UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Escáner 3D Móvil utilizando Lego Mindstorms y Dispositivos Android

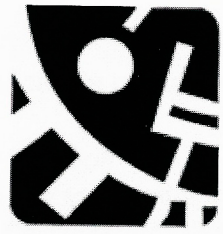
Trabajo Especial de Grado

Iñaki Xabier De Tejada Plessmann y Fernando De Freitas

Tutores: Prof. Eugenio Scalise y Prof. Esmitt Ramírez

Caracas,21deAbril de 2016
Universidad Central de Venezuela

FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN



ACTA DE VEREDICTO

Quienes suscriben, miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por los Bachilleres De Tejada Plessmann Iñaki Xabier y De Freitas Ascanio Fernando Luis, portadores de las cédulas de identidad números V-17555599 y V-18390587, respectivamente, con el título: "Escáner 3D Móvil utilizando Lego Mindstorms y Dispositivos Android", a los fines de optar al título de Licenciados en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 21 de abril de 2016 a las 4 pm, para que sus autores lo defendieran en forma pública, lo que hicieron en Centro de Computación Gráfica, de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

Adicionalmente, se hace constar que el Br. De Freitas Ascanio Fernando Luis realizó la defensa del Trabajo de Grado vía videoconferencia desde la ciudad de Londres, Reino Unido, en condiciones que a juicio del jurado no comprometieron la validez académica del acto.

En fe de lo cual se levanta la presente Acta, en Caracas a los ventiún días del mes de abril del año dos mil dieciséis.

Prof. Esmitt Ramirez
Tutor

Prof. Eugenio Scalise
Tutor

Prof. Andrés Sanoja
Jurado

Prof. Francisco Moreno
Jurado

Agradecimientos y Dedicatorias

A mis padres, Soneac e Iñaki, y a mi familia por apoyarme a pesar de no saber muy bien qué es lo que estaba haciendo con una Jirafa y un robot.
A Ignacio, por el challengemodesupport y especialmente por el robot!
A mis amigos, Alexander, Shei, Mantu, Sarpi, Lizardo, Aintzane (y los 4 millones de apodos que le puse), Goyo (doble mención jajaja), Carlo, Nunzio, Angela, Christian, Michelle, Juan, Joe, Kari, Alfonso, Jesús, Gustavo, Maria y Reiner por el ánimo, apoyo y birras aportadas durante el proceso.
A Adelis, Cesar, Masa y demás por preguntar cómo iba este proyecto y reírse cada vez que nos veían sufrir.
Este TEG va dedicado a Carmen, mi abuela, quien no pudo llegar a verme en el Aula Magna pero que se hubiese alegrado mucho por estar en ese momento.
Iñaki

Para Clara Ascanio, José De Freitas, Agustín De Sousa
y José De Sousa†, mis ídolos, mis héroes y modelos a seguir.
Esto es de ustedes y para ustedes
Y un agradecimiento muy especial a la señora SoneacPlessmann,
por todo el apoyo y hacerme sentir como otro hijo suyo más.
A todos, muchas gracias.
Fernando

Resumen

La popularización de las impresoras 3D, en los últimos años, ha creado la necesidad de encontrar nuevas formas de generar modelos 3D de objetos de la vida real y la creación de repositorios donde acceder a éstos o compartirlos. El propósito de este trabajo fue diseñar e implementar un escáner 3D sin contacto, basado en un robot, teléfonos y una tableta Android para capturar diferentes ángulos del objeto, procesar y finalmente desplegar el modelo generado, utilizando herramientas de software gratuitas y de código abierto, así como el sistema operativo Android y su SDK (*Software Development Kit*), desarrollando un conjunto de aplicaciones para los dispositivos, que utilizando comunicación via Bluetooth, puedan aplicar técnicas de procesamiento digital de imágenes y operaciones de cálculo numérico sobre matrices, mediante bibliotecas especializadas, tales como OpenCV y VTK, para escanear objetos y generar modelos 3D de los mismos, de forma autónoma, utilizando un robot Lego Mindstorms y una aplicación hecha con la máquina virtual especial de Java llamada LeJOS, para controlar su movimiento y comunicación. Los resultados obtenidos fueron mixtos, ya que no se pudo generar el modelo 3D final, dejando un escáner que sólo captura y segmenta el objeto.

Palabras Clave: Robótica, Procesamiento de Imágenes, Escaneo 3D, Lego Mindstorms, Android

Tabla de Contenido

Lista de Figuras

Introducción

Con el avance de la tecnología y la amplia difusión de las ciencias de la computación, la gran cantidad de personas interesadas en el área de Inteligencia Artificial, robótica y demás disciplinas similares, así como el mercado para los productos derivados de éstas, se muestra un cambio en la brecha entre la tecnología y la población del mundo.

El momento en el cual los robots puedan encargarse de la tareas cotidianas tal como sucede en las obras de ciencia ficción es muy distante. Sin embargo, desde hace muchos años se emplean robots en la industria, las líneas de producción de las principales compañías automotrices los usan para ensamblaje, para la elaboración de piezas que necesiten de una precisión milimétrica y para hacer mediciones de control de calidad. También en medicina se usan robots para realizar cirugías de manera remota y en casos donde la precisión es fundamental.

Existen otros campos donde también son utilizados actualmente los robots, en la medicina se utilizan para realizar intervenciones quirúrgicas de manera remota. En el ejército y la policía se emplean para misiones riesgosas como desactivar bombas o para combate y reconocimiento, como es el caso de los *Drones* o UAVs (*UnmannedAerialVehicle*).

Hasta ahora se han enumerado casos donde son usados para tareas de alto impacto y notoriedad, pero su alcance también llega a tareas más simples y poco críticas. Esto genera la necesidad de robots de bajos costos y de fácil acceso al público en general.

A partir de esta necesidad varias compañías dirigen su atención a ese mercado, con un gran déficit de oferta y un alto potencial de rentabilidad, y decididas a sacarle provecho comenzaron a buscar formas de satisfacer esta demanda. Para ello era necesario hacerlos más simples de diseñar y programar, ya que hasta hace pocos años sólo aquellos grupos con conocimientos avanzados en electrónica y mecánica podían incursionar en el campo de la creación de robots.

Es así como nacen Lego Mindstorms, Arduino y Raspberry Pi, entre otras alternativas de hardware a bajo costo. De éstos, el Lego Mindstorms es uno de los más populares gracias a su enfoque didáctico, a la fama de la marca Lego y al trabajo de la comunidad para explotar el potencial del bloque.

En este trabajo se propone el diseño e implementación de un sistema compuesto por un robot, cámaras y un controlador, utilizando un robot LEGO Mindstorms y dispositivos Android, para cumplir el objetivo de escanear objetos de forma autónoma.

En el capítulo 1 se plantea el problema con sus objetivos general y específicos, como el necesario movimiento omnidireccional del robot. En el capítulo 2 hace una reseña histórica al bloque Mindstorms y las especificaciones de la versión utilizada para el desarrollo, así como la definición de algunos conceptos básicos del contexto de procesamiento de imágenes y las herramientas utilizadas en el desarrollo. En el capítulo 3 se enumeran trabajos previos similares en el área. En el capítulo 4 se plantea el diseño del sistema. En el capítulo 5 el desarrollo el mismo en sus diferentes fases. En el 6 se documentan las Pruebas y Resultados. Y, finalmente en el 7 y 8 se plantean las Conclusiones así como Trabajos Futuros.

Capítulo 1 - Planteamiento del Problema

En este capítulo se presenta la propuesta de desarrollo del escáner 3D utilizando la capacidad del bloque NXT para comunicarse por Bluetooth con otros dispositivos, que se ocupan de la captura y procesamiento de imágenes, las cuales serían transmitidas a un dispositivo central para hacer procesamiento de los datos y presentar el resultado.

1.1 El Problema

En los últimos años las impresoras 3D han logrado gran popularidad dada la capacidad de generar prácticamente cualquier objeto a partir de un modelo generado por computadora, estos objetos pueden ser tan simples como un cubo o tan complejos como un brazo mecánico. En sitios web como *Thingiverse*[1] y *3D PrintingModelMarketplace*[2], se pueden conseguir un gran número de modelos que se pueden imprimir y utilizar.

Para generar estos modelos a partir de objetos cotidianos es necesario disponer de un escáner 3D y un ambiente controlado donde hacer el proceso. Existen 2 tipos de escáner 3D: en contacto con el objeto y sin contacto. Un escáner de contacto es preciso pero no muy efectivo ante objetos maleables. Los que no requieren contacto pueden afectar al objeto dependiendo de la técnica, una de estas técnicas requiere que el objeto sea sumergido en líquido lo cual podría dañarlo.

Teniendo en cuenta estos enfoques surge la idea de desarrollar una propuesta de escáner 3D móvil sin contacto, con componentes de bajo coste disponibles en el mercado, que emplee fotografías tomadas con teléfonos móviles para generar un modelo 3D del objeto escaneado.

Al ser móvil haría relativamente sencillo al usuario escanear objetos del mundo real, siendo sólo necesario tener consigo el robot, los dispositivos Android que funcionen como cámaras y una tableta Android para controlarlo.

Una vez obtenidos los modelos generados por el robot, así como para imprimirlos con una impresora 3D, podrían ser utilizados para otras aplicaciones, por ejemplo en software para modelos de videojuegos, despliegues de material audiovisual y diseño gráfico en general.

[1] http://www.thingiverse.com
[2] http://3dprintingmodel.com

1.2 Objetivo General

Diseñar e implementar un escáner 3D sin contacto basado en un robot, teléfonos y una tableta Android para capturar diferentes ángulos del objeto, procesar y finalmente desplegar el modelo generado.

1.3 Objetivos Específicos

- Diseñar un robot que se traslade alrededor de un objeto en un número determinado de ángulos.
- Implementar una aplicación que le permita al robot desplazarse omnidireccionalmente y medir su distancia a otros objetos.
- Implementar una aplicación Android que capture imágenes bajo demanda y realice el pre-procesamiento necesario a las mismas para los pasos previos a la creación del modelo 3D en 2 dispositivos Android para generar una visión estereoscópica del objeto a escanear.
- Implementar una aplicación Android que se comunique con el robot y con los dispositivos Android para coordinar el proceso de captura de imágenes, que procese dichas imágenes para generar un modelo 3D del objeto y presente éste como una visualización que el usuario pueda manipular mediante la aplicación.
- Implementar comunicación por Bluetooth entre todos los dispositivos que conforman el sistema.

Para cumplir estos objetivos se construirá un sistema con un robot Lego Mindstorm NXT, dos teléfonos y una tableta Android. Desde la tableta se controlará el robot el cual se desplazará alrededor del objeto deteniéndose una cantidad específica de veces para capturar las caras hasta cubrir 360º, una vez capturados todas las caras del objeto se procederá a reconstruir el objeto a partir de los datos provista por las cámaras.

Capítulo 2 - Marco Teórico

El diseño y desarrollo de este sistema comprende un conjunto extenso de componentes tanto de hardware como de software que es necesario estudiar con cierto nivel de detalle para tener una mejor comprensión del mismo.

2.1 Lego Mindstorms

En los últimos 60 años Lego ha sido la compañía líder en entretenimiento infantil, manteniéndose a la vanguardia de la tecnología para conservar este puesto. Con el pasar de los años la robótica ha sido una de esas tecnologías vanguardistas que se ha convertido en el nuevo campo a explorar por la empresa para satisfacer a las nuevas generaciones. En esta sección se hace una reseña del origen del Lego Mindstorms y sus primeros prototipos no comerciales, se describen las diferentes versiones que han salido al mercado, junto con sus especificaciones técnicas.También se presentan al final de este capítulo las principales alternativas a éstos que se encuentran actualmente en el mercado.

2.1.1 Reseña Histórica

El Lego MindstormsRoboticInvention Kit, conocido también como “set Mindstorms”es el resultado de la cooperación entre un conjunto de institutos y empresas para proporcionarle al mundo académico, y el público en general, una línea de productos de robots programables que disminuyeran las barreras de entrada al mundo de la electrónica y la programación así como impulsar dichos campos [1]. En esta sección se muestra el origen de esta cooperación y los resultados que han logrado que el set Mindstorms se haya hecho tan popular. Así como también una breve reseña de las versiones más importantes de esta línea de productos de Lego, así como algunas las alternativas existentes en el mercado.

2.1.1.1 Cooperación Lego- MIT Media Lab - Resnick and Papert’s Epistemology and Learning Research Group (1987)

El set Mindstorms es el resultado de un esfuerzo conjunto entre Lego, el MIT Media Lab (MIT-ML) y el Resnick and Papert’sEpistemology and LearningResearchGroup (RAPEALG) que en el año 1987 unieron fuerzas para desarrollar un bloque programable que se ajustara a las necesidades que tenía cada uno de los involucrados.

Lego quería recuperar parte del mercado que había perdido frente a los juguetes electrónicos y consolidarse como la marca líder en el mercado infantil, el RAPEALG quería plantear nuevos modelos de aprendizaje y el MIT-ML quería crear un modelo de investigación que fuera atractivo al público y sirviera de trampolín al desarrollo de la robótica.

Esta no fue una tarea sencilla, para desarrollar un bloque programable se debian tomar en cuenta muchos aspectos técnicos en el diseño del hardware y software:

- En cuanto al software, se decidió utilizar el lenguaje de programación Logo [2], desarrollado por Bobrow et al. del grupo RAPEALG en el año 1960, como un lenguaje de programación para niños, orientado al aprendizaje y con una sintaxis simple pero poderosa que sigue la filosofía "*low-thresholdhigh-ceiling*" de Papert [3],el compilador de Logo, para la versión RCX, fue desarrollado por Brian Silverman.
- Para el hardware, Fred G. Martin fue el principal encargado del diseño junto con otros estudiantes del MIT-ML, quien presentó muchas versiones del bloque hasta que en 1996 surgió la versión final, un bloque rojo con cuatro puertos de salida y seis puertos de entrada conocido simplemente como el bloque "rojo" (ver Figura 2.1). Este bloque contaba con seis puertos de entrada, cuatro motores, una entrada infrarroja y una salida de audio. Además se podían agregar dispositivos externos como un micrófono o una salida infrarroja.

Figura 2.1. Bloque "rojo", creado por Martin en 1996.

Luego de haber demostrado que el bloque rojo cumplia con las necesidades de Lego, MIL-ML y RAPEALG se decidio salir al mercado con el modelo RCX, este fue seguido por el NXT en el año (2006) y la tercera generación nació con el EV3 (2013). Para el desarrollo del Escáner 3D de este Trabajo Especial de Grado. se seleccionó el modelo NXT, porque tenía mayor soporte

técnico y herramientas de desarrollo disponibles al momento.

2.1.2 NXT

El NXT representa un gran avance con respecto al RCX. En este bloque decidieron no simplificar sino agregar nuevas habilidades, por ejemplo el uso de Bluetooth y de I²C, un protocolo de bus serial utilizado para interactuar con periféricos de baja velocidad (100 kbps). El bloque tiene 3 puertos de salida y 4 puertos de entrada (uno más que el RCX). Otro cambio del NXT fue en los conectores, el NXT abandonó los conectores parecidos a un bloque de Lego y pasó a usar conectores RJ11 (ver Figura 2.2).

Figura 2.2. Bloque NXT con sensores y accesorios.

El NXT se comercializa en un kit llamado Lego Mindstorms NXT 2.0, el cual consiste del bloque NXT, un sensor ultrasónico, un sensor de color, dos sensores de contacto, tres servomotores, un cable USB, cables con conectores RJ11, un manual de instrucciones, el software básico para ensamblar programas y 619 piezas de Lego.

El NXT representó un gran avance en el mundo de los Mindstorms, el uso de RJ11 en los conectores permitió que muchas compañías externas a Lego (y miembros de la comunidad con conocimientos en electrónica) pudieran crear nuevos sensores. Esto disparó el potencial del NXT para aplicar soluciones a miles de problemas. La integración de Bluetooth también permite que se puedan utilizar varios NXT para una tarea.

En el mercado se consiguen 3 compañías principales en el desarrollo de sensores y accesorios para el NXT:

- **Hitechnic[3]:** Fue la pionera en el desarrollo de sensores no oficiales, al punto de tener un trato con Lego para obtener las carcasas plásticas de los sensores oficiales para utilizarlos en sus sensores pero con colores diferentes (negro y gris) para diferenciarse.
- **Mindsensors[4]:** Tiene más variedad de sensores a un precio más reducido, y dan la posibilidad de comprar los conectores RJ11 que utiliza el NXT. No tiene convenio con Lego para obtener las carcasas de plástico lo que hace que sus sensores no mantengan el estilo visual y que parezcan más como tarjetas de circuitos conectadas a través de pines al NXT.
- **Vernier[5]:** Es una compañía que se especializa en hacer sensores químicos para cualquier tipo de necesidad, estos sensores no están diseñados para el NXT, pero al ver el potencial de ventas decidieron desarrollar un adaptador que permitiera conectar cualquiera de sus sensores.

Las especificaciones del bloque NXT se indican a continuación:

- 1 Microcontrolador ARM7 de 32 bits con 256 KB de memoria flash y 64 KB de memoria RAM.
- 1 Microcontrolador AVR de 8 bits con 4 KB de memoria flash y 512 B de memoria RAM.
- 7 puertos RJ12, 4 de entrada (identificados del 1 al 4) y 3 de salida (identificados de la A a la C).
- Pantalla LCD de 100 x 64 píxeles
- Corneta con sonido de 4 Hz y resolución de 8 bits
- 4 botones, izquierda, derecha, arriba, abajo, seleccionar y apagar, no etiquetados
- Bluetooth clase 2 versión 2.0
- USB 1.0 Full Speed

2.4 Escáneres y tipos

Los escáner 3D pueden clasificarse de 2 formas [4]:

[3] www.hitechnic.com
[4] www.mindsensors.com
[5] www.vernier.com/NXT/

- De contacto, que crean el modelo 3D mediante tacto con el objeto, utilizando piezas especiales como agujas con sensores de presión.
- Sin contacto con el objeto, que utilizan principalmente cámaras, iluminación y sensores especiales (como cámaras y laser) para el reconocimiento del objeto, a su vez estos se separan en 2 grupos. Los pasivos que son aquellos que no emiten ningún tipo de luz (e.g. cámaras) y los activos que utilizan alguna fuente de luz para medir el objeto (e.g. láser) .

