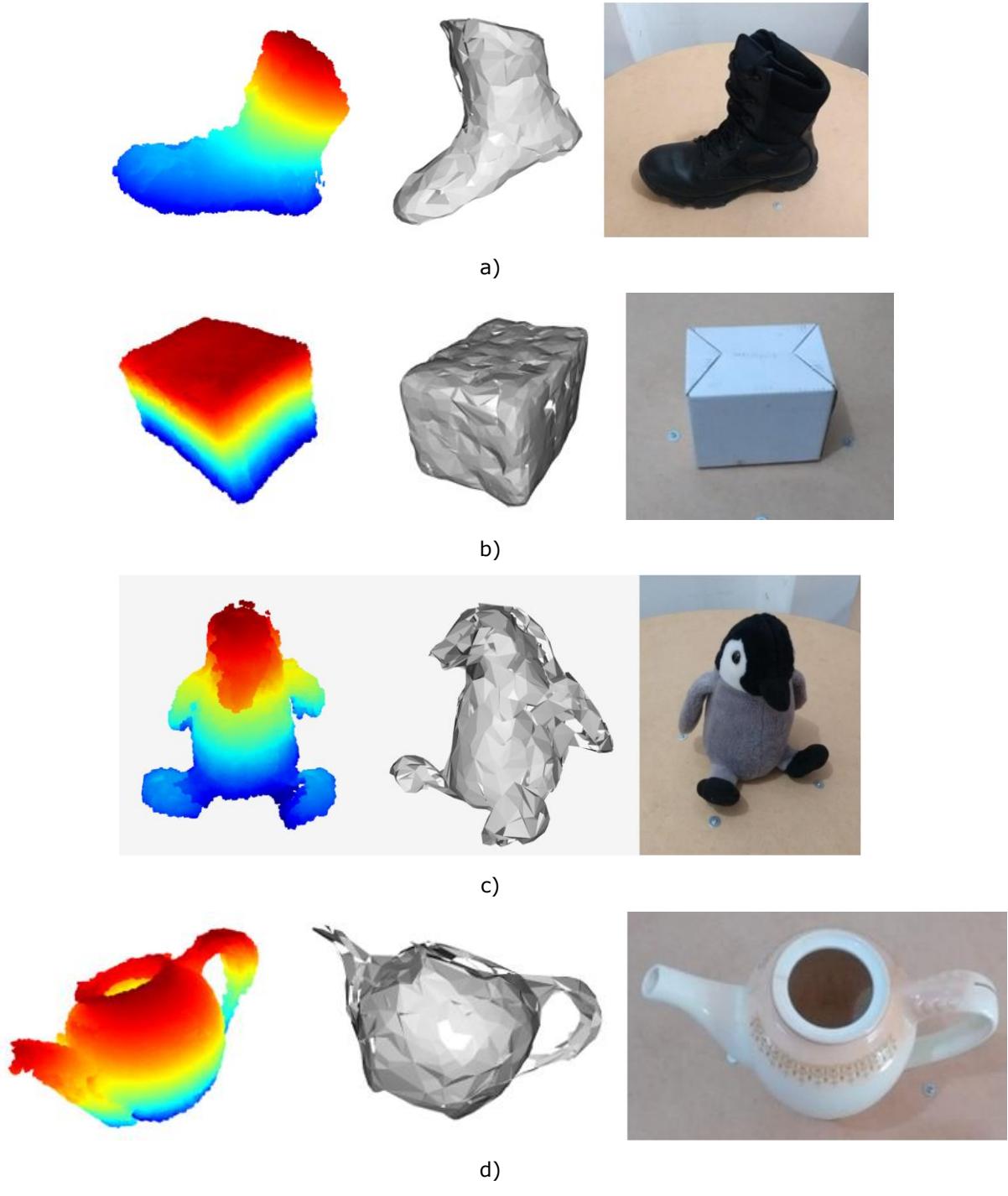
De la misma manera a la prueba 1, se tiene una distribución t de dos colas con 11 grados de libertad y con zona de no rechazo de la hipótesis entre  $-3.106 < t < 3.106$ .

Realizando la segunda prueba t de Student con la hipótesis 1 se obtiene un valor de distribución de 3.150, valor que se encuentra fuera de la zona de no rechazo. Mientras, si se comprueba la hipótesis nula se obtiene un valor de distribución de 3.091, el cual se encuentra en la zona de no rechazo.

