INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

"Realidad Aumentada Aplicada a la Construcción de Modelos"

2012-B019

Que para cumplir con la opción de titulación curricular en la carrera de:

"Ingeniería en Sistemas Computacionales"

Presentan

Ángeles González Mariana
Basurto Esquivel Yael
Díaz Cortés Daniel
Reséndiz Ortega Guillermo

Directores

M. en C. Mario Augusto Ramírez Morales *M. en C. Jorge Cortés Galicia*

ESCOM

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

No. de registro: 2012-B019 Serie: Amarilla Noviembre del 2013

Documento técnico

Realidad Aumentada Aplicada a la Construcción de Modelos

Presentan

Ángeles González Mariana[1]
Basurto Esquivel Yael[2]
Díaz Cortés Daniel[3]
Reséndiz Ortega Guillermo[4]

Directores
M. en C. Mario Augusto Ramírez Morales
M. en C. Jorge Cortés Galicia

RESUMEN

En este reporte se presenta la documentación técnica del Trabajo Terminal 2012-B019 titulado "Realidad Aumentada Aplicada a la Construcción de Modelos", cuyo objetivo es desarrollar un sistema que reconozca figuras básicas de un entorno real y a partir de los resultados obtenidos, representarlas en modelos 3D sobre la pantalla de un dispositivo móvil, así como poder interactuar con estos modelos.

Palabras Clave: Dispositivos móviles, Realidad Aumentada, Reconocimiento de Patrones, Análisis de imagen

[1]E-mail: mariana_sp_1@hotmail.com
[2]E-mail: yael.basurto@gmail.com
[3]E-mail: daniel.diazco@gmail.com
[4]E-mail: guillermo.rortega@gmail.com

ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITUCIONAL

ESCOM

COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES

México, D.F. a 3 de Diciembre de 2013.

DR. FLAVIO ARTURO SÁNCHEZ GARFIAS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJOS TERMINALES
P R E S E N T E

Por medio del presente, informamos que los alumnos que integran el **TRABAJO TERMINAL** 2012-B019 titulado "Realidad Aumentada Aplicada a la Construcción de Modelos" concluyeron satisfactoriamente su trabajo.

El empastado del Reporte Técnico Final y el Disco Compacto (CD) fueron revisados ampliamente por sus servidores y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

M. en C. Mario Augusto Ramírez Morales

M. en C. Jorge Cortés Galicia

NOMBRE Y FIRMA DEL DIRECTORES DEL TRABAJO TERMINAL

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000 Extensión
52000.

Agradecimientos

Le agradezco a mi madre por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad, por brindarme un vida llena de aprendizajes, experiencias y sobre todo felicidad.
Le doy gracias a mis compañeros por apoyarme en todo momento y a pesar de los problemas que surgieron para realizar este trabajo, estuvieron siempre ahí poniendo de todo esfuerzo para que todo saliera bien.
Agradezco a mis amigos que durante el transcurso de mi formación siempre estuvieron ahí para sacarme una sonrisa, escucharme, pelear, llorar, etc.
Agradezco a la Escuela Superior de Cómputo por la formación académica.

Mariana Ángeles González.

Quiero agradecer en primer lugar a mis papás por brindarme su apoyo a lo largo no sólo de la carrera, sino a lo largo de toda mi vida, por ayudarme a cumplir está meta y reconfortarme en momentos difíciles.
A mis amigos y compañeros de TT por sacar este proyecto adelante, a todos nuestros amigos por mostrarnos su constante apoyo en todo el transcurso de la carrera y especialmente durante el desarrollo del TT, por los momentos compartidos, por las risas, las peleas y todos esos instantes de alegría que pasamos juntos.

Daniel Díaz Cortés

Este agradecimiento quiero dedicarlo especialmente a mis padres que sin su apoyo y confianza a lo largo de toda mi vida no hubiera sido posible cumplir este logro que no solo es mío sino también de ellos. También les quiero agradecer por estar en todos los momentos importantes de mi vida, al darme consejos y al ser tan comprensivos por muchas cosas que han tenido que soportarme.
A mis amigos y amigas de la Escuela Superior de Cómputo por todas aquéllas carcajadas que siempre teníamos en clases y en las excursiones que llegamos a ir.
También agradezco a mis amigos que ya son como parte de mi familia por todas las veces que me ayudaron cuando se me presentaba algún problema y porque siempre tenemos alguna anécdota chistosa cada que salimos.

Guillermo Reséndiz Ortega

Agradezco a mi madre, María Elena Esquivel Esquivel, por darme la vida, por el apoyo incondicional, por sus consejos y por su ejemplo, por enseñarme tantas cosas.
A mi familia en general, mi abuelita, mis tíos, tías, primos, por siempre estar ahí, porque de ellos he aprendido muchas cosas.
A mi mejor amiga y compañera Cristina Itzel Alberto Pimentel por ser parte de mi vida y por todas las vivencias que hemos pasado juntos.
Y a todas esas amistades que siempre han estado presentes en mi vida.