2.5 Procesamiento Digital de Imágenes

Una imagen se puede definir como la representación de un objeto la cual puede ser captada de diversas formas ya sea por los sentidos de un humano o los sensores de un computador, que pueden ser almacenadas en la memoria humana o en un dispositivo electrónico, y reproducidas ya sea en una impresión o una representación artística.

Dando una definición técnica, una imagen es una función que calcula la distribución de la intensidad lumínica de un objeto. Esta función puede representar una imagen monocromática de la forma f(x, y) donde x,y corresponden a un punto del objeto y f calcula el valor de la intensidad lumínica de dicho punto. Para imágenes cromáticas el valor de la intensidad lumínica viene definido por los valores de los canales Rojo, Verde y Azul (RGB) de un punto, por lo cual se definiría como 3 funciones de 2 variables de la forma $f_r(x, y)$, $f_g(x, y)$ y $f_b(x, y)$.

Cada uno de los puntos de una imagen reciben el nombre de píxel (contracción del término inglés *pictureelement*) estos píxeles pueden contener la información de una imagen monocromática o cromática. Los aspectos más importantes de una imagen son:

1. **Profundidad de color**: Se define como la cantidad de bits que definen a un píxel y sirve para calcular el máximo número de colores que puede tener una imagen. Una imagen monocromática puede tener 1 ó 2 bits para ser representada, para blanco y negro o escala de grises respectivamente. En una imagen cromática se definen los bits por canal y pueden ser representadas a partir de 3 bits.
2. **Resolución**: La resolución es la cantidad de píxeles por unidad de longitud de una imagen y por lo general se define como los píxeles por pulgada. Mientras mayor sea la resolución de una imagen, mejor será su definición.

2.5.1 Técnicas de Procesamiento Digital

Se define como el conjunto de técnicas que permiten manipular una imagen y comprende un gran espectro. Las técnicas se pueden clasificar por la cantidad de pixeles en las cuales trabajan.

1. **Procesamiento puntual**: Estas técnicas operan sobre un píxel a la vez, las operaciones más comunes de este tipo suelen ser las de modificación de brillo y contraste de la imagen.
2. **Procesamiento grupal**: Son técnicas que operan un píxel usando información de los píxeles que lo rodean, para esto se definen máscaras que calculan la influencia que tendrá cada píxel que rodea al que está siendo calculado. Estas técnicas suelen usarse para atenuar o realzar detalles de una imagen o hacer segmentación de elemento.

Del subconjunto de técnicas de procesamiento grupal resaltan las siguientes:

1. **GrabCut**: Es una técnica creada por Carsten et al. [5] para segmentar una imagen en primer plano (*foreground*) y fondo (*background*) minimizando la interacción del usuario. Se basa en cálculos probabilísticos para definir el primer plano(*foreground*) de una zona definida por el usuario a través de un rectángulo, todo lo que esté fuera del rectángulo se asume como fondo (ver Figura 2.3).

Figura 2.3. Ejemplo de funcionamiento de GrabCut.

2. **Algoritmo de Canny**: Es un método para detección de bordes de un objeto en una imagen usando la primera derivada para detectar los cambios de intensidad. El principio del algoritmo es que un cambio en la intensidad genera un cambio brusco en la primera derivada y se considera como un borde (ver Figura 2.4).

Figura 2.4. Ejemplo de funcionamiento de Canny.

2.5.2 Vision estereoscópica

La visión estereoscópica es un procedimiento para la determinación de la forma de los objetos de una escena. Se basa en el modelo estereoscópico biológico, el cual muestra que la separación entre los ojos permite calcular la profundidad a la que se encuentra un objeto en la realidad a través de la triangulación de las 2 imágenes capturadas (ver Figura 2.5).

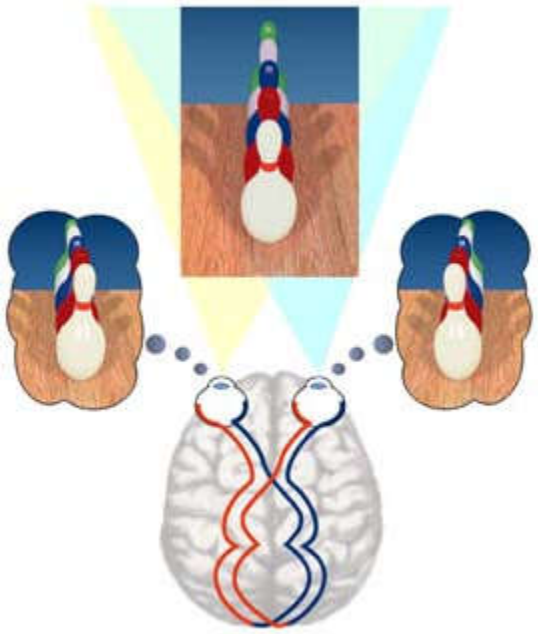


Figura 2.5. Diagrama de visión estereoscópica.

Para replicar este efecto se usan 2 cámaras con una separación definida apuntando al mismo punto. Usando este esquema debe buscarse la correspondencia de los puntos de la imagen de cada cámara.

Dada una secuencia de imágenes de un misma escena tridimensional capturadas desde diferentes ángulos se puede conseguir un conjunto de puntos en una imagen A que son correspondientes a una imagen B, esta relación entre los puntos de las imágenes 2D de una escena tridimensional se encuentra en el flujo óptico de la imagen el cual se define como el movimiento aparente causado por la interacción entre el espectador y la escena.

Este flujo óptico puede ser calculado usando el algoritmo de Lucas-Kanade Piramidal [6]. Este algoritmo, propuesto por Lucas y Kanade, fue planteado para calcular el flujo óptico en videos por lo cual solo funciona con imágenes con poca distancia entre sí.

Para esto se plantea el uso de pirámides basada en la reducción de tamaño de la imagen original y a partir de ahí se calcula el flujo óptico entre 2 imágenes para la región de interés avanzando desde la regiones con poca información para generar información para el siguiente nivel de la imagen.

2.5.3 Triangulación

Dadas las matrices de proyección P y P' obtenidas a partir del proceso de calibración, y un conjunto de puntos en correspondencia, entonces es posible obtener los puntos 3D originales. Uno de los problemas que se encuentran es que los rayos que unen las correspondencias con los centros de las cámaras generalmente no intersectan en un punto, sino que determinan rectas convergentes que no se llegan a tocar. Esto es así debido a que la precisión de las imágenes es finita y a que las correspondencias no son exactas.

El método más simple para reconstruir un punto es la triangulación. Este método consiste en determinar los dos rayos que unen cada correspondencia con el foco de la cámara y después encontrar el punto cuya distancia a dichas rectas sea mínima. Las proyecciones $x = PX$ y $x' = P'X$ se pueden combinar de tal manera sistema de la forma $AX = 0$. Para eliminar el factor de escala se utiliza el producto escalar de la forma $x \times PX = 0$ y se obtiene lo siguiente

$$x(p^{3T}X) - (p^{1T}X) = 0$$
$$y(p^{3T}X) - (p^{2T}X) = 0$$
$$x(p^{2T}X) - y(p^{1T}X) = 0$$

Estas ecuaciones son lineales en los componentes de X y una de ellas depende linealmente de las otras dos. Al introducir la información del otro punto, entonces se obtiene la matriz

$$A = \begin{pmatrix} xp^{3T}-p^{1T} \\ yp^{3T}-p^{2T} \\ x'p'^{3T}-p'^{1T} \\ y'p'^{3T}-p'^{2T} \end{pmatrix}$$

donde p^{nT}y $p'^{nT}n = 1,2,3$ son los vectores fila de las matrices P y P'. Para resolver el sistema se puede utilizar la descomposición SVD (Singular ValueDecomposition).

2.5.4 Descomposición de Valores Singulares

La descomposición por valores singulares (SVD) es una técnica muy utilizada en el campo de la visión artificial para descomponer matrices. Una matriz A puede descomponerse de la siguiente forma:

$$A = UDV^T.$$

Donde U y V son matrices ortogonales, y D es una matriz diagonal cuyos valores $(d_1, \ldots, d_n)$ son los autovalores de A.

Es importante destacar que la descomposición por autovalores solamente existe para matrices cuadradas. Sin embargo, también se puede realizar esta descomposición en matrices no cuadradas.

La Descomposición por Valores Singulares puede definirse para cualquier matriz $A \epsilon \Re^{mxn}$ donde existe una descomposición del tipo $A = UDV^T$ tal que:

- U es una matriz mxn con columnas ortogonales. Sus columnas son los valores singulares izquierdos de A.
- D es una matriz diagonal nxn con todos sus valores positivos. Tiene todos sus elementos igual a 0, a excepción de los elementos $min(m,n)$ de su diagonal principal.
  Esos valores son reales y están ordenados de mayor a menor, y reciben el nombre de valores singulares de A.
- V^T es una matriz ortogonal nxn. Los valores de la diagonal de V se llaman valores singulares derechos de A.

Esta descomposición se puede representar como:

$$[\square] = [\square][\square][\square\square]$$

Dicho de otra manera, consiste en aplicar a la matriz A una transformación lineal y ortogonal, de manera de obtener una matriz $\square\ \square\ \square^{\square\square\square}$que preserva la norma cuadrada de A, pero utilizando pocas columnas.

La técnica SVD permite resolver fácilmente sistemas de ecuaciones lineales sobredeterminados por mínimo error cuadrático, tanto homogéneos como no homogéneos.
En el caso de un sistema homogéneo $Ax = 0$, las filas de Vcorrespondientes a los menores valores singulares son vectores unitarios que minimizan $||Ax||^2$sería el espacio nulo, pero debido a errores numéricos u otras causas, los valores singulares que deberían ser cero no lo son exactamente. Aún así, las filas de Vcorrespondientes a los menores valores singulares son las mejores aproximaciones a la solución en el sentido de mínimo error cuadrático.
Calcular la SVD consiste en encontrar los valores propios y los vectores propios de AA^T y A^TA. Los vectores propios de A^TA forman las columnas de V y los vectores propios de AA^Tlas columnas de U. Además, los valores singulares en S son las raíces cuadradas de los valores propios de AA^To A^TA.
Los vectores propios, autovectores o *eigenvectores* de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección. Este escalar λ recibe el nombre de autovalor.

2.6 Herramientas de Desarrollo

2.6.1 Android SDK

El *Software Development Kit* (SDK) [7] de Android comprende un conjunto de bibliotecas, documentación, herramientas y APIs necesarias para desarrollar, compilar, probar y depurar aplicaciones para el Sistema Operativo Android.
Aunque no es obligatorio el uso de un IDE para desarrollar aplicaciones para Android, se puede utilizar cualquier editor de texto para desarrollar y una interfaz de línea de comandos para compilar. Este IDE provee soporte para herramientas que facilitan tanto la estructuración de los proyectos como su correcta compilación y mantenimiento.
Entre dichas herramientas se encuentran:

- Apache Ant, que permite automatizar la compilación y mantenimiento, similar a make para C/C++,
- Apache Maven, al igual que Ant, permite automatizar tareas de compilación y mantenimiento, de una manera más simple ya que su configuración es almacenada en archivos XML, está enfocado a ciclos de vida del proceso de compilación e incluye un Project ObjectModel (POM) donde se describe el proyecto, sus dependencias, componentes externos e incluso el orden de compilación. Está diseñado para funcionar en red y puede administrar repositorios externos, tanto para descargar dependencias u

otros elementos como para subir el resultado final de la compilación a repositorios preestablecidos.

- Gradle,proporciona la misma funcionalidad de las dos herramientas previas, pero utilizando archivos en Groovy en vez de XML y utiliza un Grafo Acíclico Dirigido (DAG), como Ant, para determinar el orden como deben ser ejecutadas las tareas. Soporta compilación de multi-proyectos así como también compilación incremental, determinando cuales compilaciones de componentes están actualizadas para evitar ejecutar las tareas que afectan a esos componentes.

Otras herramientas incluidas en el SDK son Fastboot, utilizada para modificar el contenido de la memoria Flash de los dispositivos mediante USB, empleando la línea de comandos y AndroidDebug Bridge (ADB) que le permite a PC y dispositivos Android comunicarse mediante línea de comandos, muy parecido a como funciona un Secure Shell (SSH).

2.6.2 Android NDK

El AndroidNativeDevelopment Kit [8] es un conjunto de herramientas que permite implementar parte de una aplicación Android utilizando lenguajes nativos como C/C++. Su propósito es dar la posibilidad de interactuar directamente con el procesador en partes críticas de aplicaciones que hagan uso intensivo de la CPU y la GPU para manejar el uso de lo mismos de una manera eficiente. Ejemplos de este tipo de aplicaciones son videojuegos y simulaciones matemáticas.

2.6.3 LeJOS NXJ

LeJOS NXJ es una API para desarrollar programas para los bloques en lenguaje Java. Estos programas corren sobre la máquina virtual LeJOS JVM, escrita en C. LeJOS permite crear programas más complejos que NXT-G. Entre las funcionalidades que provee leJOS se encuentran [9]:

- Programación Orientada a Objetos
- Hilos de ejecución
- Arreglos multidimensionales
- Recursión
- Sincronización
- Excepciones
- Todos los tipos de datos de Java
- La mayoría de las funciones de los paquetes `java.lang, java.util y java.io`

- Documentación de la API
- Paquetes para manejar Sensores y Motores
- PC API

Al ser una API para Java, LeJOS permite desarrollar desde cualquier sistema operativo, aprovechándose de la cualidad multiplataforma de este lenguaje. Y no es limitativo para el uso en Mindstorms, ya que también se puede desarrollar en LeJOS para GameBoy (excluyendo los elementos del paquete NXJ).

Aparte de su API principal también posee una PC API, el cual permite la ejecución de un gran subconjunto de instrucciones de la API principal, como mover los motores o reproducir sonidos en el bloque, desde otros dispositivos con Java mediante el uso de la tecnología Bluetooth. Es llamada PC API debido a que inicialmente estaba enfocada para PCs, pero debido a la expansión de los teléfonos inteligentes, éstos también se ven incluidos dentro de su alcance, siendo los dispositivos Android los más usados para su integración con bloques Mindstorms debido a que también funcionan sobre Java.

Para hacer que un dispositivo Android y un bloque Mindstorms interactúen entre ellos se puede utilizar la PC API de leJOS [10], que utiliza comunicación mediante Bluetooth, desde Android, esto se puede lograr de dos maneras:

- Comunicándose con programas hechos en leJOS corriendo en el Mindstorms. Esto requiere que el firmware de leJOS se encuentre instalado en el bloque
- Comunicándose con el bloque mediante el Protocolo de Comunicación de LEGO (LCP). Esto no requiere tener el firmware de leJOS en el bloque

La única limitación de utilizar estas bibliotecas es que no proporcionan soporte para el paquete robotics de la API de leJOS, lo cual impide la manipulación directa de motores o sensores desde la aplicación de Android, pero que puede ser solucionado utilizando técnicas para controlar el bloque, como paso de mensajes, para enviar órdenes y obtener información de dichos componentes.

2.6.4 OpenCV

OpenCV[6] (Open SourceComputerVision Library) es una biblioteca especializada en visión computacional, está escrita en C/C++ y ofrece *wrappers* o *bindings* para C++, C, Python y Java. Gracias a esta variedad de lenguajes, puede ser utilizado en aplicaciones para Windows, Linux, Mac OS, iOS y Android.

[6] http://opencv.org/

Dado que se publica bajo la licencia BSD, puede ser utilizada tanto para fines comerciales como para fines académicos. OpenCV fue diseñada teniendo en cuenta el procesamiento en tiempo real, lo que hace que sea altamente eficiente. Entre sus principales funcionalidades se encuentran:

- Procesamiento de imágenes
- Procesamiento de videos
- Diseño de GUIs
- Reconstrucción de objetos 3D
- Detección de objetos
- Machine Learning

2.6.5 VTK

VTK[7] (VisualizationToolkit) es una biblioteca multiplataforma para el procesamiento de imágenes y visualización 3D, escrita en C++, con interfaces para Java y Python. Entre los principales algoritmos que implementa esta biblioteca se encuentran:

- Algoritmos de visualización, incluyendo escalares, vectoriales y de texturas
- Métodos volumétricos
- Algoritmos de modelado avanzado, como modelado implícito, reducción de polígonos y triangulación de Delaunay.

También posee un extenso *framework* de visualización, *widgets* 3D y soporte para procesamiento en paralelo. Al igual que OpenCV, VTK se distribuye con la licencia BSD lo que permite su uso en cualquier tipo de proyectos.

2.6.6 CGAL

CGAL[8] (*ComputationalGeometryAlgorithms Library*) es una biblioteca compuesta por un conjunto de algoritmos de geometría, implementados en C++, estos algoritmos han sido implementados eficientemente para hacer que esta biblioteca sea robusta y confiable. Las principales funcionalidades de CGAL son:

- Estructuras de datos geométricas
- Algoritmos de búsqueda
- Algoritmos de ordenamiento espacial

[7] http://www.vtk.org/
[8] http://www.cgal.org/

CGAL es distribuida con un esquema de doble licencia. Puede ser utilizada como software de código abierto, gracias a que su código base está disponible bajo licencia LGPL y los niveles superiores de la biblioteca bajo GPL, o de manera comercial, en aquellos casos en que el uso que se le vaya a dar se encuentre limitado por las licencias LGPL y GPL, adquiriendo una licencia comercial en *GeometryFactory*[9].

2.12 Bluetooth

Bluetooth [11] es una especificación industrial para Redes Inalámbricas de Área Personal (o WPAN), su denominación según IEEE es 802.15.1, aunque no es mantenida por IEEE sino por el Bluetooth SpecialInterestGroup (o Bluetooth SIG). Esta especificación y tecnología está enfocada a la transmisión de datos en distancias cortas. Utilizando ondas UHF de radio a 2.4GHz.