Por lo que se concluye que los modelos generados por DoEverything3D presentan un error medio de 0.3988 mm con una desviación estándar ( $5\sigma$ ) de 8.846 mm.

### 5.3.2. Extracción de Modelos 3D de diferentes Objetos

En este apartado presentaremos una muestra de diferentes modelos 3D extraídos mediante el sistema DoEverything3D. Dichos modelos serán expuestos en la figura 5.3, en donde se presentarán: su formato en nubes de puntos, su formato en superficie y como son en realidad.



**Figura 5.3.** Muestra de diferentes modelos 3D extraídos mediante el sistema DoEverything3D.

- a) Modelo 3D generado por el sistema DoEverything3D de una bota.
- b) Modelo 3D generado por el sistema DoEverything3D de una caja.
- c) Modelo 3D generado por el sistema DoEverything3D de un juguete en forma de pingüino.

d) Modelo 3D generado por el sistema DoEverything3D de una tetera.

Ahora, discutiremos sobre algunos objetos a los cuales el sistema no logra generar modelos 3D. Dichos objetos son los que presentan las siguientes características:

- ✓ *Objetos de vidrio o superficies reflectantes*: debido a la tecnología en la que está basado el sensor Kinect (Luz estructurada) no es posible obtener información de profundidad sobre superficies de vidrio o reflectantes, ya que la grilla infrarroja que se proyecta no es reflejada desapareciendo de la escena captura por el sensor. Por lo tanto, el aplicativo no es capaz de retornar un modelo 3D de dichos objetos.
- ✓ *Objetos simétricos*: Si la información de profundidad es idéntica sobre todas las vistas capturadas al objeto a modelar, la etapa de registro alineará todas las nubes de puntos de manera sobrepuerta como se aprecia en la figura 5.4, ya que para dicha etapa el objeto no ha sido rotado.



**Figura 5.4.** Modelo 3D generado por el sistema DoEverything3D de objeto simétrico (rollo de papel).

## 5.4. CONCLUSIONES

A partir de las tablas de pruebas y resultados, que hacen parte de la metodología RUP, se observó la integración de cada uno de los requerimientos funcionales del proyecto, demostrando por medio de las capturas de pantalla la integración de cada una de las funciones que implican los requerimientos del sistema para el objetivo final del software DoEverything3D.

El análisis de operación del sistema DoEverything3D parte de una prueba de verificación del proceso de registro de usuarios, la cual resultó correcta al momento de realizarla con un usuario nuevo, añadiéndolo al modelo persistencia del sistema DoEverything3D. En la misma prueba, se corroboró que dicho usuario pudiera iniciar una sesión en el aplicativo y a la vez pudiera fijar y guardar sus propios parámetros de operación del aplicativo. Este proceso también resultó satisfactorio actualizando las configuraciones asociadas al usuario en el módulo persistencia. Dicha prueba confirma el cumplimiento de los siguientes requisitos funcionales asociados a los sub-módulos de usuarios y configuración:

- ✓ permitir crear un usuario,
- ✓ permitir acceder a una sesión de usuario y
- ✓ permitir seleccionar y almacenar en una base de datos la configuración de operación del aplicativo por usuario.

Seguidamente se corroboró mediante una prueba, que el usuario del aplicativo DoEverything3D pudiera realizar, visualizar y guardar los resultados de la calibración del sensor Microsoft Kinect, prueba la cual resultó satisfactoria debido al correcto procedimiento que indica el aplicativo de como posicionar el sensor respecto a un plano. Posteriormente en la misma prueba, se verificó que el usuario del aplicativo pudiera generar modelos 3D de objetos mediante el uso del sistema 360PA, la calibración del sensor Microsoft Kinect y los parámetros de funcionamiento del aplicativo. Lo anterior se prueba con resultado satisfactorios debido a una correcta selección de la región de interés (Región en donde se encuentra el objeto a modelar) y a que se evidenció un modelo 3D resultante acorde al objeto escaneado. Dicha prueba corrobora el cumplimiento de los siguientes requisitos funcionales asociados a los sub-módulos de calibración, escaneo y al sistema 360PA:

- ✓ permitir calibrar la cámara de profundidad del Microsoft Kinect,
- ✓ permitir adquirir de profundidad de un objeto mediante el Microsoft Kinect,
- ✓ permitir controlar el giro del objeto a modelar,
- ✓ permitir la conversión de las imágenes de profundidad capturadas a nubes de puntos 3D,
- ✓ permitir segmentar las nubes de puntos 3D a partir de la detección del plano tierra,

- ✓ permitir ejecutar el registro de nubes de puntos 3D segmentadas,
- ✓ permitir ejecutar una reducción de ruido de la nube de puntos 3D registrada,
- ✓ permitir generar una superficie sobre la nube de puntos 3D con ruido reducido,
- ✓ permitir visualizar el modelo 3D del objeto terminado,
- ✓ permitir establecer una comunicación con el aplicativo DoEverything3D mediante USB,
- ✓ permitir recibir órdenes de rotación desde el aplicativo DoEverything3D mediante el puerto USB y
- ✓ permitir realizar rotaciones controladas de un objeto sobre sus 360 grados y visualizar el estado de la magnitud angular de la rotación.

Posteriormente se verificó mediante otra prueba, que el usuario del aplicativo DoEverything3D pudiera almacenar un modelo 3D resultante del escaneo de un objeto, prueba que concluyó de manera satisfactoria debido a que se evidenció dicho modelo ingresaba en el historial de modelos asociado a dicho usuario. Luego, en la misma prueba se corroboró que el usuario pudiera visualizar y exportar modelos 3D que se encuentren en su historial de modelos, prueba que terminó de manera satisfactoria evidenciando la visualización del modelo seleccionado y los archivos exportados del modelo en la ruta y bajo el nombre ingresado. Dicha prueba corrobora el cumplimiento de los siguientes requisitos funcionales asociados a los sub-módulos de escaneo e historial de modelos:

- ✓ permitir almacenar en una base de datos los modelos 3D resultantes de cada usuario,
- ✓ permitir visualizar el historial de modelos almacenados del usuario en la base de datos,
- ✓ permitir visualizar un modelo 3D almacenado en el historial de trabajos y
- ✓ permitir exportar un modelo almacenado en la base de datos del usuario bajo un nombre y en una ruta específica.

Además, se realizaron pruebas para evaluar la calidad del modelo generado por DoEverything3D. Dichas pruebas bajo la ayuda del software de licencia gratuita *CloudCompare*, las cuales tenían como objetivo encontraban el error (distancia Hausdorff [60]) entre el modelo 3D de una caja generada por el sistema DoEverything3D, y el modelo ideal de la caja. Luego mediante dos pruebas t de Student, se evaluó el error medio y desviación estándar ( $5\sigma$ ) de doce modelos 3D de cajas de diferentes dimensiones generadas por DoEverything3D. Los resultados de las pruebas t de Student permiten concluir que los modelos 3D generados por DoEverything3D tienen como error medio de 0.3988 mm con una desviación estándar ( $5\sigma$ ) de 8.846 mm.

Por último, se presentó una muestra de diferentes objetos escaneados mediante el sistema DoEverything3D. Se plantea además una de las principales limitaciones del sistema referente a los objetos que no se pueden modelar, dicha limitación es enfocada hacia los objetos reflectantes y los objetos simétricos.

# CAPÍTULO 6

## 6. CONCLUSIONES Y TRABAJO FUTURO

---

### 6.1. Conclusiones

### 6.2. Trabajo futuro

---

#### 6.1. CONCLUSIONES

Este trabajo se ha dedicado al diseño e implementación de una herramienta software para la extracción automatizada de un modelo 3D de objetos usando el sensor Microsoft Kinect la cual se ha denominado DoEverything3D. DoEverything3D funciona a través de una interfaz gráfica de fácil uso, en donde el usuario tendrá a disposición principalmente la configuración de operación de la herramienta, la calibración del sensor Microsoft Kinect, la generación de modelos y administración sus modelos 3D almacenados.