Yael Basurto Esquivel

Índice general

1. Introducción **1**

1.1. Problemáticas de desarrollo con realidad aumentada. 1
1.2. Aplicaciones. 1
1.3. Estado del Arte. 2
1.4. Planteamiento del problema. 4
1.5. Objetivo. 4
1.5.1. Objetivo General. 4
1.5.2. Objetivos Específicos. 4
1.6. Alcance. 6

2. Marco Teórico. **7**

2.1. Extracción de Características. 7
2.1.1. Procesamiento digital. 7
2.1.2. Segmentación. 9
2.1.3. Seguimiento. 9
2.1.4. Detección de bordes. 10
2.2. Creación del modelo de realidad aumentada. 11
2.3. Manipulación del modelo. 13

3. Propuesta de Solución. **14**

3.1. Modelo de despliegue. 14
3.2. Modelo funcional. 14
3.2.1. Requerimientos Funcionales. 15
3.2.2. Requerimientos No Funcionales. 15
3.3. Selección de la plataforma del sistema. 15
3.4. Herramientas de desarrollo. 15
3.5. OpenCV. 16
3.6. Diagrama de bloques. 19
3.7. Metodología 20
3.8. Entorno del sistema. 20
3.8.1. Pantalla de inicio. 21
3.8.2. Pantalla de instrucciones. 21
3.8.3. Pantalla principal. 21
3.8.4. Menú. 24

4. Prototipo I. Detección de figuras básicas. 25
4.1. Análisis de tecnología. . . . 25
4.1.1. OpenCV. . . . 25
4.1.2. Umbralización. . . . 26
4.1.3. Tipos de umbralización. . . . 26
4.1.4. Algoritmo de Canny. . . . 29
4.1.5. Algoritmo de Ramer-Douglas-Peucker. . . . 30
4.1.6. Algoritmo de Hough. . . . 30
4.2. Requerimientos. . . . 32
4.2.1. Requerimientos Funcionales. . . . 32
4.2.2. Requerimientos No Funcionales. . . . 32
4.3. Diagrama de Casos de Uso. . . . 33
4.3.1. Caso de Uso 1. Gestión de archivos . . . 33
4.3.2. Caso de Uso 2. Extracción de características . . . 34
4.4. Diagrama de Actividades. . . . 35
4.5. Diagramas de Secuencia. . . . 36
4.5.1. Gestión de archivos . . . 36
4.5.2. Extracción de características . . . 37

5. Prototipo II. Representación del Modelo. 38
5.1. Análisis de tecnología. . . . 38
5.1.1. OpenGL ES. . . . 38
5.1.2. Mapeo de coordenadas para dibujar objetos. . . . 39
5.1.3. Caras y devanado de las figuras. . . . 40
5.2. Requerimientos. . . . 40
5.2.1. Requerimientos Funcionales. . . . 40
5.2.2. Requerimientos No Funcionales. . . . 40
5.3. Diagrama de Casos de Uso. . . . 41
5.3.1. Caso de Uso 3. Construcción de modelo 3D con realidad aumentada. . . . 41
5.4. Diagrama de Actividades. . . . 42
5.5. Diagramas de Secuencia. . . . 44
5.5.1. Construcción de modelo 3D con realidad aumentada. . . . 44

6. Prototipo III. Manipulación del Modelo. 45
6.1. Requerimientos. . . . 45
6.1.1. Requerimientos Funcionales. . . . 45
6.1.2. Requerimientos No Funcionales. . . . 45
6.2. Diagrama de Casos de Uso. . . . 45
6.2.1. Caso de Uso 4. Rotación del modelo. . . . 46
6.2.2. Caso de Uso 5. Traslación del modelo. . . . 48
6.2.3. Caso de Uso 6. Escalamiento del modelo. . . . 49
6.3. Diagrama de Clases. . . . 50
6.4. Diagrama de Actividades. . . . 52
6.5. Diagramas de Secuencia. . . . 52
6.5.1. Rotación del modelo. . . . 52
6.5.2. Traslación del modelo. . . . 53
6.5.3. Escalamiento del modelo. . . . 53

7. Pruebas y resultados. **55**
7.1. Pruebas del prototipo I. 55
7.1.1. Entorno controlado 55
7.1.2. Pruebas 55
7.2. Pruebas del prototipo II. 65

Conclusiones y trabajo a futuro. **67**
7.3. Conclusiones. 67
7.4. Trabajo a futuro. 67

Índice de figuras

2.1. Ruido Gaussiano 8
2.2. Operador de Canny 11
2.3. Scketchup y Build AR 12
2.4. Blender 12

3.1. Modelo de despliegue del sistema 14
3.2. Diagrama de Bloques del Sistema Nivel 0 19
3.3. Diagrama de Bloques del Sistema Nivel 1 19
3.4. Prototipado Incremental 20
3.5. Entorno del sistema 21
3.6. Pantalla de inicio 22
3.7. Pantalla con las instrucciones del sistema 22
3.8. Pantalla Principal. Reconocimiento de cuadriláteros y triángulos 23
3.9. Pantalla Principal. Reconocimiento de círculos 23
3.10. Menú 24

4.1. Técnica de umbralización 26
4.2. Gráfica que muestra la técnica de umbralización 27
4.3. Gráfica de umbralización binaria 27
4.4. Gráfica de umbralización binaria invertida 27
4.5. Gráfica de la umbralización truncada 28
4.6. Gráfica de la umbralización a cero 28
4.7. Gráfica de la umbralización a cero invertida 29
4.8. Representación de una imagen al efectuar Canny 30
4.9. Recta inicial del algoritmo de Ramer-Douglas-Peucker 31
4.10. Toma de la distancia epsilon 31
4.11. Algoritmo de Ramer-Douglas-Peucker 31
4.12. Recta final del algoritmo de Ramer-Douglas-Peucker 31
4.13. Algoritmo de Hough 32
4.14. Diagrama General de Casos de Uso 33
4.15. Diagrama de Actividades 36
4.16. Gestión de archivos 37
4.17. Extracción de Características 37