Fue inventado por la empresa Ericsson en 1994. Su nombre es en honor al rey Harald Bluetooth, quien unió a las tribus danesas en un solo reino. Y se eligió bajo la implicación de que esta tecnología hace lo mismo con los protocolos de comunicación, uniéndolos en un estándar universal. Su logo proviene de las iniciales de dicho rey en forma de runas, ᚼ ᛃ ᛒ .

Está clasificado en clases, las cuales definen el alcance del dispositivo, y versiones, las cuales definen la velocidad de transferencia.

La tecnología Bluetooth también define perfiles. Mediante los cuales se especifican diferentes tipos de dispositivos como auriculares, módems, teclados, ratones, controles remotos, micrófonos, impresoras, entre otros.

Un dispositivo actuando como “maestro” puede mantener un máximo de 7 conexiones simultáneas con otros dispositivos.

[9] http://geometryfactory.com/

Capítulo 3 - Trabajos Previos

La idea de un escáner 3D ya ha sido implementada previamente usando diferentes tipos de escáner, técnicas y enfoques, dando resultados con diferentes grados de calidad. A continuación algunos ejemplos de escáneres 3D de ambos tipos y relativamente de bajo costo.

3.1. Lego NXT 3D

Entre los escáneres de contacto se encuentra este desarrollado por PhilippeHurbain [12] usando un Lego Mindstorms NXT. El sistema consiste de una PC, un NXT, y dos plataformas, una donde se monta el objeto a escanear y otra donde se encuentra el sensor de contacto (ver Figuras 3.1 y 3.2)

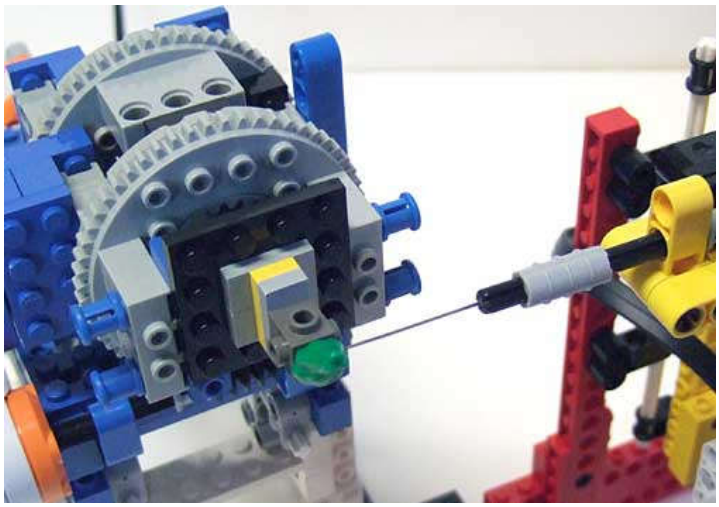


Figura 3.1. Plataformas del sistema, objeto a escanear y sensor de contacto del Lego NXT 3D.

El funcionamiento de este escáner es simple, utilizando la aguja, montada sobre un componente de Lego Technic llamado actuador lineal, conectada a un motor, registra la profundidad de los puntos del objeto con los que hace contacto, enviando esta información a la PC, la cual se encarga de procesar los datos para generar un archivo CAD del modelo.

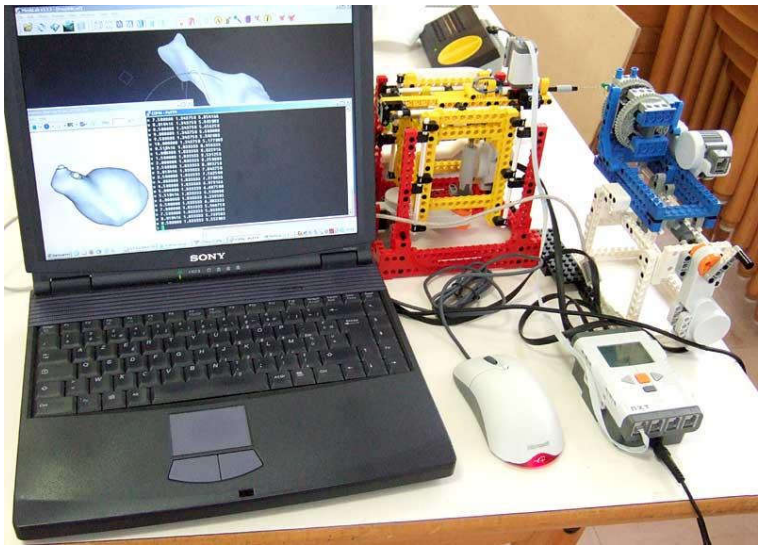


Figura 3.2. Componentes del sistema Lego NXT 3D.

3.2 Escáner de bajo costo empleando webcams

Es un escáner desarrollado por Narváez [13] en el 2010 como un Trabajo Especial de Grado. Es un escáner sin contacto y visión pasiva que consta de 2 webcams conectadas a un computador (ver Figura 3.3).

Figura 3.3. Componentes del escáner propuesto por Narváez.

En cuanto a su funcionamiento, el objeto a escanear debe colocarse en un ambiente controlado, diseñado específicamente para este escáner, y debe rotarse manualmente para capturar una cantidad determinada de imágenes, estas imágenes son procesadas y alineadas para generar una nube de puntos, la cual es procesada usando el algoritmo de triangulación de Delaunay que genera como resultado una malla, o modelo 3D (ver Figura 3.4).

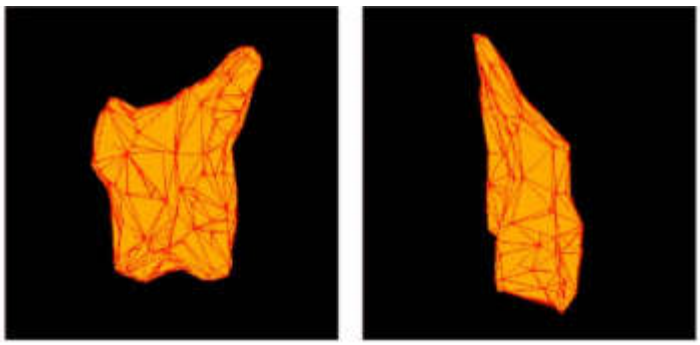


Figura 3.4. Muestra de modelo resultante del proceso de escaneo de Narváez.

3.3 Milkscanner

Catalogado como escáner sin contacto y de visión pasiva, este escáner [14] funciona sumergiendo el objeto en un líquido opaco, como leche o tinta, y capturando imágenes a medida que se va agregando líquido al envase (ver Figura 3.5).

Figura 3.5. Representación del proceso de escaneo de Milkscanner.

A medida que se va agregando líquido se van capturando imágenes de las diferentes capas o cortes del objeto, de estas imágenes se remueve el líquido y el resultado se alinea para generar los detalles de profundidad del objeto escaneado.

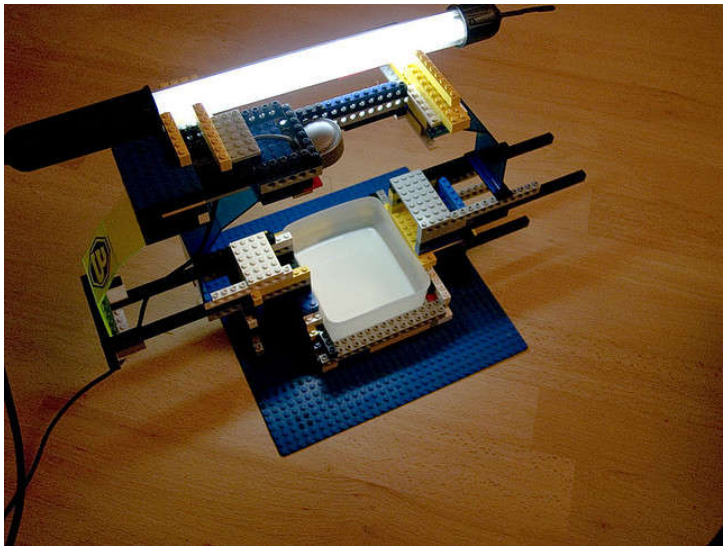


Figura 3.6. Plataforma de escaneo del Milkscanner.

Entre los problemas que posee este escáner es que no todos los objetos pueden ser sumergidos en líquido para ser escaneados debido a la capacidad de su plataforma (ver Figura 3.6), y su poca practicidad para escanear objetos relativamente grandes.

3.4 Proforma

ProbabilisticFeature-based On-line Rapid ModelAcquisition es un software para generación de modelos 3D a partir de imágenes (ver Figura 3.7) y algoritmos probabilísticos, sin la necesidad de láseres o un ambiente controlado para el escaneo. Fue diseñado [15] en 2009 en la Universidad de Cambridge y se categoriza como un escáner sin contacto de visión pasiva. Funciona usando una cámara y una computadora para procesar las imágenes.

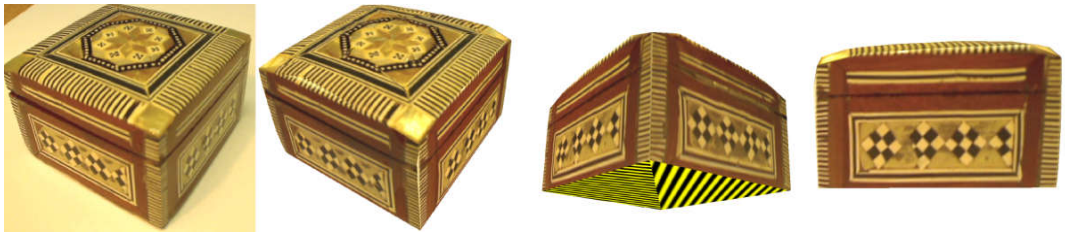


Figura 3.7. Ejemplo de escaneo de un cofre con Proforma.

El objeto debe ser rotado a mano para capturar sus diferentes caras que son procesadas en tiempo real mediante un conjunto de algoritmos probabilísticos, generando una nube de puntos, los triángulos que conforman al objeto y su texturizado, adaptando el modelo a medida que se gira el objeto (ver Figura 3.8).

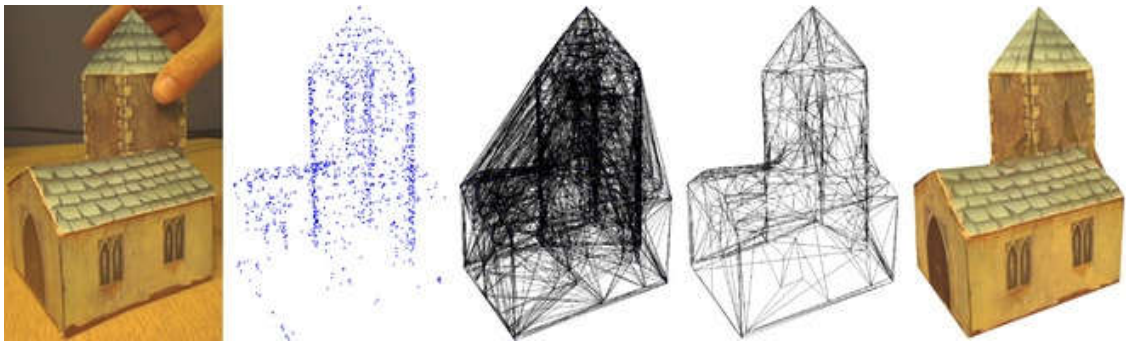


Figura 3.8. Escaneo de una maqueta de una iglesia con Proforma.

3.5 Makezine

Makezine [15] es un escáner de fabricación casera diseñado por Alessandro Grossi usando una cámara fotográfica DSLR, un apuntador láser y un Arduino para el control de la captura de imágenes (ver Figura 3.9) y Matlab para la generación del volumen.

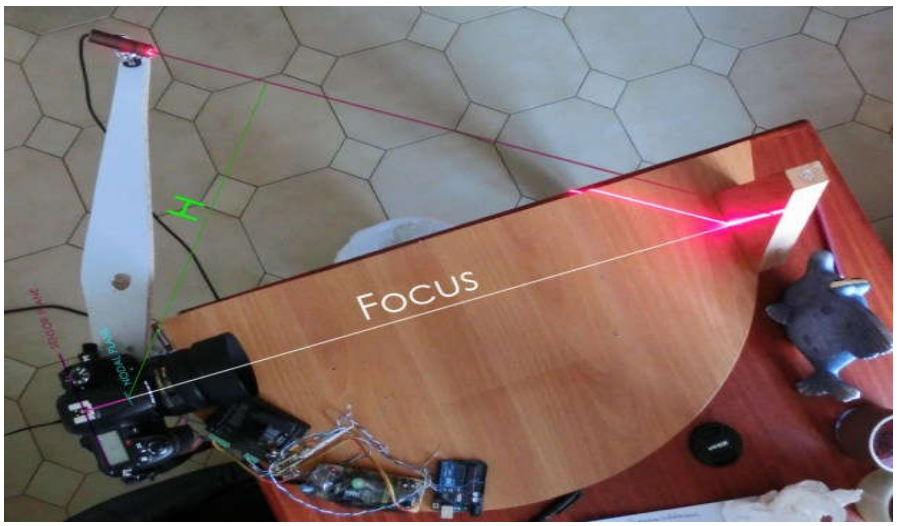


Figura 3.9. Sistema Makezine.

El láser marca el objeto y la cámara procede a capturar imágenes, a partir de estos datos y usando leyes de óptica y trigonometría se genera el eje Z en Matlab aprovechando las capacidades matemáticas de esta herramienta.

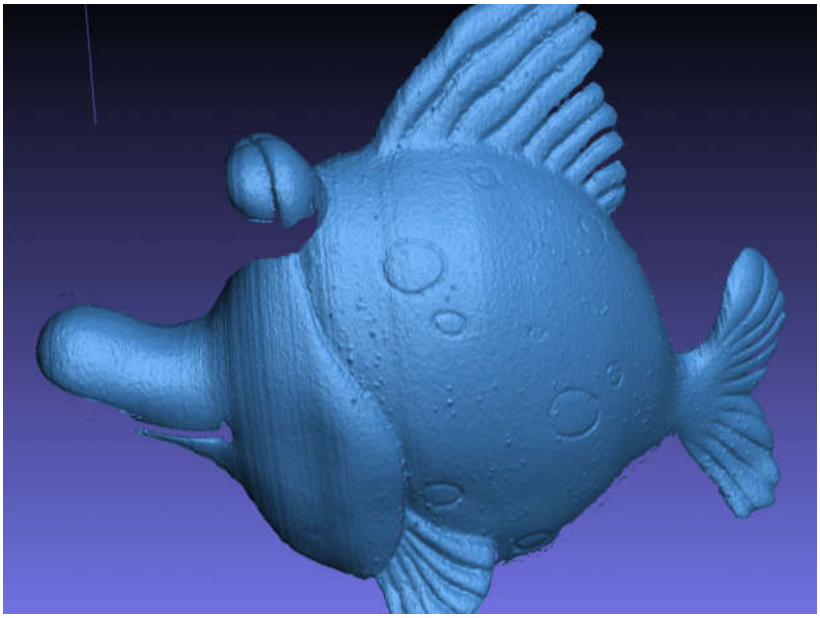


Figura 3.10. Ejemplo de resultado de escaneo con Makezine.

Los resultados dependen de la intensidad del láser y la precisión en la rotación del objeto (ver Figura 3.10).

3.6 Pi 3D Scanner Project

Este es otro ejemplo de escáner sin contacto de visión pasiva, el Pi 3D Scanner [17] fue desarrollado por Richard Garsthagen. Este escáner se basa en el uso de bloques Raspberry Pi y cámaras.

Figura 3.11. Pilares de escaneo de Pi 3D Scanner.

El sistema se compone de 13 pilares equipados que describen un círculo, cada uno con 3 Raspberry Pi con sus respectivas cámaras situadas en el tope, medio y base del pilar (ver Figura 3.11). Dada esta configuración la velocidad de este escáner es superior a los otros trabajos parecidos ya que los 39 Pi capturan todos los ángulos en el mismo instante de tiempo.

Figura 3.12. Comparación de modelo original y resultados de Pi 3D Scanner.

Los resultados del Pi 3D Scanner son muy precisos (ver Figura 3.12) dependiendo de la cantidad de pilares que se utilicen, según los resultados a partir de 4 pilares se obtienen buenos resultados. Recientemente fue utilizado para escanear un grupo de 200 personas con resultados muy fieles.

3.7 3D DigitizerPrototypewithCalibration

Este proyecto [18], desarrollado por Roger Muellerr, es el resultado de un proceso de evolución y experimentación. Originalmente fue un escáner de contacto. Pero en su última versión pasó a ser un escáner sin contacto y con visión activa.

- **Versión 1**: Esta versión cuenta con un cabezal que se encarga de hacer contacto con cada punto del objeto, esto hace que el tiempo de escaneo sea muy grande y limita el tamaño del objeto a escanear pero genera gran precisión en el resultado.
- **Versión 2**: Esta versión abandona el contacto y se basa en una cámara y un láser para su labor, permite escanear objetos más grandes y a mayor velocidad pero el movimiento del objeto es manual. Este escáner es muy simple y solo requiere calibrar la cámara.
- **Versión 3**: Es muy parecido a la versión 2 pero intenta hacer más automatizado el proceso de rotar el objeto y también de eliminar la luz ambiental, para esto se utilizó una

especie de caja negra con una base rotatoria, los resultados mejoran con respecto a la versión 2 pero limitando de nuevo el tamaño del objeto.

- **Versión 4**: La versión más reciente es una mejora de la versión 2, ésta se calibra usando un objeto específico, un método para detectar el láser de forma automática y elimina el uso de la caja negra de la versión 3, además que tiene un software para controlar el proceso de escaneo (ver Figura 3.13).

Figura 3.13. Sistema de Escaneo en su versión 4.

3.8 insight3d

Este software [19] se encarga de reconstruir un objeto a partir de imágenes que deben ser obtenidas por el usuario, básicamente el usuario es el escáner y el programa procesa el trabajo del usuario.

El programa se encarga de detectar automáticamente los parámetros de la cámara usando las fotos y luego busca la correspondencia entre las imágenes usando puntos relevantes de las fotos. El proceso puede tomar unos 10 minutos y por ahora no funciona perfectamente con todas las cámaras comerciales. A partir de las imágenes genera una nube de puntos que luego es texturizada usando las imágenes originales (ver Figura 3.14).