DoEverything3D brinda, al campo de modelamiento 3D de objetos, una alternativa a los métodos tradicionales y soluciones presentes en el mercado para este fin como lo son: los escáneres 3D de tiempo de vuelo [27], los escáneres 3D de diferencia de fase [30], la fotogrametría [26], escáneres 3D estereoscópicos [17] y los escáneres 3D de detección de silueta [26]. Proporcionando las siguientes ventajas sobre dichos métodos: la accesibilidad económica, el fácil uso del sistema, la automatización del modelamiento y la decente precisión de los modelos resultantes.

El diseño de este trabajo inició principalmente a la recolección y análisis de antecedentes sobre la generación de modelos 3D. Dicho análisis dio como resultado la extracción de una metodología común presente en cada uno de los antecedentes analizados para la generación de los modelos 3D. La metodología extraída consiste principalmente en una serie de algoritmos ejecutados de manera secuencial, que tienen como finalidad pre procesar las imágenes de profundidad adquiridas sobre el objeto de interés, obteniendo nubes de puntos 3D únicamente del objeto, para luego proceder a realizar un registro global de dichas nubes y un filtrado del resultado. Obteniendo así el modelo 3D del objeto en formato de nube de puntos. Para efectos de visualización el algoritmo también genera una superficie sobre dicha nube de puntos 3D y de esa manera el modelo 3D del objeto resultará bajo dos formatos muy comunes entre los softwares de manejo de objetos 3D: nube de puntos (PCD) y superficie (PLY). Cabe aclarar que este procedimiento se implementó de manera transparente para el usuario y se realiza al momento de solicitar un modelo 3D en la interfaz de DoEverything3D.

Como ya se mencionó el diseño y desarrollo de DoEverything3D fue orientado para su fácil uso. De manera que el usuario observará una interfaz gráfica intuitiva en donde podrá realizar de manera sencilla todas las tareas correspondientes a la configuración de operación de la herramienta, la calibración del sensor Microsoft Kinect, la generación de modelos 3D a partir de imágenes de profundidad adquiridas mediante el sensor Microsoft Kinect, y administración de sus modelos 3D almacenados. Para ello, la interfaz gráfica se ha diseñado con la finalidad que todas estas tareas se realicen de la manera más simple posible para el usuario, y además con pocos parámetros de configuración para evitar que el usuario se sienta abrumado. Por otro lado, se diseñó e implementó un sistema de rotación de objetos que permite al aplicativo DoEverything3D en conjunto al sensor Microsoft Kinect realizar de manera automática la adquisición de imágenes de profundidad sobre el objeto a modelar. Ha dicho sistema de rotación se le ha denominado 360 Platform Assitant (360PA), una característica más que permite el fácil uso del sistema.

Cabe recalcar que toda la etapa de diseño, desarrollo y validación de la herramienta DoEverything3D (Aplicativo DoEverything3D y sistema 360PA) fue basada en la metodología RUP, mediante el uso de:

- ✓ Requerimientos funcionales.
- ✓ Requerimientos no funcionales.
- ✓ Casos de uso reales.
- ✓ Diagramas de secuencia.
- ✓ Diagrama de clases.
- ✓ Diagramas conceptuales.
- ✓ Pruebas de integración.

La documentación correspondiente a estos pasos se encuentra registrada en este documento, a excepción de los casos de uso real y diagramas de secuencia que se encuentran en el apartado de anexos. Por parte, para la validación de DoEverything3D se desarrolló un protocolo de pruebas de integración y cuantitativas.

Las pruebas de integración resultaron exitosas evidenciado la correcta implementación de cada requerimiento del sistema. Las pruebas cuantitativas por su parte comprobaron la buena calidad de los modelos 3D generados por el sistema. Dichas pruebas cuantitativas se realizaron mediante la obtención del error presente entre doce modelos 3D de cajas de diferentes dimensiones generadas por DoEverything3D y su respectivo modelo 3D ideal. Luego, mediante dos pruebas t de Student, se evaluó el error medio y desviación estándar ( $5\sigma$ ) presente en los doce modelos 3D de cajas con la finalidad de permitir concluir que los modelos 3D generados por DoEverything3D presentan un error medio de 0.3988 mm con una desviación estándar ( $5\sigma$ ) de 8.846 mm. La integración de toda la aplicación permite al usuario no depender de varios programas para realizar todo el proceso de modelamiento 3D mencionado, siendo el sistema integrado y compacto para el objetivo final.