5.1. Sistema de coordenadas de OpenGL 39
5.2. Modo en el que se dibuja la cara de un triángulo 40
5.3. Diagrama General de Casos de Uso 41
5.4. Diagrama de Actividades 43

5.5. Construcción de modelo 3D con realidad aumentada. 44

6.1. Diagrama General de Casos de Uso 46
6.2. Diagrama de Clases 51
6.3. Diagrama de Actividades 52
6.4. Rotación del modelo. 53
6.5. Traslación del modelo. 54
6.6. Escalamiento del modelo. 54

7.1. Pantalla de inicio. 63
7.2. Pantalla con las instrucciones de uso 63
7.3. Pantalla que muestra el menú 64
7.4. Reconocimiento de las figuras. 64
7.5. Reconocimiento de círculos. 65
7.6. Representación del modelo. 65
7.7. Representación del modelo. 66

Índice de cuadros

1.1. Estado del arte 5

3.1. Tabla comparativa de los S.O. móviles 16
3.2. Tabla comparativa de las API's de Realidad Aumentada 17
3.3. Tabla comparativa de las API's de Realidad Aumentada 18

7.1. Resultados de detección de cuadriláteros 58
7.2. Resultados de detección de triángulos 60
7.3. Resultados de detección de círculos 62

Capítulo 1

Introducción

La realidad aumentada es la rama de la computación que combina aspectos visuales de un entorno físico del mundo real con elementos visuales creados por computadora; es decir, realidad virtual. Mediante la realidad aumentada podemos visualizar el mundo real con información virtual sobrepuesta.

Los sistemas que utilizan realidad aumentada deben poseer las siguientes características principales:

1. Integrar información virtual dentro de una escena real de un modo realista e intuitivo.
2. Interactividad en tiempo real.
3. Objetos virtuales coherentes con el mundo real [1].
4. Está registrada en 3D.

Tiempo real: "Se dice que una aplicación es de tiempo real cuando su rendimiento temporal es el suficiente para resolver el problema al que está dedicado. Para ellos el sistema requiere responder a estímulos dentro de un límite de tiempo muy pequeño. (Usualmente milisegundos o microsegundos)" [2]

1.1. Problemáticas de desarrollo con realidad aumentada.

En este apartado se listan y describen brevemente las principales problemáticas que se tienen al desarrollar aplicaciones usando realidad aumentada.

Compatibilidad entre el mundo real y el virtual. Se basa en la necesidad de alinear los sistemas de coordenadas del mundo real con el virtual de forma que sean coherentes.

Seguimiento. Se refiere a la actualización de los sistemas de coordenadas del mundo real y virtual con respecto a la posición del dispositivo de entrada y del usuario en tiempo de ejecución.

Superposición de objetos. Se refiere a la posición en la que se presenta la información virtual sobre la secuencia de objetos del mundo real. Estos pueden colocarse al frente y en ocasiones es necesario que permanezcan ocultos total o parcialmente.

1.2. Aplicaciones.

Las aplicaciones desarrolladas con realidad aumentada han tenido un crecimiento considerable en los últimos años, por lo que podemos encontrar este tipo de aplicaciones en diversas áreas. Estas aplicaciones

pueden ser implementadas en medios donde el usuario requiere información adicional acerca del entorno en el que se encuentre o en tareas que involucren un alto riesgo al ser ejecutadas.

Algunas de las áreas y las aplicaciones donde se puede ocupar esta tecnología con un nivel de eficiencia aceptable son las siguientes: [1] [2]

Entretenimiento.

Esta área es la que obtiene los mayores ingresos económicos, ya que con ella se pueden crear juegos o aplicaciones móviles que interactúen con el entorno en el que los usuarios se desenvuelven, brindándole al usuario un nuevo modo de juego. Otra aplicación que llama mucho la atención de los observadores es el uso de publicidad virtual. En el mundo deportivo también es posible y muy común utilizar esta tecnología, ya que en eventos de este tipo se puede visualizar información adicional como anotaciones de los comentaristas, fueras de lugar, primera y diez, marcadores, etc.

Anotaciones y Visualización.

El uso principal en esta área es mostrar información que el usuario no sea capaz de observar a simple vista basándose en información geográfica que el GPS del dispositivo móvil nos brinda como pueden ser lugares o momentos de relevancia que se encuentren próximos al usuario como pueden ser restaurantes, librerías, museos, tráfico, accidentes, entre otros.

Milicia.

Esta es una de las primeras áreas que usó aplicaciones con realidad aumentada. Algunas de estas aplicaciones abarcan el cálculo de las trayectorias de los proyectiles; visores semireflejantes sobre los cuales se proyecta información sobre rutas, objetivos, el estado de la nave y otros, sistemas de localización en simuladores para localizar a los compañeros aún con obstrucción visual, como por ejemplo paredes.[3] [4]

Medicina.