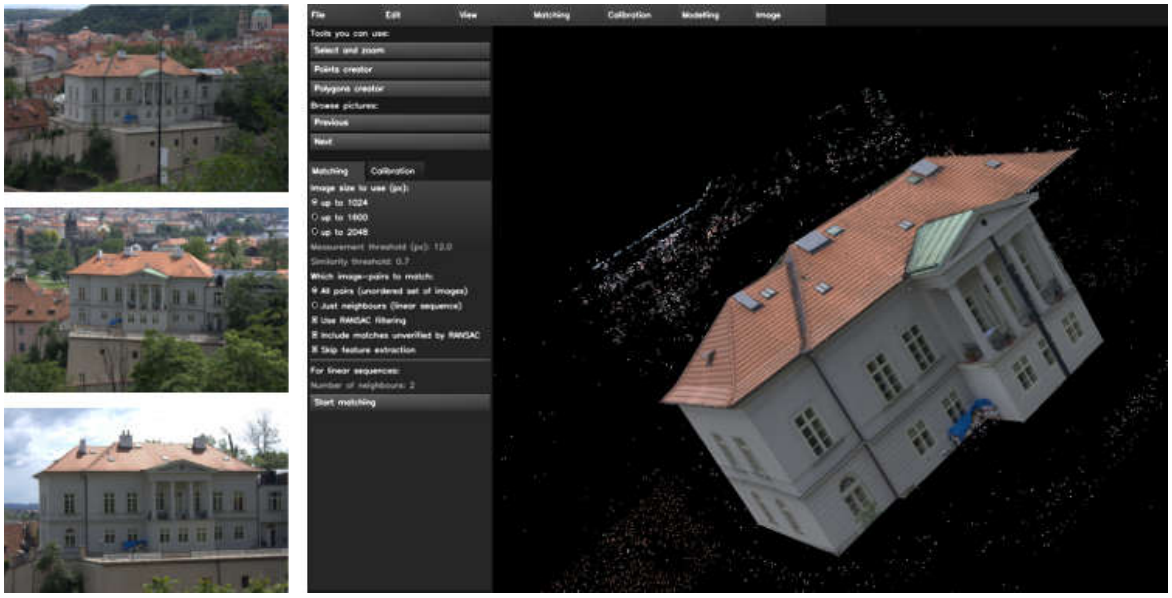


Figura 3.14. Modelo texturizado de una casa en base a fotografías a distancia.

Capítulo 4 - Diseño de la Solución

Se propone crear un sistema compuesto por 3 dispositivos con Sistema Operativo Android y un bloque Lego Mindstorms NXT, que se comunicarán entre sí mediante Bluetooth como se muestra en la Figura 4.1. Asi, están distribuidos de la siguiente manera, y con las siguientes funciones:

- **Robot (Lego Mindstorms NXT):** Desplazamiento circular alrededor del objeto a escanear.
- **Cámaras (2 Teléfonos Android):** Calibración, Captura y Procesamiento de imágenes
- **Controlador (Tableta Android):** Coordinar comunicación entre dispositivos, Establecimiento de parámetros de trabajo para proceso de escaneo, calibración estéreo, generación de nube de puntos y modelo 3D.

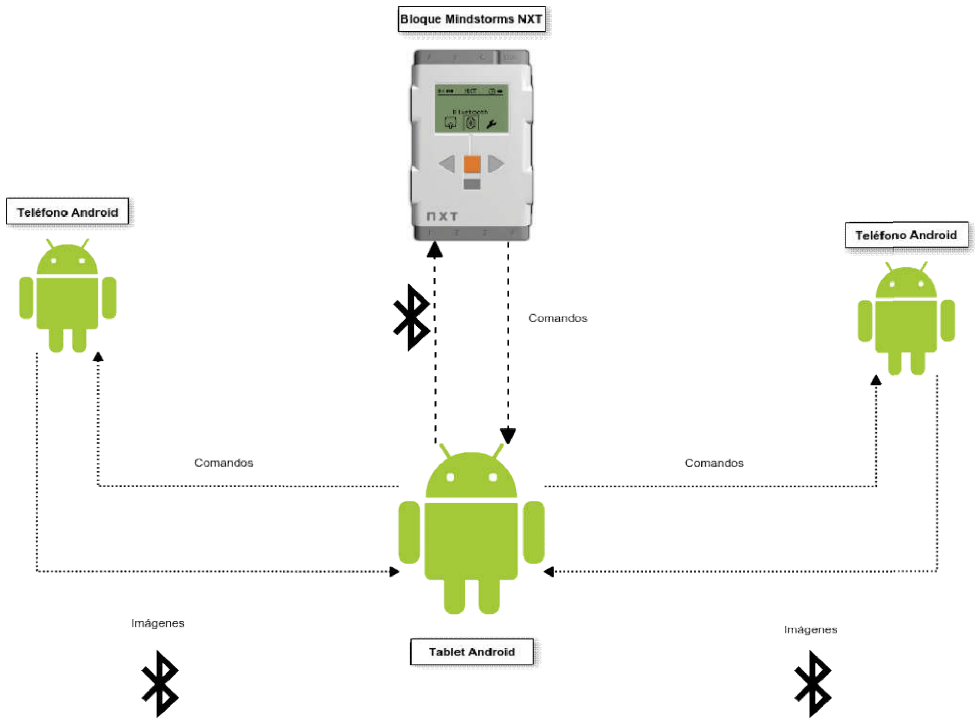


Figura 4.1. Esquema Propuesto.

Una vez que se ubique el objeto a escanear éste se deberá posicionar sobre una guía especial para el Robot (ver Figura 4.2), la misma debe ser desplegada en una superficie plana sobre la

que el Robot se pueda desplazar, a continuación se puede iniciar el proceso de escaneo, en el cual el Controlador coordinará el movimiento del Robot, haciendo que se desplace de forma circular alrededor del objeto a escanear.

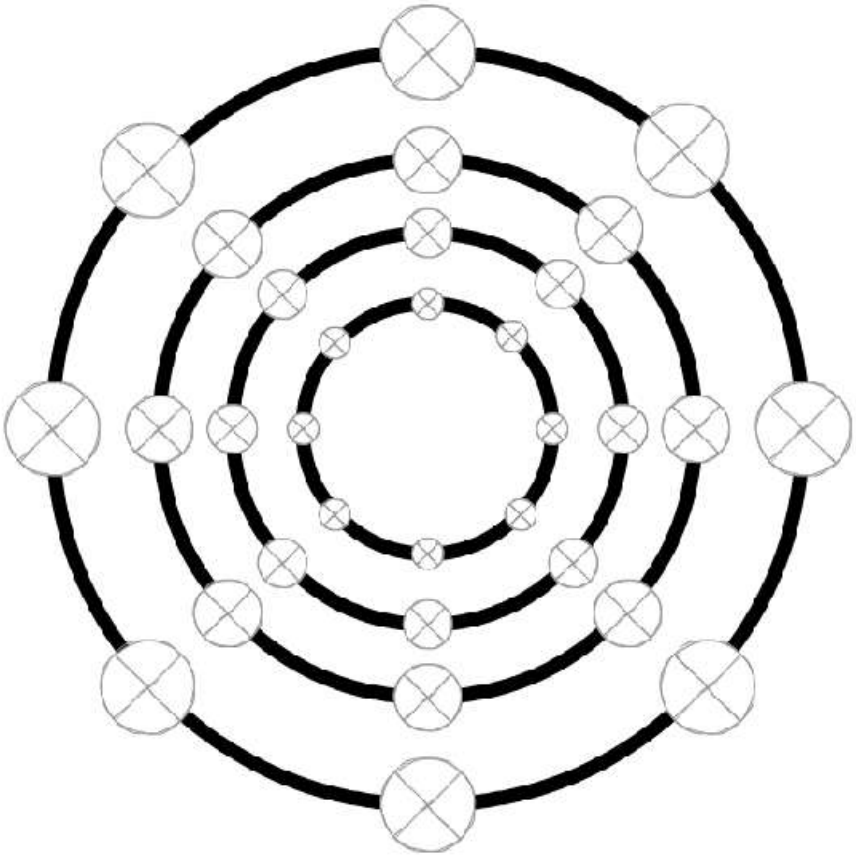


Figura 4.2. Prototipo de guía para el desplazamiento del Robot.

Cuando llegue a uno de los puntos de control marcados en la guía, notificará al Controlador, el cual ordenará a las Cámaras iniciar la captura de las imágenes. Una vez capturadas, se procederá a procesarlas, luego las Cámaras notificarán de nuevo al Controlador, enviando primero los puntos y luego la imagen procesada, la cual ordenará al Robot iniciar su movimiento hacia el siguiente punto de control.

Los pasos del proceso de escaneo se describen a mayor detalle en la sección 4.3 del documento.

4.1 Hardware

En esta sección se describen los componentes de Hardware necesarios para desarrollar esta propuesta.

Robot

El robot que se utilizará es el Mindstorm NXT, este bloque tiene 3 puertos de salida y 4 puertos de entrada. El NXT [20] cuenta con 2 microcontroladores (un ARM7 de 32 bits y un AVR de 8 bits), 256 KB de memoria flash y 64 KB de memoria RAM. para poder comunicarse cuenta con una antena Bluetooth.

El robot debe ser capaz de llevar ambas cámaras sujetadas firmemente sobre su base, así como tener la capacidad de desplazarse alrededor del objeto escaneado. A continuación se describen las distintas versiones del robot.

MADN3S MK.1

Este es el primer prototipo para el robot, donde se incorporan 2 sensores de luz, que utiliza para seguir la guía, y un sensor ultrasónico, para estimar la distancia con respecto al objeto a escanear (ver Figura 4.3).

Los sensores de luz están fijados al brazo, lo cual permite que el robot pueda ajustarse a la cuerda más cercana en la circunferencia que sigue dentro de la guía. Por encima de este brazo se encuentra la base para las cámaras. Para minimizar el número de partes móviles del robot, las cámaras siempre están orientadas hacia el frente.

De los tres motores de los que dispone el robot, se utilizan dos para el desplazamiento del mismo y el tercero para mover el brazo con los sensores. Este último tiene la limitante de que no puede hacer un recorrido de 360º cada sensor tiene un rango de 90º para efectuar las mediciones necesarias.

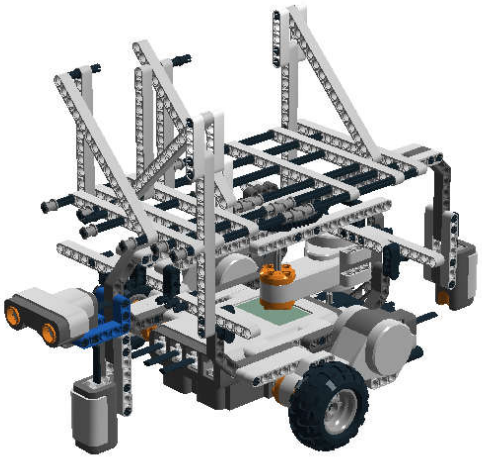


Figura 4.3. Prototipo robot Mindstorms, MADN3S MK.1.

MADN3S MK.2

Esta es una versión más simplificada del MK.1, se pasa de un esquema de vehículo convencional a un vehículo holonómico y se elimina el brazo móvil fusionándolo con el cuerpo del robot, como se ve en la figura 4.4. Los tres motores están colocados formando un triángulo y con la incorporación de las ruedas rotacaster[10] tiene la habilidad de moverse en cualquier direccion sin mayor problema. Como en el MK.1, ambas cámaras se encuentran en un receptáculo enfocando hacia el frente.

El primer sensor de luz se encuentra justo bajo el bloque en el centro de los 3 motores y el segundo se encuentra al final del brazo, el principio de este prototipo es buscar las 2 marcas que estén alineadas en la guia y situarse usando los sensores hasta que ambos detectan el color de las marcas. Sobre el bloque se encuentra un receptáculo para poder insertar las cámaras enfocando hacia el frente.

[10] Es una rueda multidireccional distribuida por HiTechnic

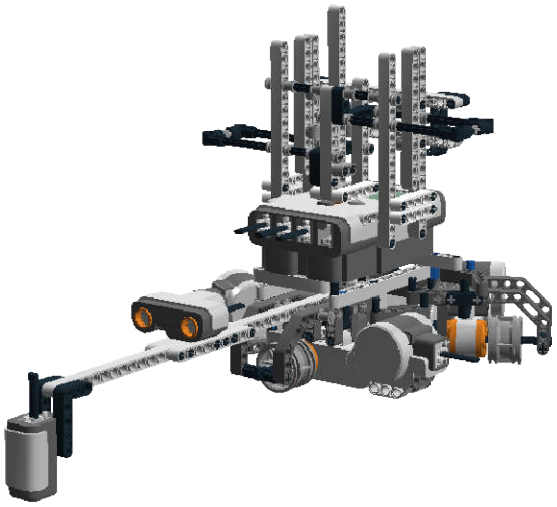


Figura 4.4. Prototipo robot Mindstorms, MADN3S MK.2.

MADN3S MK.3

Buscando un poco más de independencia del robot, se propone un tercer prototipo, ilustrado en la figura 4.5, que no necesita la guia ni los sensores de color. Este prototipo se basa en la clase `OmniPilot` que ofrece LeJOS. Esta clase tiene la habilidad de ejecutar movimientos en arco muy precisos partiendo de un robot holonómico, las dimensiones del círculo que se debe recorrer, de la distancia entre las ruedas del robot y el centro y finalmente de la ubicación de cada motor, esto para definir qué motor usar para rotar. El único requisito para utilizar esta clase es que las ruedas se encuentren distribuidas en el cuerpo del robot formando un triángulo equilátero.

La forma de este prototipo es muy parecida al MK.2 pero se eliminan los sensores de color y el brazo que sostenía el segundo sensor de color, esto permite que las baterías duren más ya que los sensores de color no consumen energía.

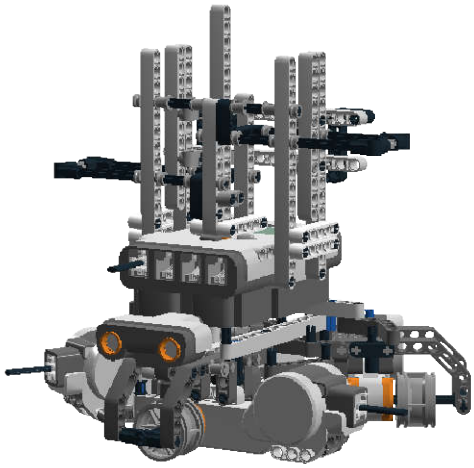


Figura 4.5.Prototipo robot Mindstorms, MADN3S MK.3.

Controlador

Para el controlador se utilizará una tableta AsusNexus 7 la cual dispone de un procesador Quad-core 1.5 GHz Krait, un GPU Adreno 320 y 2GB de Memoria RAM [21].

Cámaras

Las cámaras serán 2 Motorola Moto G 4G (1era Generación), que cuenta con un procesador Quad-core 1.2 GHz Cortex-A7, un GPU Adreno 305, 1 GB de Memoria RAM y una cámara de 5 MP [22].

4.2 Software

En esta sección se describen los componentes de software necesarios para desarrollar esta propuesta.

4.2.1 Robot

Una aplicación escrita en Java con leJOS que controla el movimiento del robot, para mantener al mismo a una distancia constante del objeto y moverse alrededor de éste trazando una circunferencia para que las cámaras puedan capturar las imágenes necesarias, asi como la comunicación Bluetooth con el Controlador.

4.2.2 Cámara

Cada cámara requiere una aplicación para Android, compatible con versiones entre 4.0 y 5.1, que se encargue de capturar imágenes bajo demanda, que las procese para eliminar todos los elementos que no sean relevantes ni sean parte del objeto a escanear, obtenga los puntos de interés y finalmente envíe los puntos e imágenes resultantes del proceso al controlador. Estas operaciones se harán usando la biblioteca OpenCV. La aplicación contará con una interfaz simple que refleje el estado en el que se encuentren en el proceso de escaneo (ver Figura 4.6).

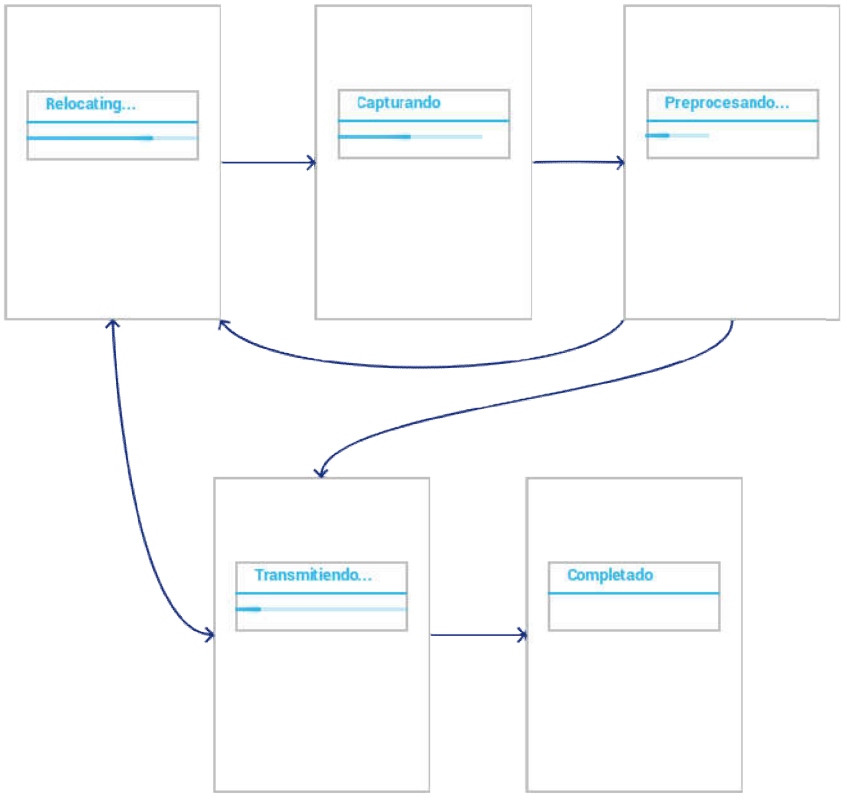


Figura 4.6. Vistas de la aplicación para las cámaras.