Finalmente, se concluye que este sistema demuestra la utilidad que tienen los sistemas de modelamiento 3D de objetos de no contacto, permitiendo generar un modelo 3D de un objeto garantizando su integridad; además da pie a que mucho más público inexperto en el tema se familiarice con el mundo del modelamiento 3D, y se genere mucho más contenido de manera más asequible ya sea en áreas como: videojuegos, cine, industria, comunicación, arqueología, entre otras.

## 6.2. TRABAJO FUTURO

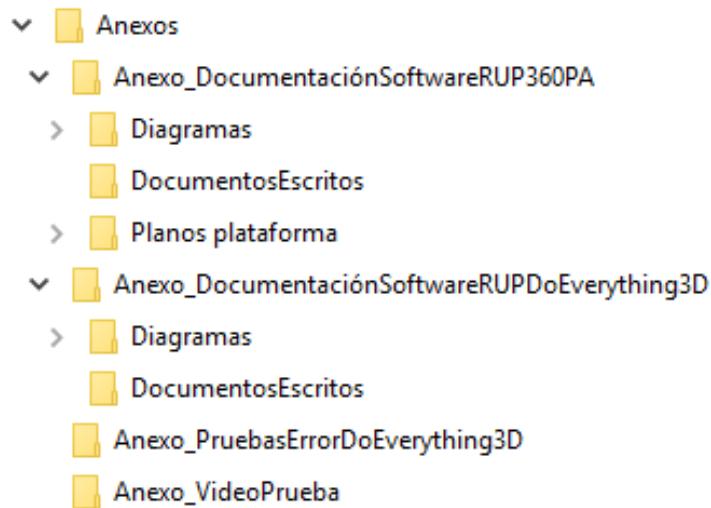
Con el objetivo de mejorar lo realizado en este documento, integrarlo con algún futuro proyecto, corregir limitaciones o mejorar el desempeño del aplicativo se plantean los siguientes trabajos futuros:

- **Modelos con información RGB:** Realizar la integración de información RGB al modelo 3D del objeto, que permita visualizar el modelo 3D del objeto de una manera más parecida a la realidad.
- **Ejecución en tiempo real:** Permitir la generación de modelos 3D en tiempo real, esto con la finalidad de permitir llevar este sistema al modelamiento de objetos estáticos de dimensiones mucho más grandes, en donde el Sensor Kinect rotaria alrededor del objeto a modelar.
- **Registro no-rígido:** Permitir la generación de modelos 3D de un elemento que varíe su forma (Partes del cuerpo humano, animales pequeños, plantas, entre otras). Ya que los algoritmos de registros vistos en este documento solo contemplan transformaciones rígidas (translación y rotación).
- **Objetos con superficies reflectantes:** Permitir la generación de modelos 3D de objetos de vidrio y con superficie reflectante, ya que debido a la tecnología en la que está basado el sensor Kinect (Luz estructurada) no es posible obtener información de profundidad sobre superficies de vidrio o reflectantes.
- **Objetos simétricos:** Permitir el modelamiento 3D de objetos simétricos, ya que los algoritmos de registro que usan únicamente información espacial en las nubes 3D convergen de manera errónea.
- **Generación de superficie:** Con la finalidad de obtener mejores resultados en la generación de superficie (ruido reducido) se plantea la implementación de softwares externos en el sistema como lo es MeshLab [62].
- **Evaluación del error presente en los modelos:** Poder realizar una evaluación de error sobre diferentes modelos 3D generados por el sistema DoEverything3D, usando los modelos obtenidos a partir de un sistema de referencia de altas prestaciones como lo es el EVA de Artec3D [63].

# CAPÍTULO 7

## 7. ANEXOS

A lo largo del documento se citan archivos y programas, dicho material se encuentra presente en la carpeta de anexos del CD entregado y se encuentra seccionado de la forma como se observa en la Figura 7.1.



**Figura 7.1.** Distribución de anexos.

*Anexo\_DocumentacionSoftwareRUP360PA:* Este directorio en sus sub-directorios “Diagramas” y “DocumentosEscritos” se condensa toda la documentación de desarrollo RUP del sistema 360 Platform Assitant (360PA). Dicha documentación RUP consta de: requerimientos funcionales, requerimientos no funcionales, casos de uso reales, diagramas de secuencia y diagrama conceptual. Además, en el sub-directorio “Planos plataforma” se encuentran planos esquemáticos y físicos ampliados del sistema 360PA.