Las aplicaciones que se usan en esta rama permiten visualizar y practicar cirugías. Esto es posible, mediante sensores no invasivos que usan los doctores como la resonancia magnética, tomografías o imágenes de ultrasonidos se pueden tomar conjuntos en 3D de un paciente en tiempo real para que posteriormente el doctor pueda visualizar los órganos del paciente y poder tratarlo.

1.3. Estado del Arte.

En el cuadro 1.1 se muestran productos que implementan técnicas de realidad aumentada así como una comparación entre cada producto tomando en cuenta la descripción, las plataformas en las que se puede instalar o si tiene una aplicación móvil.

Sistema de desarrollo para aplicaciones de realidad aumentada. [1]

El propósito de esta tesis es experimentar con la construcción de una herramienta que permita la creación de sistemas de realidad aumentada con el uso de video digital, proporcionando captura de

video, procesamiento de imágenes, reconocimiento de patrones, calibración de la cámara y desplegado de un objeto virtual en los marcos del video del mundo real.

Para verificar el correcto funcionamiento de la herramienta, experimentaron con dos diferentes métodos para la calibración de la cámara, se realizaron dos aplicaciones de realidad aumentada:

- Uso de un patrón de círculos en un plano.
- Calibración utilizando círculos concéntricos utilizando un patrón con dicha forma.

Augmented Reality Scouting for Interactive 3D Reconstruction. Primer prototipo de sistema de reconstrucción interactiva 3D de escenas urbanas. [5]

Un Augmented Reality scout es una persona equipada con un ultra-mobile PC, con una cámara USB y un receptor de GPS. Esta persona explora el ambiente urbano y determina una secuencia de imágenes en 2D. Estas imágenes son almacenadas y posteriormente se genera el modelo en 3D.

Sistema para la integración de Entornos Reales y Virtuales mediante Realidad Aumentada. [6]

Es un sistema que permite visualizar objetos que dentro del mundo real están ocultos y también cambiar la apariencia de objetos que si son visibles. Los objetos ocultos que muestra son redes de tubería y redes de cableado, y los otros objetos, incluyen paredes, techos, pisos, puertas y ventanas, esto con la finalidad de realizar diseño de interiores con ayuda de la AR. Este sistema utiliza OpenGL para el modelado de los objetos virtuales.

Realidad Aumentada en Interfaces Hombre Maquina. [2]

El trabajo propone un modelo de desarrollo de aplicaciones basadas en realidad aumentada sin marcadores, que incorpora un algoritmo sencillo de simulación de la superposición de objetos y con bajos requerimientos tanto en hardware y software. Para ello se propone la utilización de detectores y descriptores de puntos de interés, aprendizaje de máquina y geómetra proyectiva para la calibración de la cámara y un método de seguimiento para la posterior operación. También se muestra la aplicación del modelo propuesto en un sistema asistente de dibujo basado en realidad aumentada.

Desarrollo de un sistema de Realidad Aumentada en dispositivos móviles. [7]

Los objetivos principales del desarrollo de este trabajo fue implementar y validar una aplicación basada en RA y que a su vez se haya desarrollado para un dispositivo móvil. Por un lado, contaron con una aplicación funcional que se encargo de servir de herramienta de ayuda para el tratamiento de fobias a las cucarachas.

Layar 3D Model Converter. [8]

Layar es una empresa líder en la industria a la vanguardia de la realidad aumentada (AR). AR es una manera de ver la información digital que se ha superpuesto o aumentado en una vista en vivo de la física del mundo real que te rodea.

La herramienta Modelo Convertidor Layar3D se utiliza para convertir los modelos de Wavefront (.obj / .mtl) a formato Layar3D (. l3D) que se puede utilizar en Layar app.

ARToolKit. [9]

ARToolKit es una biblioteca de software para la creación de Realidad Aumentada (AR) en aplicaciones. Estas son las aplicaciones que implican la superposición de imágenes virtuales en el mundo real.

1.4. Planteamiento del problema.

En la actualidad existe un amplia gama de aplicaciones que utilizan realidad aumentada para distintos fines y en distintas áreas como hemos visto anteriormente. Sabemos, también, que éstas aplicaciones pueden ser utilizadas en dispositivos móviles tales como smartphones, tablets, y lentes, así como también en dispositivos fijos.

Una de las características más importantes con las que debe contar una aplicación que utiliza realidad aumentada es la interacción que hay entre el mundo real, el mundo virtual y cómo estas se comunican con el usuario. Para lograr esto, existen elementos con los que cuentan los diferentes dispositivos como pueden ser la cámara o el GPS de un teléfono móvil. Este trabajo Terminal se enfoca al uso de la cámara del dispositivo móvil.

Para añadir información en un ambiente real mediante el uso de la cámara de un dispositivo móvil la mayoría de las aplicaciones hacen uso de marcadores, códigos QR, imágenes precargadas, y otros para que, con base en el reconocimiento de estos, se sobreponga el entorno virtual. Sin embargo, son pocas las aplicaciones que hacen uso de un reconocimiento previo de los elementos del mundo real para posteriormente realizar la carga del entorno virtual sobre los resultados de la detección.

1.5. Objetivo.

En este apartado se plantean los objetivos específicos que hay que cumplir para lograr el objetivo general o principal.

1.5.1. Objetivo General.

Reconocer figuras básicas de un entorno real y en base a los resultados obtenidos de esta tarea, representarlas en modelos 3D por medio de la cámara de un dispositivo móvil con ayuda de realidad aumentada y una vez generados los modelos permitir al usuario interactuar con ellos.