4.2.3 Controlador

La tableta que funciona como controlador, requiere una aplicación para Android compatible con versiones entre 4.0 y 5.1, que funcione como control de todo el escáner. Esta aplicación se encargará de controlar las acciones del Robot, así como también de ordenar a las cámaras iniciar el proceso de calibración, captura y transmisión de imágenes, una vez que el robot esté en posición y se hayan recorrido todos los puntos de control, debe procesar las imágenes capturadas para generar el objeto 3D que finalmente será desplegado en pantalla para ser manipulado por el usuario.

La aplicación contará con tres vistas: la principal que mostrará los dispositivos disponibles para conectarse, la segunda vista mostrará el estado de las conexiones y la última vista de la aplicación, presentará un diálogo bloqueante mientras se esté realizando el escaneo en el punto de control. Este último proporcionará las opciones de cancelar el proceso y la de poder iniciar el visualizador de modelos 3D (ver Figura 4.7).

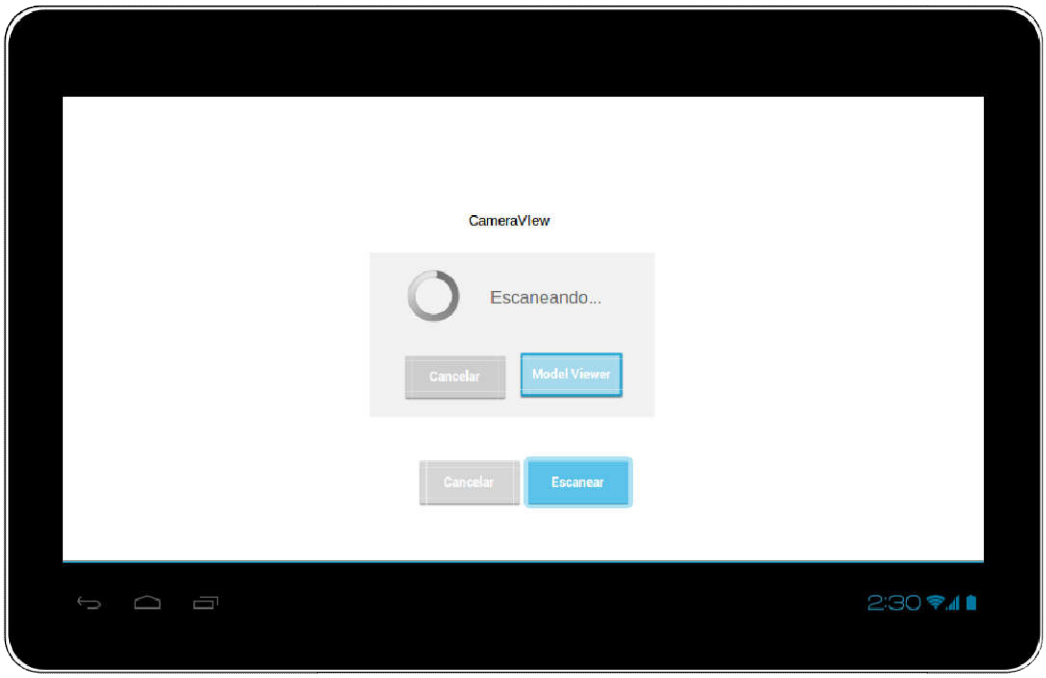


Figura 4.7. Vista alterna “Escaneando” aplicación Controlador Android.

4.3 Proceso de Escaneo

El proceso de escaneo consiste en 5 pasos:

1. **Conectar los dispositivos**: Una vez posicionados los teléfonos en la base del Robot, se inicia la aplicación de la Cámara en cada teléfono, asi como el Controlador en la tableta, desde la cual se verán los dispositivos Bluetooth disponibles separados en dos listas (NXT y Cámaras) según el tipo de dispositivo, es obligatorio seleccionar un dispositivo para el Robot y dos Cámaras, a las cuales se les designa su posición en una pantalla posterior, para continuar a la siguiente fase.
2. **Selección de la acción**: Una vez conectados todos los dispositivos se dispone de una vista con el estado de la conexión de cada dispositivo, así como los botones para elegir entre las acciones disponibles, las cuales son: Ver modelos ya escaneados y Escanear un nuevo objeto.

3. **Crear proyecto y calibrar las cámaras**: Si se selecciona la opción de escanear se verá la interfaz de proyecto en la cual se debe insertar un nombre y ejecutar la acción de calibrar. Al iniciarse este proceso se debe ubicar el patrón de calibración (ver Figura 4.8) frente a las cámaras a la distancia que estará el objeto a escanear para que las cámaras puedan detectar el patrón y que el usuario pueda seleccionar los puntos de calibración. El patrón es de 11 x 4 puntos, se deben seleccionar mínimo 2 pero mientras más se seleccionen más precisa será la calibración.

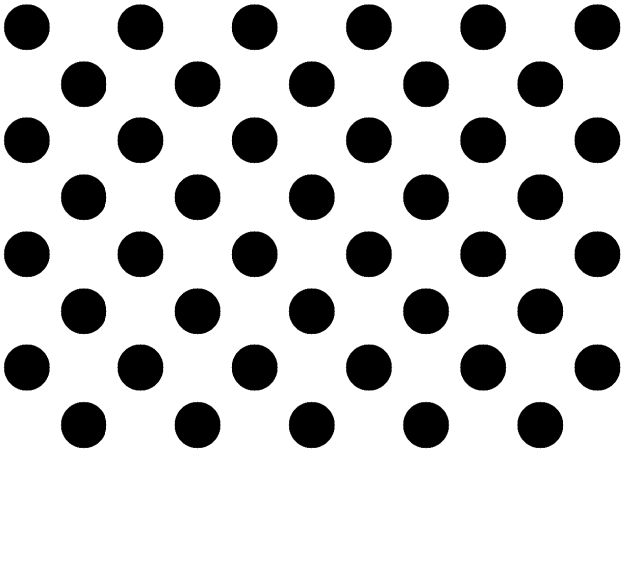


Figura 4.8. Patrón de calibración de OpenCV.

4. **Escanear el objeto**: Cuando el controlador notifique que el proceso de calibración terminó se procede a seleccionar la opción de escanear, a partir de este momento el sistema se encarga de capturar las imágenes y transmitirlas al controlador sin intervención del usuario.
5. **Generar el modelo**: Una vez terminado el proceso de escaneo el controlador habilitará la opción de generar el modelo, este proceso tomará el resultado del paso anterior y genera un modelo que será visualizado en el *KiwiViewer*[11]

4.4 Herramientas a Utilizar

Para el desarrollo de estas aplicaciones se utilizarán diferentes herramientas.

[11] http://www.kiwiviewer.org/

APIs, SDKs y frameworks:

- Android SDK
- Android NDK
- OpenCV
- VTK
- Json-cpp
- leJOS JVM (Java Virtual Machine) y NXJ API

Interfaz de Desarrollo:

- Android Studio 1.1.0

Sistemas de Manejo de Versiones:

- GIT

Capítulo 5 - Desarrollo de la solución

Tomando en consideración el diseño del sistema, descrito en el capítulo 4, a continuación se desarrolla el progreso de la implementación del sistema y las múltiples aplicaciones que lo componen a modo de iteraciones.
Para el desarrollo de este Trabajo Especial de Grado se utilizó una metodología ad-hoc basada ligeramente en Kanban[12], agrupando las tareas en iteraciones dependiendo de los objetivos definidos para las mismas.

5.1 Iteración 0 - Control y Desplazamiento del Robot

Esta iteración se realizó sobre la versión MK.1 del Robot, mencionada en la sección 4.1, dado que era una necesidad primordial que éste pudiera desplazarse correctamente alrededor del objeto para iniciar con las pruebas de escaneo.
La aplicación de Robot será un manejador de instrucciones, se conecta por Bluetooth al Controlador, de la cual recibirá configuraciones y comandos de movimiento al siguiente punto a escanear. Para establecer la conexión con el robot se utilizó la aplicación NXT Remote Control[13] desarrollada por JacekFedorynski. La cual provee una forma simple de levantar un túnel Bluetooth entre una aplicación Android y una aplicación en LeJOS.
El robot de la versión MK.1 seguirá la guía de color sobre la cual fue ubicado hasta llegar a los puntos de color rojo, usando el segundo sensor de color buscará trazar una cuerda de la circunferencia para apuntar las cámaras al objeto y notificará al Controlador que ya está ubicado en el lugar correcto para capturar la imagen. Para controlar el movimiento del Robot se utilizó la clase `Motor` de control de los Motores de la biblioteca LeJOS.

5.2 Iteración 1 - Optimización del movimiento del Robot

En esta iteración se trabajó nuevamente sobre el Robot dado que surgió la necesidad de modificar el diseño en la búsqueda de un desplazamiento correcto.

[12] Metodología ágil para la administración de procesos de desarrollo de software. http://kanbanblog.com/explained/
[13] https://github.com/jfedor2/nxt-remote-control

Robot:

El robot de la versión MK.1 demostró ser muy complejo al momento de alinearse con el objeto debido a la dificultad de alinearse con los puntos de control de la guía, dado que ambos sensores eran externos al cuerpo del Robot y estaban ubicados sobre un brazo que se ajustaba utilizando el tercer motor del Robot, como se puede observar en figura 4.3 de la sección 4.1 mencionada previamente.

En la versión MK.2 se redistribuyeron los sensores, colocando uno en el cuerpo del Robot y el segundo en un brazo fijo y hace su recorrido manteniendo las cámaras apuntando hacia el objeto, esto lo hace siguiendo la circunferencia sobre la que está ubicado el robot y la inmediatamente más interna (una para cada sensor) dibujadas en la guía, deteniéndose en los puntos de control para notificar al Controlador.

5.3 Iteración 2 - Comunicación, Captura de imágenes y Movimiento del robot

Ésta es la primera iteración sobre la que se trabajó en las tres aplicaciones en desarrollo. En esta iteración también se completó el desarrollo del Robot, consiguiendo un desplazamiento correcto alrededor del objeto a escanear, y es la última iteración donde está presente el mismo.

Robot:

El uso de una guía y sensores de color para mantener el Robot alineado con respecto al objeto a escanear resultó ser poco eficiente y propensa a errores. Por lo tanto, se ensambló la versión MK.3 del robot, descrita en la sección 4.1, que abandona el uso del patrón y calcula los puntos de control de acuerdo a la cantidad de paradas necesarias para escanear el objeto, valiéndose de que se traslada trazando una circunferencia, cuyo radio es proveído por el usuario.

Cámara:

La cámara tendrá tres responsabilidades principales:

1. **Manejar la calibración**: Ya que las cámaras se encuentran en posiciones diferentes, ambas deben ser sometidas a un proceso de calibración para aplicar correcciones a las mismas y así obtener un conjunto de parámetros que permiten hacer una correlación entre la información obtenida de las imágenes capturadas por cada cámara. Para lograr esto la aplicación se comunicará mediante `Intents` de Android con la aplicación "*OCV Camera Calibration*" de OpenCV, en la cual el usuario debe seleccionar los puntos del patrón en la pantalla y pulsar sobre el botón "Calibrar" para iniciar el proceso de

calibración interno de la aplicación. Una vez calculada la misma, los datos serán transmitidos de vuelta a la aplicación de Cámara para ser almacenada en el dispositivo y enviada al Controlador.

2. **Captura y procesamiento de imágenes**: La aplicación capturará las imágenes del objeto a escanear usando la cámara base de Android.
   Luego de capturarlas hará pasar por el siguiente pipeline de procesamiento:
   a. Aplicar `GrabCut`para aislar el objeto.
   b. Obtendrá una máscara usando Canny para detectar los bordes del objeto resultante del GrabCut.
   c. Con los resultados de los pasos previos aplicar `GoodFeaturesToTrack` de la biblioteca OpenCV a la imagen, el cual devuelve los puntos que definen el contorno del objeto escaneado en forma de coordenadas `(x,y)`.
   d. Enviar el arreglo de puntos y la imágen al Controlador.

   Este proceso se repite tantas veces como puntos de control se definan. Todas las operaciones de manipulación de imagen utilizadas en la cámara son de elaboración propia o contenidas en OpenCV.

3. **Comunicación con el Controlador**: Para esto se implementó un pequeño protocolo usando Bluetooth y documentos JSON, mediante el cual se transmiten tanto comandos como datos (documentos JSON, imágenes, entre otros). Un ejemplo de estos documentos es el siguiente:

```
{
    "command" : "scan" | "abort",
    "action" : "move" | "calibrate" | "wait" | "finish" | "config",
}
```

Controlador:

Las 3 tareas principales de esta aplicación son:

1. **Gestionar el proceso de escaneo**: Dado que esta aplicación funciona como el árbitro de todo el sistema, su tarea principal durante la fase de escaneo es comunicar a los dispositivos la tarea que debe ejecutar en un momento dado. Los mensajes que la aplicación puede enviar son los siguientes:
   a. Informar al Robot que debe moverse al siguiente punto de escaneo.

b. Una vez que el Robot se detenga, notificar a las Cámaras que deben capturar y procesar la imágen.
c. Notificar a la Cámara cuando se recibió el resultado o la imagen de un *frame.*

Para esto se implementó un pequeño protocolo usando Bluetooth y documentos JSON, dentro de los cuales se define la acción que debe ejecutar el dispositivo así como datos e imágenes.

2. **Procesar los resultados de la calibración de las Cámaras**: Una vez recibidos los resultados de la calibración de las Cámaras, los mismos pasan por el siguiente flujo para ser procesados:
    a. Los resultados son pasados como parámetros a la función `stereoCalibrate` de OpenCV, la cual genera las matrices de rotación (matriz R) y traslación (matriz T) junto a los coeficientes de distorsión de cada cámara en base a las matrices de calibración de las cámaras.
    b. Estos resultados deben ser luego procesados por `stereoRectify`, que devuelve las transformadas de rectificación (R1, R2) y matrices de proyección (P1, P2) para cada cámara.
    c. El resultado de `stereoRectify` es luego utilizado para llevar los puntos de ambas cámaras a un mismo sistema de coordenadas, un espacio unificado y rectificado, usando la función `initUndistortRectifyMap`, la cual devuelve los mapeos Map1 y Map2, que representan la calibración normalizada de ambas cámaras.
    d. Los mapeos Map1 y Map2 obtenidos de `initUndistortRectifyMap` se deben aplicar a los puntos procesados para llevarlos al sistema de coordenadas mencionado anteriormente.
3. **Procesar el resultado del escaneo**: Para esto se sigue el siguiente flujo para cada *frame*(ver Figura 5.1):
    a. Calcular los puntos relacionados entre cada par de imágenes, para esto se usa la función `calcOpticalFlowPyrLK`, este método calcula la correspondencia entre los puntos de dos las imágenes y retorna una lista de bytes donde la posición de la lista para un punto común está en `1` y los que no tienen correspondencia están en `0`.
    b. Una vez conseguidos los puntos comunes para un *frame*, se procede a buscar el valor Z para cada punto, este valor se obtiene al aplicar el método de triangulación explicado en la sección 2.5.3.

c. Luego se aplica una Descomposición por Valores Singulares del sistema de matrices $A = UDV^T$, ver definición en sección 2.5.4, el cual puede calcularse utilizando el método `SVDecomp` de OpenCV. En la última columna de la matriz V se encuentran los valores (x, y, z) del punto común.
d. Los valores obtenidos de resolver la matriz A mediante SVD en el paso anterior se ajustan a un sistema de coordenadas común usando la función nativa `IterativeClosestPoints` de VTK, en lenguaje C++, mediante el NDK de Android. Este proceso genera la nube de puntos del objeto escaneado.
e. Una vez obtenida la nube de puntos se procede a calcular los triángulos que unen esta nube de puntos, para esto se utiliza la función `vtkDelaunay3D` de VTK, también en lenguaje C++ y con la que el código en Java se comunica, de nuevo, mediante el NDK de Android.

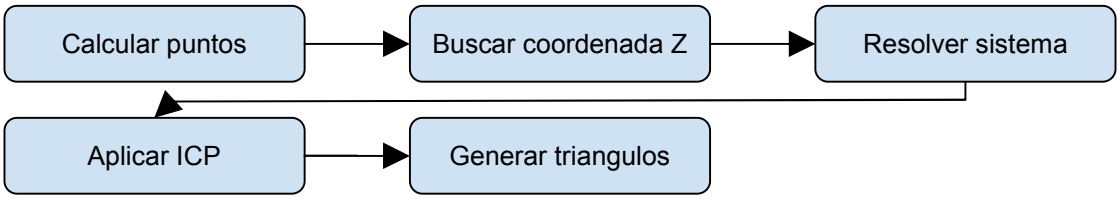


Figura 5.1. Diagrama de pasos para procesamiento del resultado del escaneo.

La función `IterativeClosestPointses` una implementación para VTK del algoritmo IterativeClosestPoints (ICP) el cual minimiza la diferencia entre dos nubes de puntos haciendo coincidir cada vértice en una imágen con el vértice más cercano en la siguiente y luego aplicar la transformación que modifique una superficie para hacerla coincidir con la otra. Generalmente se hacen varias iteraciones para conseguir mayor exactitud entre puntos.

Luego de hacer pruebas en el escenario, los resultados no correspondían con los objetos escaneados. Luego de hacer unas verificaciones de los datos, generando las imágenes originales con los puntos de interés encontrados por los algoritmos utilizados, se concluyó que la discrepancia entre datos y resultado era debido a que no se aplicaba la calibración a la información obtenida del escaneo.

5.4 Iteración 3 - Migración a cámara de OpenCV

Para solventar los problemas encontrados en la iteración 2, los cuales eran causados por la imposibilidad de aplicar los resultados a la cámara nativa de Android, usando la implementación de camara de OpenCV. Entonces, se procedió a modificar las aplicaciones de Cámara. El Robot y el Controlador no requirieron cambios.