*Anexo\_DocumentacionSoftwareRUPDoEverything3D:* Este directorio en sus sub-directorios “Diagramas” y “DocumentosEscritos” se condensa toda la documentación de desarrollo RUP del sistema DoEverything3D. Dicha documentación RUP consta de: requerimientos funcionales, requerimientos no funcionales, casos de uso reales, Pruebas funcionales, diagramas de secuencia, diagrama conceptual, diagrama de clases y diagrama del modelo de información.

*Anexo\_PruetasErrorDoEverything3D:* En este directorio se condensan las capturas de las 12 pruebas de error realizadas en la sección 5.3.2.

*Anexo\_VideoPrueba:* En este directorio se encuentra un video que evidencia el correcto funcionamiento de la herramienta DoEverything3D.

# CAPÍTULO 8

## 8. REFERENCIAS

- [1] Laserdesign. Cuerpo visible: ¿What is 3D Scanning? Recuperado de <https://www.laserdesign.com/what-is-3d-scanning>.
- [2] 3Deling. Cuerpo visible: ¿What is a point cloud? Recuperado de <http://www.3deling.com/whta-is-a-point-cloud/>.
- [3] Cimec. Cuerpo visible: Curvas y superficies para modelado geométrico, recuperado de [http://www.cimec.org.ar/~ncalvo/curvas\\_doc.pdf](http://www.cimec.org.ar/~ncalvo/curvas_doc.pdf).
- [4] Artec 3D. Cuerpo visible: Applications of 3D scanning. Recuperado de <https://www.artec3d.com/applications>.
- [5] Mayoore Jaiswal, Jun Xie and Ming-Ting Sun, "3D Object Modeling with a Kinect Camera", 2012.
- [6] Amazon. Cuerpo visible: Kinect for Windows, recuperado de <https://www.amazon.com/Microsoft-L6M-00001-Kinect-for-Windows/dp/B006UIS53K>.
- [7] Thomas Schöps (2015). Figura 1.1 Reconstrucción 3D de un edificio mediante visión estereoscópica. Recuperado de <http://people.inf.ethz.ch/schoepst/>.
- [8] Grand view research. Cuerpo visible: 3D Scanning Market Size Is Expected To Be Worth \$8.04 Billion By 2025. Recuperado de <http://www.grandviewresearch.com/press-release/global-3d-scanning-industry>.
- [9] Aniwaa. Cuerpo visible: 3D scanning technologies and the 3D scanning process. Recuperado de <http://www.aniwaa.com/3d-scanning-technologies-and-the-3d-scanning-process/>.
- [10] 3DPRINT. Cuerpo visible: 3D Scanner Buying Guide. Recuperado de <https://3dprint.com/138629/2016-3d-scanner-buying-guide/>.
- [11] Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan and Hao Yan, "Scanning 3D Full Human Bodies Using Kinects", 2015.
- [12] Csaba Kazó and Levente Hajder, "High-quality Structured-light Scanning of 3D Objects using Turntable", 2012.
- [13] Dong Won Shin and Yo Shung Ho, "Implementation of 3D Object Reconstruction using a Pair of Kinect Cameras", 2014.
- [14] Lu li, Zhenjiang Miao, Mangui Liang, "3D reconstruction based on Kinect", 2014.
- [15] Lopez Daniel Escogido, "Escaner 3D de alta precisión", 2016.
- [16] Crespo Julio Marcela, Fernández Zuñiga Ernesto, "diseño e implementación de un escáner 3d para prototipado y modelado geométrico de objetos", 2014.
- [17] Achmad, M.S.H., Findari, W.S.a, Ann, N.Q.b, Pebrianti, D.b, Daud, MR. b, "Stereo camera – Based 3D object reconstruction utilizing Semi-global Matching Algoirthm", 2016.
- [18] Li, S.-R.ab , "Real-time accurate 3D reconstruction based on kinect v2", 2016.
- [19] Valgma L., Daneshmand M., Anbarjafari G., "Iterative closest point based 3D object reconstruction using RGB-D acquisition devices, 2016.
- [20] Morteza Daneshmand, MSc, "3D reconstruction using Kinect v2 camera", 2016.
- [21] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, Slobodan Ilic, "SDF-2-SDF: Highly Accurate 3D Object Reconstruction", 2016.
- [22] Li Guangsong, Lou Jiehong, "A Method of Reconstruction Based on Kinect Camera", 2013.
- [23] Srikanth Varanasi, Vinay Kanth Devu, "3D Object Reconstruction Using XBOX Kinect v2.0", 2016.
- [24] Pedro Miguel Ferro da Costa, "Kinect Based System for Breast 3D Reconstruction", 2014.
- [25] François Pomerleau, Francis Colas, Roland Siegwar, "A Review of Point Cloud Registration Algorithms for Mobile Robotics", 2015.