1.5.2. Objetivos Específicos.

- Analizar patrones específicos y reconocer las figuras básicas (círculos, cuadriláteros y triángulos) en un entorno real a partir de la imagen presentada en la cámara de un dispositivo móvil.
- Representar las figuras básicas identificadas que conforman al objeto analizado con realidad aumentada.
- Interactuar con el modelo de realidad aumentada representado.

Título	**Autores**	**Año**	**Tecnología empleada**	**S.O.**	**Mobile /Desktop**	**Enfoque**
Sistema de desarrollo para aplicaciones de realidad aumentada	CINVESTAV Rosa Atzín Vázquez del ángel	2010	▪ DVGrab ▪ OpenGL utilizando glDrawPixels()	Unix	Desktop	Tratamiento de imágen digital
Augmented reality scouting for interactive 3D reconstruction	B. Reitinger D.Schmalstieg	2007	▪ Samsung Q1 ▪ GPS data	Sin especificar	Mobile	Reconstrucción de edificios mediante múltiples vistas
Realidad Aumentada en Interfaces Hombre Maquina	C.I.C. Gilberto Nájera Gutiérrez	2009	Captura de video con OpenCV en OpenGL	Sin especificar	Desktop	Propone un modelo genérico de aplicación de realidad aumentada para elaborar aplicaciones basadas en realidad aumentada
Layar 3D Model Converter	© 2013 Layar, All Rights Reserved	2013	▪ OpenGL 3D ▪ Java	Windows, Mac, y Linux	Mobile	Convierte modelos 2D en capas 3D
ARToolKit	ARToolworks, Inc	2007	▪ GPL ▪ QR code ▪ OpenGL	SGI IRIX, Linux, MacOS y Windows	Mobile	Biblioteca para aplicaciones de realidad aumentada

Cuadro 1.1: Estado del arte

1.6. Alcance.

El sistema desarrollado esta delimitado a la identificación de figuras básicas (círculos, cuadriláteros y triángulos) de objetos que se encuentren en el entorno mediante la cámara de un dispositivo móvil.

Una vez realizada esta tarea, el sistema representa cada una de estas figuras con realidad aumentada usando el SDK de Vuforia.

Así mismo, el usuario puede interactuar con las figuras representadas y realizar operaciones básicas sobre ellas, como son rotación, traslación y escalamiento usando los gestos que nos ofrece el dispositivo móvil.

Capítulo 2

Marco Teórico.

2.1. Extracción de Características.

El proceso de extracción de datos esta precedido por la captura de dicha información ya que una aplicación de realidad aumentada depende de las necesidades de la misma. Se puede trabajar con un sistema de realidad aumentada que obtenga la información de una cámara, de un sistema de posicionamiento (GPS), o de sensores de movimiento como lo es un acelerómetro en el caso de los dispositivos móviles, entre otras fuentes.

La extracción de datos o características de imagen (imagen captura o una secuencia de video) se aplican con diversas técnicas como es la detección de bordes, líneas, puntos, texturas, etc. Estas técnicas básicamente se ocupan para la descomposición de la imagen en los distintos objetos reales que la componen, facilitando la detección y jerarquización de la información de color de los pixeles para poder realizar la segmentación de los mismos.

2.1.1. Procesamiento digital.

El proceso comienza con el diseño de las propiedades de la captura, con esto nos referimos al tipo de cámara, distancia del objeto, mega pixeles, etc. Después el pre-procesamiento que consiste en reducir el entorno que no es de interés para nosotros, reconocer y extraer cada uno de los objetos que componen la imagen (segmentación).

El procesamiento digital de imágenes es uno de los procesos más importantes en la obtención de datos específicos ya que permite mejorar imágenes digitales para realizar mejores interpretaciones sobre su contenido.

En esta fase se trata de cuantificar y codificar la señal de vídeo o fotográfica recibida en forma de imagen. El objetivo es obtener una nueva imagen que bien mejore su calidad o que destaque algún atributo significativo de esta.

Los problemas de calidad pueden ser por falta o exceso de iluminación o por ruido. Al aplicar realce lo que se pretende conseguir es destacar bordes, regularizar colores, etc. Las técnicas descritas en este capítulo pueden ser agrupadas en dos conjuntos: las que proceden de las señales y aquellas de carácter heurístico.

En las técnicas para procesado procedentes de las señales se suelen aplicar los siguientes conceptos:

- Distancias entre pixeles. Dentro de esta categoría se pueden encontrar las siguientes relaciones:

    * Relaciones de distancias. Establece la distancia entre pixeles.
    * Relaciones de conectividad. Establece que dos pixeles adyacentes pertenecen a un mismo elemento.

- Procesos de convolución y operadores de correlación. Se utilizan para aplicar filtros sobre las imágenes, por ejemplo para eliminar ruido de sal y pimienta.

Por su parte, las técnicas de procesado de carácter heurístico se basan en un conjunto de procedimientos sobre el procesamiento digital de las señales y otros tipos de manipulaciones matemáticas. Este tipo de técnicas se pueden agrupar en tres conjuntos:

- Realce o aumento del contraste,
- Suavizado o eliminación del ruido y,
- Detección de bordes.

Para llevar a cabo estas técnicas conviene hacerlo sobre imágenes en escala de grises, ya que son efectivas sobre la luminancia de los objetos. A continuación se describirán brevemente cada conjunto de las técnicas recientemente expuestas.