En la aplicación de Cámara , como consecuencia de que la matriz de calibración no se aplicaba correctamente a las imágenes capturadas se migró de la cámara nativa de Android a la cámara que provee OpenCV, la cual permite aplicar las matrices de calibración sobre las mismas cámaras antes de la captura de imágenes. Como consecuencia se integró a la aplicación de la Cámara el pipeline de calibración presente en la aplicación externa de OpenCV, dedicada exclusivamente a esta tarea para simplificar este proceso, y se pudo aplicar el resultado de la calibración a la vista de la cámara directamente.

5.5 Iteración 4

Utilizar la cámara de OpenCV introdujo un fallo de memoria causado por la forma como se debe usar la misma, ya que cada cuadro capturado por la cámara (aprox. 30 cuadros por segundo) debía ser procesado por la aplicación y luego devuelto a la cámara para desplegarlo en pantalla. Esta fuga impide la culminación del proceso de escaneo, por lo cual se simplificó el sistema de escaneo, haciendo el proceso manual, con los siguientes cambios al sistema:

- Se descarta el desplazamiento automatizado del robot, de manera que el usuario tendría que rotar el objeto para capturar el siguiente ángulo
- La captura de las imágenes en la aplicación de Cámara se debe efectuar pulsando un botón en la interfaz
- La transferencia de las imágenes al Controlador se debe realizar a través de un computador.
- El procesamiento de las imágenes se trasladó completamente al Controlador.

Estas modificaciones se hicieron para demostrar la factibilidad de este proyecto usando la tecnología a disposición y así validar los resultados obtenidos.

Cámara:

Para poder trabajar con la fuga de memoria, se agregó un botón para tomar las fotos y se descartó el procesamiento de las imágenes en la Cámara, debiéndose proceder manualmente para hacer la transferencia de las imágenes al Controlador.

Controlador:

Las imágenes tomadas por las cámaras deben ser transferidas manualmente al dispositivo que ejecuta el Controlador, el cual hereda las tareas de procesamiento de imágenes que hacía la Cámara en la iteración 2 y mantiene sus tareas propias definidas anteriormente.

5.6 Iteración 5 - Solución de problemas de memoria y restauración del controlador

En esta iteración se solventó el problema de la fuga de memoria en la aplicación de la Cámara, volviendo al modo automatizado utilizado antes de la iteración 4, se devolvió la responsabilidad del desplazamiento al Robot, restaurando la transferencia de imágenes de la Cámara al Controlador mediante Bluetooth y se reinstauró parte del procesamiento de imágenes a la aplicación de Cámara, removiendo el botón de captura manual de su interfaz.

Cámara:

El procesar cada cuadro de video capturado por la cámara para desplegarlo en pantalla, implicaba un costo de procesamiento de al menos una rotación y una transformación matricial, ocupando 500KB en memoria para la operación en el modo más simple, lo cual incrementa considerablemente tanto para el modo de captura, como el de calibración. Esto fue la causa de la falla de memoria encontrado en la iteración anterior.

Con la implementación de un sistema de semáforos y utilizando una sola instancia estática de cada tipo de *renderer*, se redujo el impacto de la falla de memoria. No obstante, el problema de la fuga es inherente a la forma en que OpenCV procesa los cuadros de video y no pudo ser mitigado del todo, ya que requeriría modificar la biblioteca, sobrepasando el alcance de este trabajo.

Controlador:

Se deshabilitó el modo manual y el Controlador pasó a recibir las imágenes vía Bluetooth, manejando el proceso con la emisión y recepción de comandos, dirigiendo el funcionamiento de todo el sistema nuevamente.

Una vez implementados todos los cambios descritos anteriormente se completa el *pipeline* como se describe en la siguiente sección.

5.7 Pipeline final

Una vez resuelto el problema de la fuga de memoria en las cámaras, quedó completado el *pipeline* de escaneo planificado para este Trabajo Especial de Grado. Luego de todas las iteraciones, el *pipeline* final se describe a continuación:

Para la fase de conexión e inicialización se tienen los siguientes pasos (ver Figura 5.2):

Figura 5.2. Diagrama de pasos de conexión e inicialización.

En el Robot:

1. Encender el robot e iniciar aplicación de Robot

En la Cámara:

1. Activar Bluetooth y asegurarse que el dispositivo se encuentre visible
2. Ejecutar aplicación de Cámara

En el Controlador:

1. Iniciar la aplicación, escanear buscando dispositivos Bluetooth y seleccionar los que funcionarán como robot y Cámaras respectivamente (ver Figura 5.3).

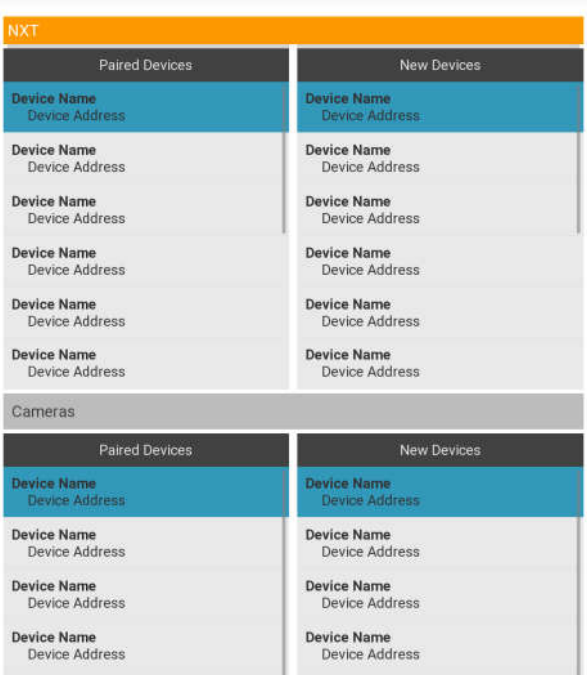


Figura 5.3. Interfaz de selección de dispositivos.

2. Especificar en una pantalla secundaria, de qué lado está cada Cámara, izquierda o derecha, para que el Controlador las trate correctamente (ver Figura 5.4).

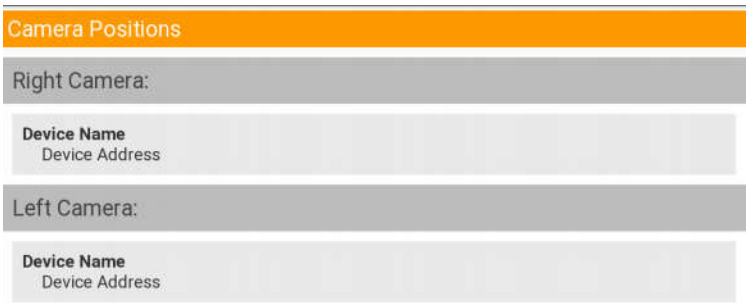


Figura 5.4. Diálogo para indicar correspondencia de Cámaras.

Al completar el proceso de conexión e inicialización, la aplicación pasa a la siguiente pantalla donde se puede observar el estado de la conexión de cada dispositivo. Al estar todos conectados exitosamente se procede a iniciar el proceso de configuración y calibración (ver Figura 5.5).

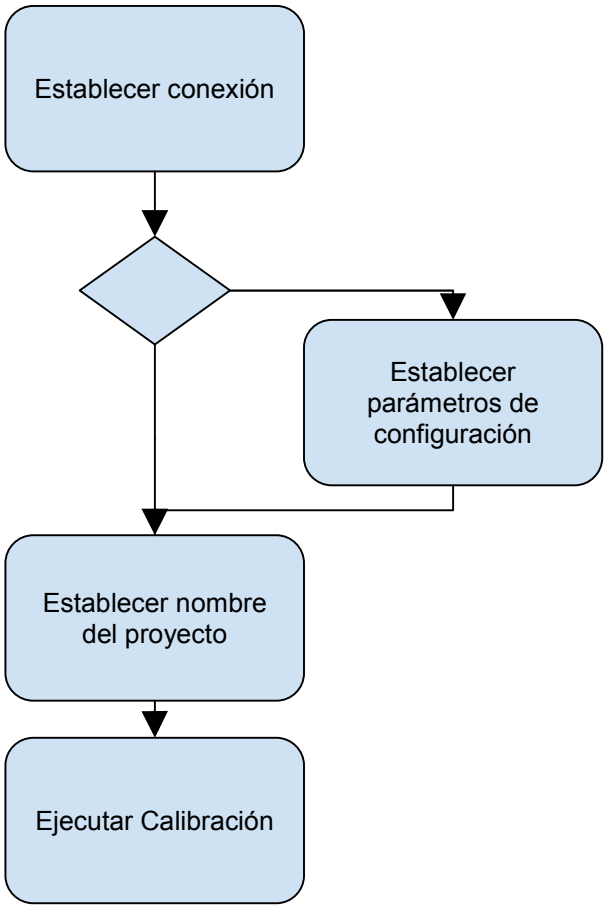


Figura 5.5. Diagrama de configuración y calibración.

En el Robot:

1. Se establece la conexión con el Controlador y se queda a la espera de la orden de movimiento o de finalización.

En la Cámara:

1. Se establece la conexión con el Controlador y se queda a la espera de la orden de captura y procesamiento de imágenes.

En el Controlador:

1. (Opcional) Se establecen parámetros de configuración, como tamaño de rectángulo de GrabCut, elección de algoritmo de detección de bordes entre otros. Este paso no es necesario ejecutarlo cada vez que se escanee. Existe una configuración por defecto, al igual que, cuando el usuario establece la configuración, ésta se persiste en la memoria del dispositivo (ver Figura 5.6).

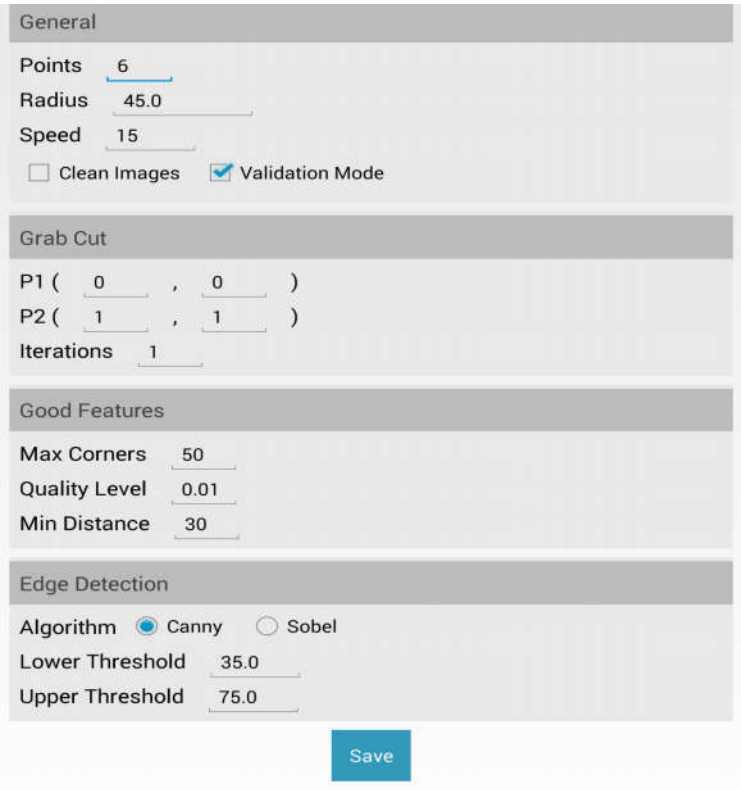


Figura 5.6. Interfaz de configuración.

2. Establecer nombre de Proyecto.
3. Iniciar proceso de Calibración (ver Figura 5.7).

Figura 5.7. Proceso de Calibración iniciado en Controlador.

En la Cámara:

1. La aplicación recibirá la orden de iniciar el proceso de Calibración desde el Controlador y pasará a dicho modo.
2. En este momento el usuario colocará el patrón de calibración frente a la cámara, la aplicación al reconocer dicho patrón dibujará sobre la pantalla los puntos. El usuario debe marcarlos en la pantalla. El patrón tiene 44 puntos en total, con un mínimo aceptable de 2, según está definido por la implementación de OpenCV. Mientras un mayor número de puntos indique el usuario más precisa será la calibración.
3. Cuando el usuario termine de seleccionar los puntos del patrón debe pulsar sobre el botón “Calibrar”. Esto iniciará el proceso de calibración de la Cámara, utilizando una versión modificada de la clase `CameraCalibrator` de la aplicación “*OCV Camera Calibration”* de OpenCV explicada anteriormente.
4. Al finalizar la calibración en la Cámara, ésta transmitirá el resultado al Controlador.

En el Controlador:

1. Al recibir el resultado de la calibración de ambas cámaras el Controlador procesa los resultados de la calibración de las Cámaras, los mismos pasan por el siguiente flujo para ser procesados:
    - Son parámetros de la función `stereoCalibrate` de OpenCV, la cual genera las matrices de rotación (matriz R) y traslación (matriz T) junto a los coeficientes de distorsión de cada cámara en base a las matrices de calibración de las cámaras, estas matrices y coeficientes son la información de las modificaciones que se deben aplicar a las imágenes capturadas por las cámaras para corregir las distorsiones presentes en ambas cámaras.
    - Los resultados del paso previo son luego procesados por `stereoRectify`, que devuelve las transformadas de rectificación (`R1, R2`) y matrices de proyección (`P1, P2`) para cada cámara.
2. Las matrices de Rotación (`R`), Traslación (`T`) son transmitidas de nuevo a cada cámara, junto a su respectiva transformada de rectificación (`R1, R2`) y matriz de proyección (`P1, P2`), 1 siendo la izquierda y 2 la derecha, pero mantenidas como números en esta fase del pipeline por convención.
3. Posterior a la transmisión de las matrices de Calibración de vuelta a las Cámaras, en el UI se habilita el botón de “Escanear”.

Al completar el proceso de configuración y calibración, una vez el usuario pulse sobre el botón de escanear se inicia el proceso de Escaneo (ver Figura 5.8). Esta fase se repetirá tantas veces como paradas hayan sido especificadas en la configuración del Controlador, mencionada previamente.

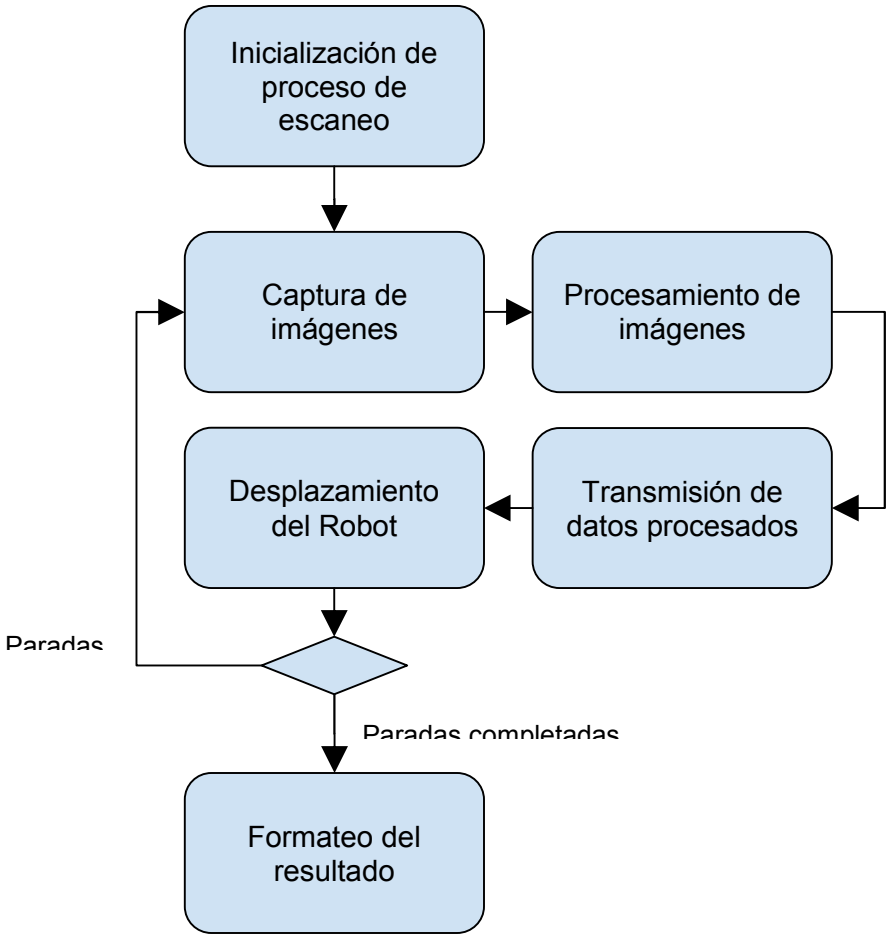


Figura 5.8. Diagrama de pasos de escaneo.

En el Controlador:

1. Se transmite la orden de capturar imágenes a las cámaras y se queda a la espera de la recepción del resultado de la captura de las mismas.

En la Cámara:

1. Se inicia proceso de captura y procesamiento activando banderas que hacen el cambio del *renderer* de la Cámara al *renderer* de previsualización, al que aplica el resultado de la calibración.
2. Se captura la imagen (ver Figura 5.9).

Figura 5.9. Imagen capturada por la cámara.

3. Se le aplica `GrabCut`para aislar el objeto (ver Figura 5.10)

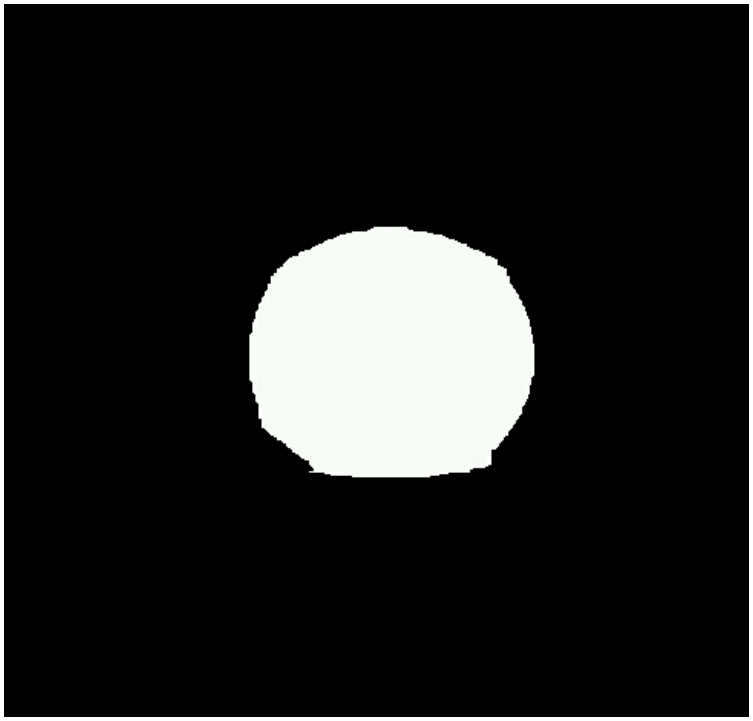


Figura 5.10. Máscara generada por el algoritmo GrabCut.