- [26] Brian Curless. From range scans to 3d models. ACM SIGGRAPH Computer Graphics, 33(4):88-41, 2000.
- [27] INRIA Grenoble Rhône-Alpes, Cuerpo visible: Three-Dimensional Sensors Lecture 4: Time of Flight Cameras (Pulse Light Modulation), Recuperado de [http://perception.inrialpes.fr/~Horau/Courses/pdf/Horau\\_3DS\\_4.pdf](http://perception.inrialpes.fr/~Horau/Courses/pdf/Horau_3DS_4.pdf).
- [28] LMI TECHNOLOGIES, Cuerpo visible: Structured Light vs. Laser Triangulation for 3D Scanning and Inspection, Recuperado de <http://lmi3d.com/company/digital-hub/blog/structured-light-vs-laser-triangulation-3d-scanning-and-inspection>.
- [29] Fofi David, Sliwa Tadeusz and Voysin Yvon, "A comparative survey on visible structured light".
- [30] San José Alonso, J.I., Martínez Rubio, J., Fernández Martín, J.J., García Fernández, J, "COMPARING TIME-OF-FLIGHT AND PHASE-SHIFT", 2011.
- [31] Microsoft, Cuerpo visible: Kinect for Windows Sensor Components and Specifications, Recuperado de: <https://msdn.microsoft.com/en-us/library/jj131033.aspx>
- [32] Microsoft (2011). Figura 2.2. Disposición de los sensores en el Microsoft Kinect. Recuperado de <https://msdn.microsoft.com/en-us/library/jj131033.aspx>.
- [33] P. Sturm, "Pinhole camera model" in Computer Vision, pp. 610-613, Springer, 2014.
- [34] Radu Bogdan Rusu, Nico Blodow, Michael Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration", 2009.
- [35] Lisha Zhang, Manuel Joao da Fonseca, Alfredo Ferreira, "Survey on 3D Shape Descriptors", pp 6, 2004.
- [36] Voxel Art (2014). Figura 2.6. Comparación geométrica entre un pixel y un voxel. Recuperado de <http://voxelart.blogspot.com/2014/04/que-es-voxel-y-el-voxel-art.html>.
- [37] PCL (2014), Figura 2.7. Comparación nube de puntos original y el resultado de su reducción mediante véxeles. Recuperado de [http://pointclouds.org/documentation/tutorials/voxel\\_grid.php#voxelgrid](http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid).
- [38] MathWorks (2009), Figura 2.8. Vectores normales a una superficie 3D. Recuperado de <https://www.mathworks.com/matlabcentral/fileexchange/23063-compute-normal-vectors-of-2-5d-triangulation>.
- [39] PCL (2014), Figura 2.9. Diagrama de región de influencia del cálculo de PFH para un punto de consulta  $pq$ . Recuperado de: [http://pointclouds.org/documentation/tutorials/fpfh\\_estimation.php](http://pointclouds.org/documentation/tutorials/fpfh_estimation.php).
- [40] PCL (2014), Figura 2.10. Marco referencial de coordenadas propuesto en donde se grafican dos puntos  $ps$  y  $pt$  y sus normales asociadas  $ns$  y  $nt$ . Recuperado de [http://pointclouds.org/documentation/tutorials/fpfh\\_estimation.php](http://pointclouds.org/documentation/tutorials/fpfh_estimation.php).
- [41] PCL (2014), Figura 2.11. Ejemplos de representaciones de Histogramas de Puntos para diferentes puntos en una nube. Recuperado de [http://pointclouds.org/documentation/tutorials/fpfh\\_estimation.php](http://pointclouds.org/documentation/tutorials/fpfh_estimation.php).
- [42] PCL (2014), Figura 2.12. Diagrama de la región de influencia para un conjunto k-vecindario centrado en  $pq$ . Recuperado de [http://pointclouds.org/documentation/tutorials/fpfh\\_estimation.php](http://pointclouds.org/documentation/tutorials/fpfh_estimation.php).
- [43] Kai Tang y Tsz-Ho Kwok (2015), Figura 2.13. Registro de 3 nubes de puntos de un objeto (tetera). Recuperado de: <http://manufacturingscience.asmedigitalcollection.asme.org/article.aspx?articleid=2442392>.
- [44] University of Coimbra, Figura 2.14. Registro de 2 nubes de puntos de un rostro humano. Recuperado de: <http://dynface4d.isr.uc.pt/database.php>.
- [45] Szymon Rusinkiewicz and Marc Levoy, "Efficient Variants of the ICP algorithm", 2001.
- [46] P. Besl and N. McKay, "A method for registration of 3D shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, pp. 239-256, 1992.
- [47] Radu Bogdan Rusu, Nico Blodow, Michael Beetz (2009), Figura 2.17. Resultados obtenidos después del registro con SAC-IA en dos vistas parciales del modelo de conejito de Stanford. Recuperado de <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.3710&rep=rep1&type=pdf>.
- [48] F. Lu and E. Milios, "Global Consistent Range Scan Alignment for Environment Mapping," Autonomous Robots, vol. 4, pp. 333-349, 1997.