Las técnicas de realce consisten en aumentar el contraste de las imágenes. Este tipo de procesado se basa en los conceptos de histograma, brillo y contraste. Por otro lado, las técnicas de suavizado lo que pretenden es eliminar el ruido que pueda tener la imagen. Existen tres tipos básicos de ruido: gaussiano como el que se muestra en la figura 2.1, impulsional y multiplicativo. Durante este proceso se utilizan filtros para eliminar ese ruido, los más comunes son:

- Filtros paso bajo.
- Filtros gaussianos.
- Filtros basados en la mediana.
- Filtros homomórficos.

La última técnica que se va a describir es la detección de bordes. Esta etapa suele preceder a las tareas de segmentación o a la búsqueda de objetos geométricos más complejos.

Figura 2.1: Ruido Gaussiano

2.1.2. Segmentación.

En el procesamiento de imágenes, la segmentación es lo primero que se debe hacer. Básicamente es el proceso en el que una imagen digital se particiona en muchos segmentos para cambiar a una representación más fácil de analizar, y terminar cuando se hayan detectado todos los puntos de interés.

Entre las técnicas de segmentación se encuentran estas cuatro:

- Técnicas basadas en valores de pixel,
- Técnicas basadas en el área,
- Técnicas basadas en el borde y,
- Técnicas basadas en la física.

De las técnicas mencionadas destaca:

Técnicas de Umbralización. Basada en valores de pixel. Busca obtener una imagen binarizada por un valor de umbral, donde se pueda separar el fondo(background) del objeto a separar (foreground).[11]

2.1.3. Seguimiento.

Uno de los principales retos durante el procesamiento de imágenes es el apareamiento (match) de puntos correspondientes en imágenes consecutivas. Para resolver esto, hacemos uso del seguimiento.

Considerando que el movimiento entre imágenes consecutivas es pequeño, se requiere obtener información para realizar el seguimiento de objetos. Para obtener dicha información podemos aplicar las siguientes lógicas:

- Posición cercana.
- Velocidad y distancia máxima.
- Inercia, cambios pequeños en la velocidad.
- Movimiento común.
- Consistencia.
- Movimiento conocido.

El seguimiento consiste en determinar la posición y velocidad de un punto en una imagen, dada su posición y velocidad en una secuencia anterior de imágenes, esto es, seleccionar una parte de una imagen y analizar su posición en cada frame de una secuencia. La información obtenida a partir del tracking de una o más partes de la imagen puede servirnos para varias cosas, desde integrar objetos en la misma, y que estos se adapten al movimiento de la secuencia, hasta estabilizar la imagen suprimiendo movimientos de cámara no deseados. La dificultad del seguimiento es el reconocimiento del objeto en cada frame.

El seguimiento puede realizarse con base en:

- Modelos tridimensionales de objetos.

- Características de la imagen.
- Modelos deformables.
- Regiones.

Seguimiento basado en modelo.

Se basa en el conocimiento del modelo del objeto. Este modelo puede ser un modelo CAD o una proyección del objeto. Esta técnica es muy robusta.

Seguimiento basado en características.

Este tipo de seguimiento se basa en la extracción de las características de la imagen. Este planteamiento suele ser computacionalmente más eficiente que el basado en modelo, pero es menos robusto.

El paso más importante antes de hacer el seguimiento es determinar las características a seguir. Para ello necesitamos discriminar entre las buenas y malas características del objeto a representar. Calculamos para ello el vector gradiente de la imagen en ese punto y con los valores propios, podremos saber la calidad de la característica. Sólo aquellas que cumplan un cierto umbral serán finalmente elegidas.

Una vez seleccionadas las características buenas a seguir, es sencillo llevar un registro del movimiento de cada característica, con lo que tenemos determinada la trayectoria de movimiento del objeto.

Predicción de la trayectoria.

Se realiza cuando los requerimientos dinámicos o la velocidad del objeto a seguir es lo suficientemente alta como para que su localización entre una imagen y la siguiente de la secuencia sea muy diferente.

La predicción nos proporciona información sobre la posible pose cuando se esté analizando una ventana de la imagen, existan oclusiones parciales o totales del objeto o incluso éste salga del marco de imagen.

Métodos para el manejo del seguimiento.

Filtro de Kalman. Usa una combinación de los datos de obtención de la pose y predicción para hacer más robusto el algoritmo.

Filtros $\alpha\beta\gamma$. El filtro $\alpha\beta$, es un sencillo estimador de la posición con un tiempo de cómputo muy bajo y que proporciona unos resultados satisfactorios. Como modificación, tenemos el filtro $\alpha\beta\gamma$, que realiza la estimación de la aceleración, efecto que no se contemplaba y que provocaba el mayor error de estimación.

Algoritmo ME. Algoritmo de estimación que calcula el desplazamiento normal entre dos imágenes sucesivas a lo largo de la proyección del contorno del modelo del objeto.

2.1.4. Detección de bordes.

Los bordes de una imagen digital pueden ser definidos como transiciones entre dos regiones de niveles de grises significativamente distintos; se caracterizan por éstas transiciones de claro a oscuro o viceversa.