4. Utilizando el resultado del paso previo se saca una máscara usando el algoritmo de Canny para detectar los bordes del objeto resultante del GrabCut (ver Figura 5.11)

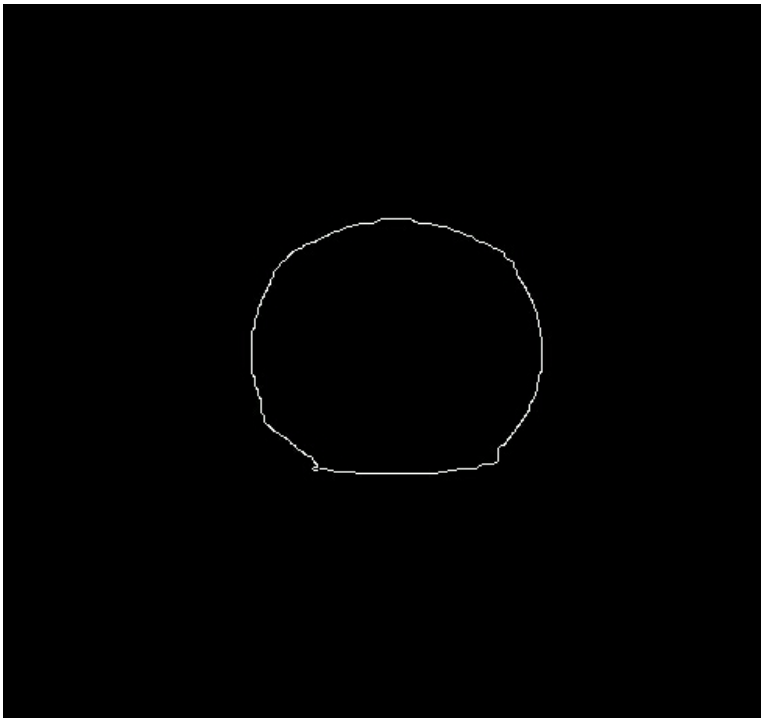


Figura 5.11. Bordes detectados por el algoritmo de Canny a partir de la máscara obtenida.

5. Con los resultados de los pasos previos aplicar `GoodFeaturesToTrack` a la imágen, el cual devuelve los puntos relevantes del objeto escaneado en forma de coordenadas `(x,y)` (ver Figura 5.12)

Figura 5.12. Puntos de interés designados por el algoritmo GoodFeaturesToTrack.

6. Ya culminado el procesamiento se transmite el documento JSON con los resultados y la imágen al Controlador.

En el Controlador:

1. Al recibirse el documento JSON con el resultado de la captura se le envía una señal a la Cámara confirmando recepción, el cual también le indica a la Cámara que ahora debe enviar la imágen asociada a los datos enviados.
2. Una vez recibidos tanto el documento JSON como la imágen de cada Cámara, se persiste toda la información en el almacenamiento interno del dispositivo y se envía al robot el comando de movimiento

En el Robot:

1. Se inicia movimiento y se notifica al Controlador cuando esté completado el mismo.
2. Se pasa de nuevo a la espera de una nueva señal de movimiento o finalización.

En el Controlador:

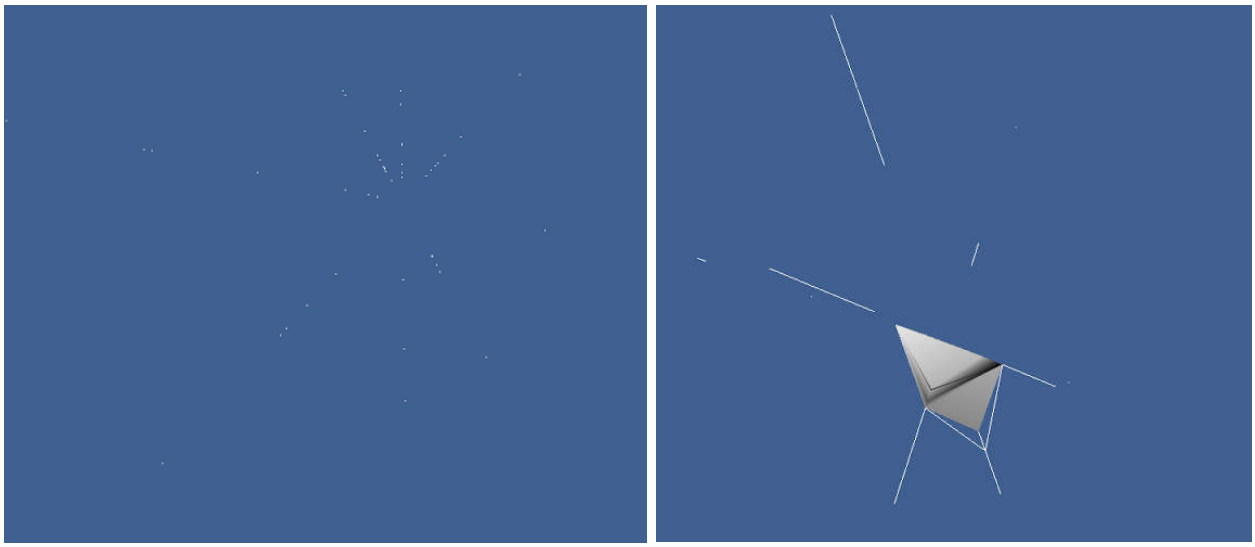
Una vez todas las paradas estén completas el Controlador pasará al procesamiento de todos los datos para generar el modelo 3D.

1. Se usa el algoritmo de Lucas-Kanade función `calcOpticalFlowPyrLK` de OpenCV para conseguir la relación entre los puntos de las imágenes de cada *frame* e ir descartando aquellos puntos no comunes del objeto escaneado.
2. Una vez filtrados los puntos comunes se procede a generar la nube de puntos a partir de los mismos, para esto se usa el algoritmo IterativeClosestPoints, función `vtkIterativeClosestPointTransform` de VTK.
3. Luego de obtenida la nube de puntos se procede a generar los triángulos que la unen usando el algoritmo de Delaunay (`vtkDelaunay3D` en VTK).
4. Obtenidos los triángulos que componen el modelo 3D se exporta el resultado a un archivo `.vtp`, para ser visualizado en programas de visualización como *KiwiViewer.*

Estos últimos tres pasos, las llamadas a `vtkIterativeClosestPointTransform`, `vtkDelaunay3D` y generar un archivo con extensión `.vtp`, se ejecutan en lenguaje C++ con el cual la aplicación Android se comunica mediante herramientas incluidas en el NDK de Android, como el protocolo JNI (Java Native Interface).

5.8 Problemas en el Pipeline

Una vez completado el pipeline descrito anteriormente, al hacer pruebas escaneando diferentes objetos, tanto simples (como pelotas de beisbol) y complejos (como figuras de acción) se observó que los resultados generados no correspondian con el objeto escaneado para confirmar este error, a estos resultados se les aplicó el algoritmo de Delaunay, que es el siguiente paso del pipeline, y efectivamente el objeto generado no tenía ningún parecido con el



objeto escaneado, en el caso de la figura, un balón de baloncesto (ver Figura 5.13).

Figura 5.13. Nube de Puntos procesada (Izq) y con Delaunay (Der).

Para tratar este problema, se cambió la forma en que se calculaba la descomposición SVD de la matriz A. Inicialmente ésta se obtenía utilizando funciones de multiplicación de matrices de OpenCV, que reciben directamente las matrices a multiplicar como parámetros y devuelven el resultado en una nueva matriz**.** En la nueva implementación se pasó a calcular el resultado utilizando multiplicación y resta de filas y columnas, basándose en la implementación de Narváez [13], pero el resultado no varió. Dado este escenario, se sospecha que el error en el cálculo de la nube de puntos puede ser debido al tamaño del epsilon[14] del hardware utilizado, lo cual hacía que, al calcular la descomposición SVD mencionada anteriormente, el vector de donde se extraían las coordenadas de los puntos mostraban una tendencia a estacionarse sobre los ejes de coordenadas, debido al ínfimo valor de las mismas.

[14] En computación es el mínimo valor numérico representable por un computador

5.9 Iteración 6 - Ajuste de Alcance

Luego de todas estas iteraciones el resultado seguía siendo incorrecto a pesar de seguir el conjunto de instrucciones planteado en [6]. Luego de muchas pruebas se llegó a la conclusión de que la razón era la baja capacidad de procesamiento del hardware, muy diferente a la de una PC. Dado este escenario se propuso cambiar el objetivo, eliminar la generación del volumen y generar un resultado basado en el escaneo y segmentación del objeto.

Para esto se genero un archivo que contiene la información de los puntos de cada *frame*junto con la ruta de la imagen asociada y en la imagen se marcaron los puntos detectados por el proceso de segmentación para ser utilizados como un método de validación del mismo. Estos archivos quedan a disponibilidad del usuario para poder evaluarlos.

En el siguiente capítulo se muestran los resultados obtenidos en el proceso de captura y segmentación de los objetos a modo de validación del ajuste de objetivo planteado

Capítulo 6 - Pruebas y Resultados

En este capítulo se analizarán los resultados obtenidos al hacer diferentes pruebas realizadas para este Trabajo Especial de Grado. Se utilizó una variedad de objetos que podrían calificarse como simples y complejos, tanto en forma como en color.

El ambiente de pruebas está compuesto por un espacio acondicionado con una pantalla verde, para ayudar con el descarte de ruido de fondo. Asi como de 3 luces blancas de fotografía para iluminar la escena y el objeto escaneado.

Estas pruebas fueron realizadas con el fin de validar que el proceso de segmentación funcionase de manera adecuada, basándose en las condiciones del ambiente de trabajo y en el nuevo alcance del T.E.G.

Prueba #1. Pelota de Béisbol

El primer objeto presentado es una pelota de béisbol. La misma puede considerarse un objeto simple, debido a que presenta una forma consistente, sin picos ni cavidades; así como también es de principalmente un mismo color con acentos en rojo. Este escaneo se hizo en 6 paradas, o deteniéndose cada 60º de la circunferencia trazada por el robot en su desplazamiento.

A continuación se presentan los resultados del escaneo. Las imágenes se presentan como la vista lado a lado de lo que capturó cada dispositivo (izquierda-derecha)

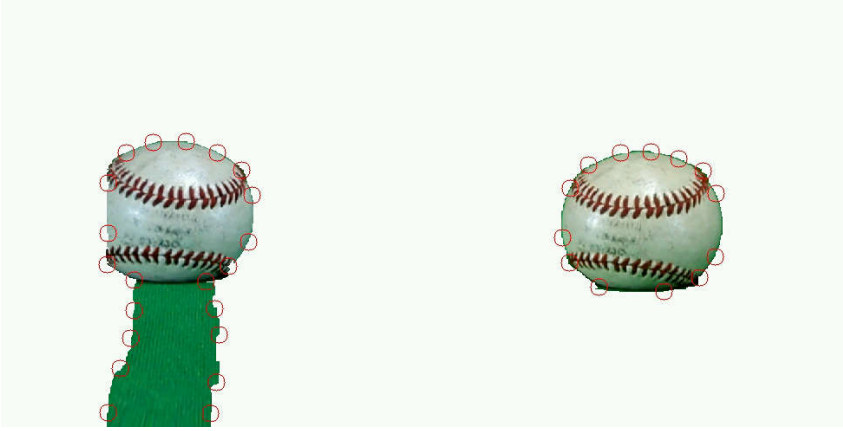


Figura 6.1. Puntos de interés en parada 0.

Figura 6.2. Puntos de interés en parada 1.

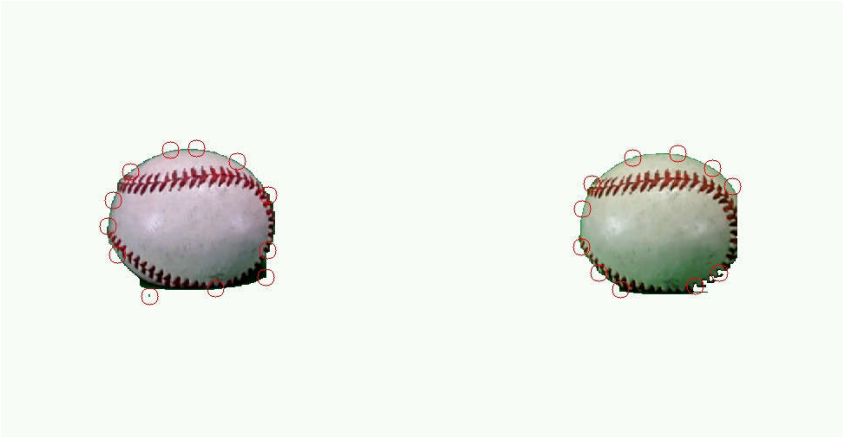


Figura 6.3. Puntos de interés en parada 2.

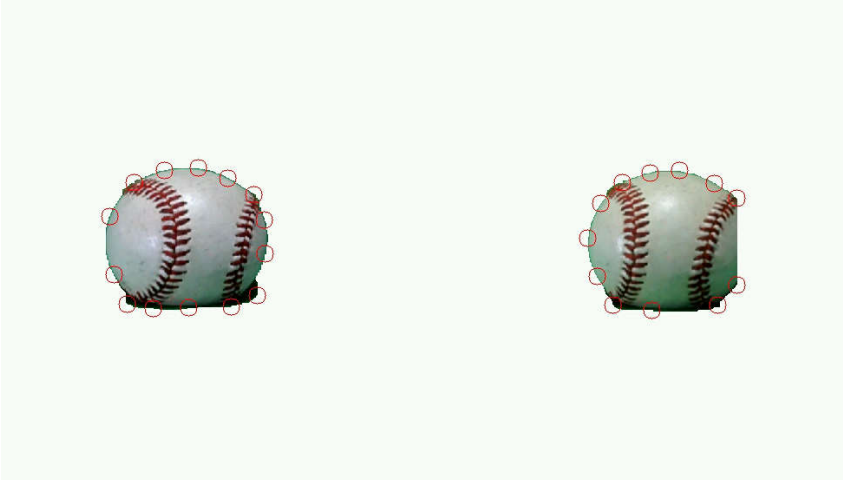


Figura 6.4. Puntos de interés en parada 3.

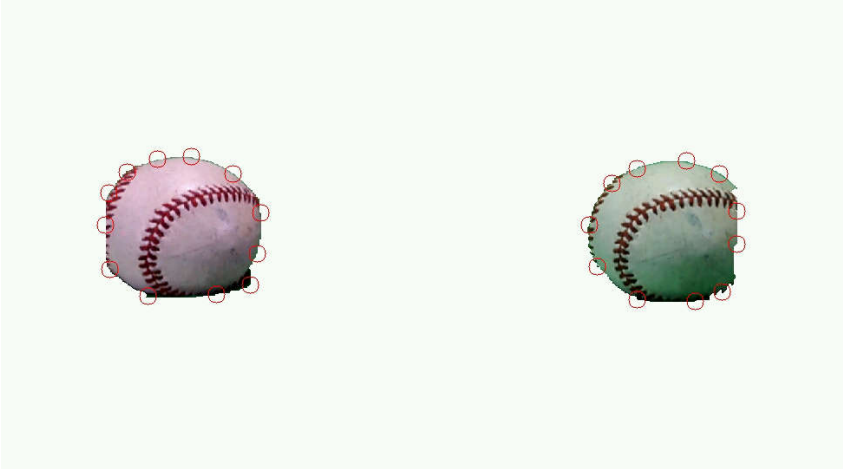


Figura 6.5. Puntos de interés en parada 4.

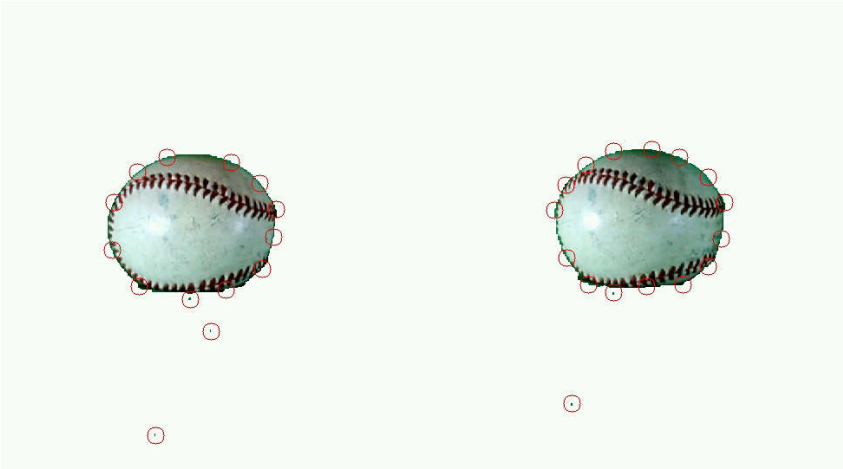


Figura 6.6. Puntos de interés en parada 5.

Prueba #2. Juguete de DarthMaul

El segundo objeto presentado es un juguete de StarWars del personaje DarthMaul. Este puede considerarse un objeto con un alto grado de complejidad, dado que tanto en forma y color es inconsistente. A diferencia de la pelota de béisbol tiene brazos, piernas, cuernos e incluso una espada. Igualmente en cuanto a color, ya que es mayormente negro con marcas rojas en su cara y cuernos de color hueso. Este escaneo se hizo en 6 paradas. O deteniéndose cada 60º de la circunferencia trazada por el robot en su desplazamiento.