- [49] J. Sprickerhof, A. Nuchter, K. Lingemann and J. Hertzberg, "A Heuristic Loop Closing Technique for Large-Scale 6D SLAM," Journal for Control, Measurement, Electronics, Computing and Communications, Special Issue with selected papers from the European Conference on Mobile Robots 2009., 2011.
- [50] R.Cupec, R.Grbic, K.E.Nayarko, K.Sabo, R.Scitovski. "Detection of Planar Surfaces Based on RANSAC and LAD Plane Fitting" University of Osijek 2009.
- [51] H. Avron, A. Sharf, C. Greif and D. Cohen, "L1-sparse Reconstruction of Sharpe Point Set Surfaces," ACM Transactions on Graphics, vol. 29, 2010.
- [52] M. Isenburg, Y. Liu, J. Shewchuk and J. Snoeyink, "Streaming Computation of Delaunay Triangulations," in SIGGRAPH, 2006.
- [53] P.-L. George and H. Borouchaki, "Delaunay triangulation and meshing: application to finite elements," 1998.
- [54] J. Gallier, "Notes on convex sets, polytopes, polyhedra, combinatorial topology, voronoi diagrams and delaunay triangulations," 2008.
- [55] H. Edelsbrunner and R. Seidel, "Voronoi diagrams and arrangements," Discrete & Computational Geometry, vol. 1, no. 1, pp. 25–44, 1986.
- [56] D. M. Mount, "Computational geometry notes," 2005.
- [57] P. Cignoni, D. Laforenza, R. Perego, R. Scopigno, and C. Montani, "Evaluation of parallelization strategies for an incremental delaunay triangulator in e3," Concurrency: Practice and Experience, vol. 7, no. 1, pp. 61–80, 1995.
- [58] IMA UDG, Cuerpo visible: Rational Unified Process (RUP), Recuperado de <http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf>.
- [59] S. Choi, Q.-Y. Zhou, and V. Koltun, Robust Reconstruction of Indoor Scenes, CVPR, 2015.
- [60] D. Girardeau-Montaut, Michel Roux, Raphaël Marc, Guillaume Thibault, "Change detection on points cloud data acquired with a ground laser scanner", 2005.
- [61] Instituto tecnico de Chihuahua, Cuerpo visible: TEORIA DE PEQUEÑAS MUESTRAS O TEORIA EXACTA DEL MUESTREO, Recuperado de: <http://www.itchihuahua.edu.mx/academic/industrial/estadistica1/cap03.html>
- [62] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool", 2008.
- [63] Artec3D, Cuerpo visible: Artec Eva Lite, Recuperado de: <https://www.artec3d.com/portable-3d-scanners/artec-eva-lite>.