Generalmente son usadas técnicas de detección basadas en operaciones o aproximaciones discretas sobre la primera y segunda derivada de los niveles de grises de la imagen digital. Algunas de éstas técnicas son:

- Operador de Roberts
- Operador de Prewitt
- Operador de Sobel
- Operador de Frei-Chen
- Algoritmo de Canny

Básicamente, estos operadores calculan el gradiente de la imagen en dos direcciones ortogonales y calculan la intensidad de la imagen en cada punto. Con base en el índice de cambio que se obtiene dando la dirección de aumento posible de luz a oscuridad se calcula qué tan probable es que esa parte en la imagen sea un borde.

En las bibliotecas OpenCV podemos encontrar los algoritmos Canny o Sobel utilizados mediante las funciones cvSobel y cvCanny respectivamente. [10]

Figura 2.2: Operador de Canny

Gradiente: Es el vector producto de la derivada de una señal continua en el caso de funciones bidimensionales $f(x, y)$. Proporciona las variaciones locales con respecto a una variable; a mayor valor de la derivada, más rápidas son estas variaciones.

El vector del gradiente apunta en la dirección de la máxima variación de $f(x, y)$.

2.2. Creación del modelo de realidad aumentada.

Para crear realidad aumentada requerimos monitores para desplegar la información agregada, cámaras que nos permitan ver el entorno físico para combinarlo con el virtual y, en ocasiones, marcadores que nos permiten reconocer dónde poner la información con realidad aumentada. Sin embargo para crear un objeto a desplegar sobre la imagen del mundo real se requiere software que permita la creación de estos modelos.

Para apoyarnos en el desarrollo de una aplicación con realidad aumentada existen varias herramientas que brindan la posibilidad de creación de escenas, marcas, etc. Existen programas de reconocimiento de imágenes, orientación espacial y programas para superposición de imágenes en tiempo real.

Por mencionar algunos de estos programas y su funcionalidad mencionaremos los siguientes.

- Para la creación de marcas y escenas de realidad aumentada, así como modelado, animación y creación de gráficos tridimensionales. [12] [13] [15]
  * BuildAR en versión gratuita y premium.
  * Atomic.Software libre.
  * Blender. De código abierto y multiplataforma.
- Creación de modelos y elementos dinámicos. [14] [9]
  * Google Scketchup. Permite exportar los objetos en formato .obj
  * ARToolKit implementado a Flash mediante la biblioteca FLARToolKit.

Además de estos y otros programas que nos permiten la creación de gráficos y animaciones tridimensionales, existen en internet páginas dedicadas que proporcionan gran cantidad de modelos tridimensionales en diversos formatos, como el .obj que es de los más usuales.

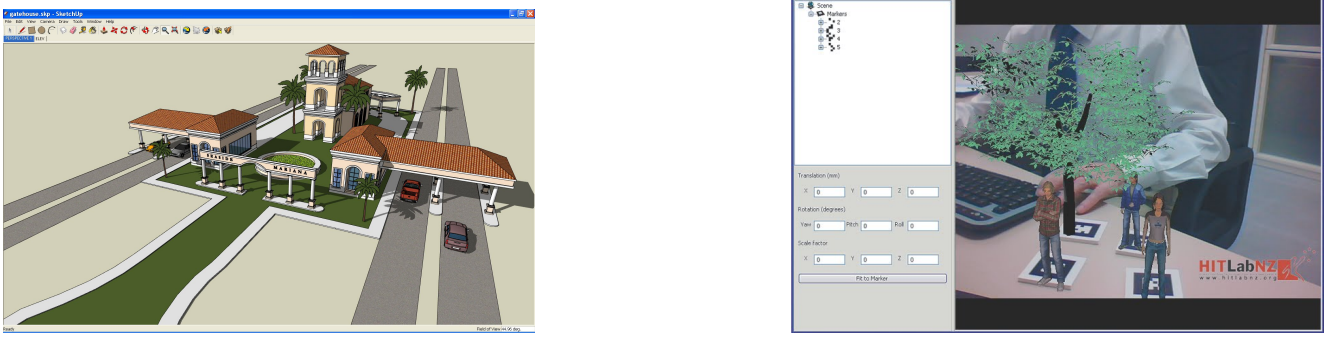


Figura 2.3: Scketchup y Build AR

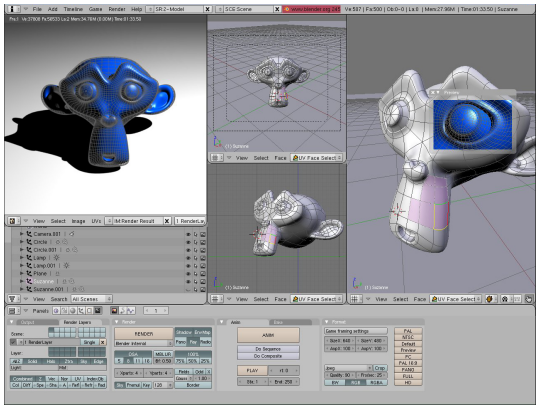


Figura 2.4: Blender

2.3. Manipulación del modelo.

Los Realidad Aumentada ofrece a los usuarios interfaces más intuitivas para la manipulación de modelos en 3D. Por ejemplo interfaces de usuario tangibles, usando el marcador de posición (el que genera el modelo) y de una forma rústica girar el marcador y obtener la rotación del modelo 3D.