A continuación los resultados del escaneo. Las imágenes se presentan como la vista lado a lado de lo que capturó cada dispositivo (izquierda-derecha)

Figura 6.7. Puntos de interés en parada 0.

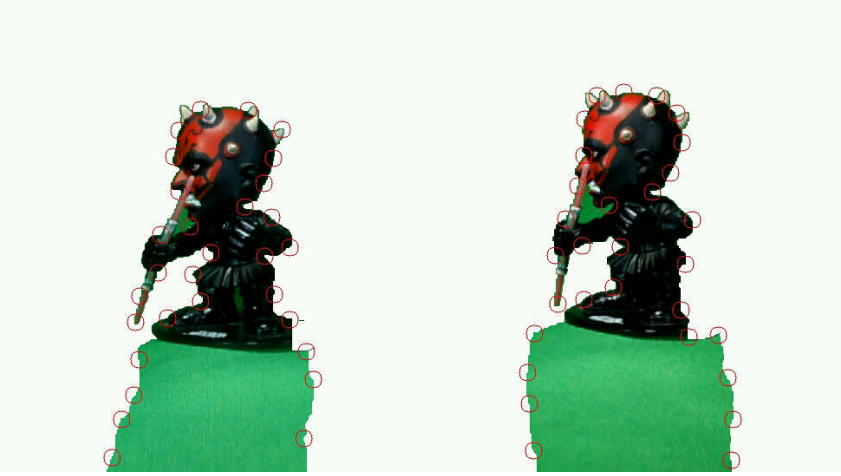


Figura 6.8. Puntos de interés en parada 1.

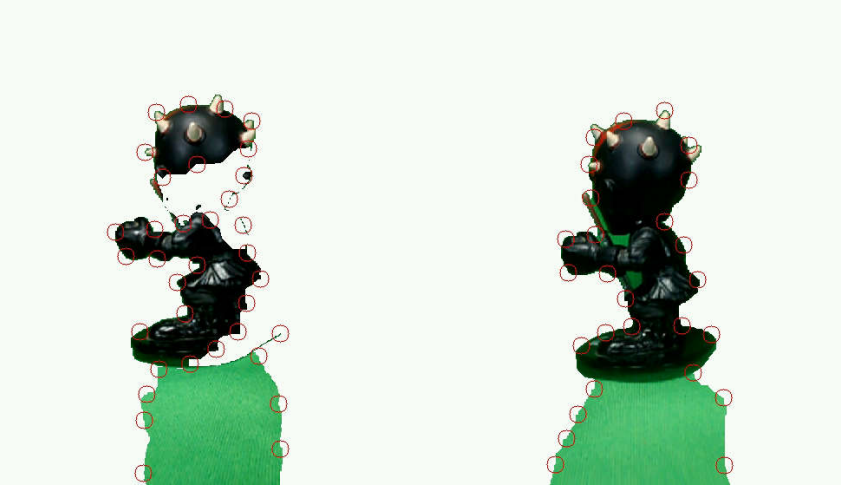


Figura 6.9. Puntos de interés en parada 2.

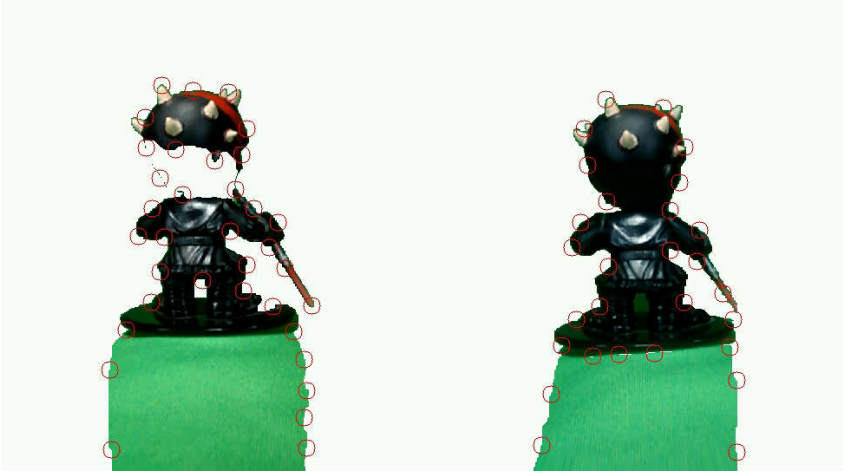


Figura 6.10. Puntos de interés en parada 3.

Figura 6.11. Puntos de interés en parada 4.

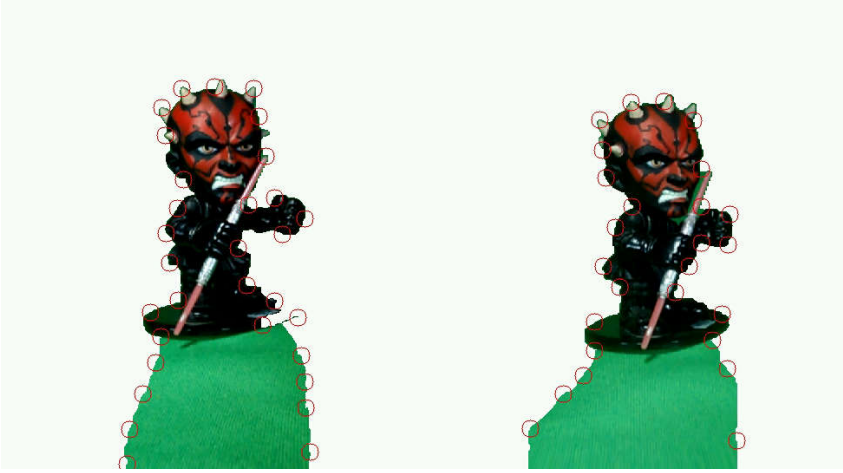


Figura 6.12. Puntos de interés en parada 5.

Prueba #3. Casco de Master Chief

Finalmente se presenta un casco de Master Chief del juego Halo 3. La principal razón de usar este objeto es la cantidad de detalles, el tamaño y la forma en que refleja la luz. Este escaneo se hizo en 6 paradas. O deteniéndose cada 60º de la circunferencia trazada por el robot en su desplazamiento.

A continuación los resultados del escaneo. Las imágenes se presentan como la vista lado a lado de lo que capturó cada dispositivo (izquierda-derecha)

Figura 6.13. Puntos de interés en parada 0.

Figura 6.14. Puntos de interés en parada 1.

Figura 6.15. Puntos de interés en parada 2.

Figura 6.16. Puntos de interés en parada 3.

Figura 6.17. Puntos de interés en parada 4.

Figura 6.18. Puntos de interés en parada 5.

Análisis

En las pruebas realizadas se observa que los objetos son aislados casi de manera perfecta. El error que se observa es debido al proceso de automatización, el cual elimina 2 factores importantes para el algoritmo de GrabCut:

- **Rectángulo de selección**: GrabCut requiere un rectángulo que delimite el área general donde se encuentra el objeto, en estos ejemplos se usó un rectángulo fijo que abarca el centro de la imagen que puede ser modificado en la configuración del Controlador, pero no en tiempo real. Para ver un ejemplo del resultado de procesar una imagen utilizando solamente el rectángulo de selección. Ver Figura 6.19.

Figura 6.19. Ejemplo de referencia de grabCut especificando rectángulo solamente.

- **Máscara manual:** En el proceso regular, GrabCut requiere que el usuario señale partes o el área general del objeto para poder filtrarlo, estas secciones el algoritmo las denota

como “primer plano seguro”, así como secciones de la imágen que no son parte del mismo, denotado como “fondo seguro”. Ver Figura 6.20 para un ejemplo de procesamiento especificando primer plano y fondo seguro.

Figura 6.20. Ejemplo de referencia de grabCut marcando zonas de fondo seguro y primer plano seguro (Izquierda) y resultado (Derecha).

Conclusiones y Trabajos Futuros

En este trabajo se implementó un Escáner Móvil basado en el Trabajo Especial de Grado presentado en [12]. Sin embargo, dadas las limitaciones expuestas en el Capítulo 5 fue necesario replantear el objetivo original, esto debido a limitaciones y aspectos técnicos por las diferentes tecnologías utilizadas en ambos trabajos. El más notable viene dado por el manejo de memoria en los sistemas operativos en ambos trabajos (Windows y Android).

Las versiones actuales de las bibliotecas empleadas(e.g. OpenCV, VTK y CGAL) están en proceso de completa adaptación para Android, lo cual agregó una mayor complejidad al proyecto, influyendo en los tiempos de desarrollo. Fue necesario compilar la biblioteca de VTK desde su código fuente, para lo cual fue configurado el entorno de compilación utilizando Make. Mientras que la biblioteca CGAL, que aumenta la precisión del modelo 3D final detectando y podando los puntos *outliers* encontrados en el proceso de escaneo, aún no es posible de compilar para Android.

Uno de los factores que tuvieron mayor influencia en la fuga de memoria fue la forma en que OpenCV procesa los cuadros de video en su cámara. La versión empleada no permite la captura de imágenes sin una superficie donde mostrar el resultado en la interfaz, requerimiento que hace más complicado el manejo de la cámara y la captura de imágenes. Con dicha API se cree que hubiera sido posible subsanar la fuga de memoria encontrado en la aplicación de Cámara en este T.E.G..

Hacer un escáner móvil en Android es posible. En un futuro podría conseguirse replicar o mejorar los resultados de Narváez [13] usando las versiones optimizadas de las bibliotecas, así como las nuevas APIs incorporadas en Android a partir de su versión 5.0. Dadas las limitaciones económicas se utilizaron dispositivos con prestaciones de hardware limitadas, los teléfonos utilizados como cámaras eran de gama media y tenían 1GB de RAM compartido con el sistema operativo, en dispositivos más actuales y de gama más alta podrían lograrse mejores resultados.

A pesar de que el enfoque utilizado en este trabajo fue probado y funcional para una PC con webcams, mostró no ser el ideal para los dispositivos utilizados. Algunas de las formas como podrían aprovechar las aplicaciones desarrolladas sería utilizando enfoques diferentes, o soluciones híbridas como:

- Las aplicaciones desarrolladas en este trabajo se podrían adaptar para integrarse con una API o Web Service externo en un servidor, con mayor capacidad de procesamiento, que se encargue de hacer los cálculos para generar el modelo 3D. Asi como servirlos al Controlador para ser desplegados en su interfaz.
- Seguir el enfoque probabilístico de Proforma [15], uno de los trabajos previos analizados podría servir para rodear las limitaciones de hardware y obtener resultados más satisfactorios.

Aunque el objetivo general no se pudo cumplir, 4 de los 5 objetivos específicos fueron cumplidos, así como el 75% del objetivo específico, generar un modelo 3D del objeto, que no se pudo completar. El robot fue diseñado e implementado con éxito, así como las aplicaciones para las cámaras y su papel en el *pipeline* de procesamiento, esto fue producto de un proceso de desarrollo incremental, hecho en 6 iteraciones, para integrar funcionalidades e ir resolviendo problemas encontrado.

El sistema de comunicación via Bluetooth fue uno de los aspectos más resaltantes de las funcionalidades completadas con éxito, tanto el pase de comandos como de datos fue implementado de una manera fácil de entender y de mantener. Al igual que la persistencia de datos dentro de los dispositivos,así como la creación de archivos en el sistema de almacenamiento del dispositivo.

Dado que no se pudieron completar todos los objetivos en el desarrollo de este trabajo de grado, en trabajos futuros se podría implementar una aplicación que tome los datos e imágenes generadas y aplique la última fase de procesamiento para generar el modelo 3D. La misma podría ser una aplicación sencilla para PC donde se copien los archivos manualmente al computador, o un Web Service con un API sencillo y una adaptación al Controlador de este trabajo mediante la cual se puedan transmitir los datos para ser procesados asi como obtener el modelo 3D final para ser almacenado y mostrado en el dispositivo del Controlador.

Igualmente, se podría implementar un repositorio público de modelos escaneados por los usuarios, implementando una opción para compartir el contenido escaneado dentro de la aplicación, siguiendo modelos como el de DeviantArt[15], con un enfoque más específico, estableciendo las bases para el desarrollo de una comunidad alrededor del escaneado de modelos con estos dispositivos. Otra alternativa sería investigar sobre plataformas ya existentes que cumplan la misma función e integrarlo con sus APIs.

[15] http://www.deviantart.com/

Glosario de Términos

- Android Studio: IDE oficial de Google para desarrollo de aplicaciones Android.
- Frame: Conjunto de imágenes izquierda y derecha capturadas desde un punto de control.
- GIT: Sistema de control de versiones. utilizado ampliamente en el desarrollo de software. Creado por LinusTorvalds. El mismo creador de GNU Linux.
- Holonómico: Es la capacidad de un vehículo para modificar su dirección instantáneamente sin necesidad de rotar previamente.
- IDE: IntegratedDevelopmentEnvironment. Un Ambiente integrado de desarrollo es un conjunto de herramientas, que suelen venir empacadas como un solo programa. Utilizado para el desarrollo de aplicaciones.
- Intents: Los Intents son mensajes asíncronos de Android que permiten a los componentes de una aplicación solicitar funcionalidades de otros componentes
- LeJOS: Firmware para Robots Lego Mindstorms. Incluye una maquina virtual de Java. Lo cual permite programar los robots utilizando este lenguaje mediante el SDK que incluye.
- Fuga de Memoria: Más conocido por el término inglés *memoryleak*, es un error de software que ocurre cuando un bloque de memoria reservada no es liberada en un programa de computación.
- OmniPilot: Clase de LeJOS SDK que permite controlar vehículos holonómicos con 3 ruedas omnidireccionales.
- WirelessPrivateArea Network (WPAN): Red Inalambrica de Área Personal. Es una red para comunicación entre dispositivos, generalmente pertenecientes a un mismo individuo. Generalmente de poco alcance, debido a su naturaleza “personal” la distancia de alcance que tienen este tipo de redes se podría definir como el del alcance de la voz del usuario.

Referencias

1. David Mindell (2000). *LEGO Mindstorms The Structure of an Engineering (R)evolution*. Massachusetts Institute of Technology (MIT). Recuperado de: http://web.mit.edu/6.933/www/Fall2000/LegoMindstorms.pdf
2. *About the LOGO Foundation*. The Logo Foundation. Consultado en Junio, 2013. Disponible en http://el.media.mit.edu/logo-foundation/about/index.html
3. *LHTC - Low Threshold High Ceiling*. Mind, Matter& Media Lab. Universidad Vanderbilt. Consultado en Junio, 2013. Disponible en http://www.m3lab.org/research/lthc
4. Brian Curless (2000). *From Range Scans to 3D Models*. ACM SIGGRAPH Computer Graphics 33 (4): 38–41. http://dl.acm.org/citation.cfm?doid=345370.345399
5. Rother C., Kolmogorov, V y Blake, A. (2004) "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts. Microsoft Research Cambridge, UK. Consultado en Agosto, 2015. Disponible en http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf
6. Lucas B., KanadeT.. (1981) An Iterative Image Registration Technique with an Application to Stereo Vision. Carnegie-MellonUniversity, Pennsylvania, USA. Consultado en Enero, 2016. Disponible en http://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf
7. *Android SDK*. AndroidDevelopers. Consultado en Octubre, 2013. Disponible en http://developer.android.com/sdk/index.html
8. *Android NDK*. AndroidDevelopers. Consultado en Octubre de 2014. Disponible en: http://developer.android.com/tools/sdk/ndk/index.html
9. *LeJOS, Java for Lego Mindstorms*. Sourceforge. Consultado en Julio, 2013. Disponible en. http://lejos.sourceforge.net/nxj.php
10. *Using leJOS with Android*. LEJOS. Java for LEGO Mindstorms. Sourceforge. Consultado en Octubre, 2013. Disponible en http://www.lejos.org/nxt/nxj/tutorial/Android/Android.htm#3.4
11. Specification | Adopted Documents. Bluetooth Technology Special Interest Group. Consultado en Octubre de 2015. Disponible en https://www.bluetooth.org/en-us/specification/adopted-specifications
12. NXT 3D Scanner. Consultado en Octubre, 2014. Disponible en http://www.philohome.com/scan3d/scan3d.htm
13. Narváez A. (2010). *Escáner 3D de bajo costo empleando WebCams*. Universidad Central de Venezuela (UCV). Recuperado de: http://busconest.ciens.ucv.ve/documento/obtener_archivo/550
14. *Fluid Scanning using milk, ink and other fun liquids*. Fluid Scanner - milk, ink and what have you. Consultado en Octubre, 2014. Disponible en http://fluidscanner.moviesandbox.net/
15. Pan Q., Reitmayr G. and Drummond (2009). *ProFORMA: Probabilistic Feature-based On-line*

Rapid Model Acquisition. Cambridge University. Recuperado de: http://www.bmva.org/bmvc/2009/Papers/Paper297/Paper297.pdf

16. *DIY 3D Laser Scanner Using Arduino*. Make Magazine. Consultado en Octubre de 2014. Disponible en http://makezine.com/projects/diy-3d-laser-scanner-using-arduino/
17. *Pi 3D Scanner: A DIY Body Scanner*. Raspberry Pi. Consultado en Octubre de 2014. Disponible en http://www.raspberrypi.org/pi-3d-scanner-a-diy-body-scanner/
18. *3D Digitizer Prototype with Calibration*. Muellerr .Consultado en Octubre de 2014. Disponible en http://www.muellerr.ch/engineering/3ddigitizer/
19. *Insight 3d: opensource image based 3d modeling software*. Insight 3d. Consultado en Octubre de 2014. Disponible en http://insight3d.sourceforge.net/
20. Bagnall, B. (2007). *Maximum Lego NXT: building robots with Java brains,Meet NXT (1-5)*. Canada: VariantPress.
21. *Asus Google Nexus 7 (2013) Full table specifications*. GSMArena. Consultado en Agosto 2015. Disponible en: http://www.gsmarena.com/asus_google_nexus_7_(2013)-5600.php
22. *Motorola Moto G 4G Full phone specifications*. GSMArena. Consultado en Agosto 2015. Disponible en: http://www.gsmarena.com/motorola_moto_g_4g-6355.php