Proyectos que conjuntan la RA con el kinect de Microsoft, lo hacen con el uso de una cámara y del cuerpo humano, la cámara del kinect enfoca el marcador del modelo de AR y es proyectado en una pantalla, la cámara del kinect también enfoca a una persona y en general los movimientos que hacen sus manos, así cuando las manos se juntan o se separan, hace sobre el modelo de realidad aumentada una operación de escalación, de una forma parecida aplica operaciones sobre el modelo de rotación y mover el modelo en el plano x, y, z.

Capítulo 3

Propuesta de Solución.

En este capítulo se realiza el análisis de los requerimientos funcionales y no funcionales generales del sistema, además de las herramientas que van a utilizarse para el desarrollo del sistema, tomando en cuenta sus principales características, ventajas, desventajas y aplicaciones en las que han destacado.

3.1. Modelo de despliegue.

En la figura 3.1 se puede observar el modelo de despliegue del sistema.

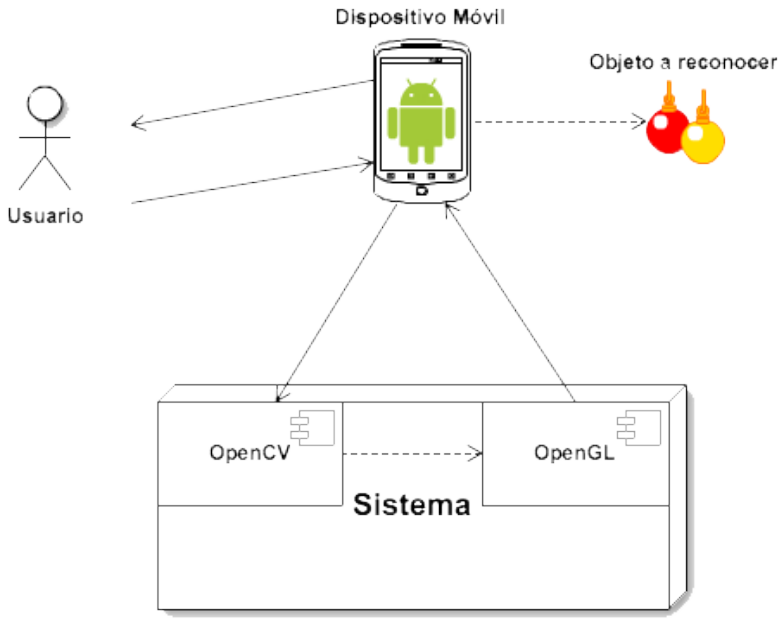


Figura 3.1: Modelo de despliegue del sistema

3.2. Modelo funcional.

En esta sección se listan los requerimientos funcionales y no funcionales del sistema. La primera lista se refiere a aquellos requerimientos indispensables para el funcionamiento del sistema, mientras que la segunda lista describe aquellos requerimientos necesarios para el correcto funcionamiento del sistema, pero que no afecten las funciones del sistema.

3.2.1. Requerimientos Funcionales.

- El sistema obtiene frames del entorno real mediante la cámara del dispositivo móvil en el que se ejecute.
- El sistema reconoce figuras básicas (círculos, cuadriláteros, triángulos) a través de los frames obtenidos.
- El sistema localiza características o atributos de interés, dependiendo de las figuras reconocidas:
  - radio y centro (x, y) para círculos.
  - coordenadas de los vértices para cuadriláteros y triángulos.
- El sistema genera el modelo de las figuras básicas con realidad aumentada coherente utilizando las características de interés obtenidas.
- El sistema puede realizar operaciones básicas de traslación, rotación y escalamiento sobre el modelo generado en el entorno virtual.
- El sistema tiene la capacidad de alterar el modelo según la operación que el usuario le solicite.

3.2.2. Requerimientos No Funcionales.

- La cámara del dispositivo debe tener una resolución mínima de 5 megapixeles.
- El sistema puede ser ejecutado sobre un sistema móvil Android con la versión 2.3 como mínimo.
- El procesador del dispositivo móvil debe ser al menos de dos núcleos.
- El sistema no hace uso de marcadores para representar los modelos de realidad aumentada.
- El sistema hace uso de gestos con los dedos para la manipulación del modelo de realidad aumentada.
- El ambiente de pruebas del sistema debe ser controlado.

3.3. Selección de la plataforma del sistema.

En este apartado se realizó un estudio comparativo acerca de los diferentes sistemas operativos que existen en el mercado para el desarrollo de sistemas móviles. Esto con la finalidad de escoger el sistema operativo que más se acomode a nuestras necesidades.

En el cuadro 3.1 se comparan las características más importantes de los sistemas operativos para dispositivos móviles que se encuentran en el mercado actualmente.

3.4. Herramientas de desarrollo.

Así mismo, se elaboró un estudio comparativo para seleccionar las herramientas que nos permitieran realizar la tarea de representar el entorno virtual con realidad aumentada.

En los cuadros 3.2 y 3.3 se comparan los diferentes SDK‘s de realidad aumentada existentes para desarrollar el sistema.

	Apple iOS 6	**Android 4.2**	**Windows Phone**	**BlackBerry OS 7**
Familia CPU soportada	ARM	ARM, MIPS, Power, X86	ARM	ARM
Lenguaje de programación	Objective-C, C++	Java, C++	C#	Java
Licencia de software	Propietaria	Software libre y abierto	Propietaria	Propietaria
Motor del navegador web	WebKit	WebKit	Pocket Internet Explorer	WebKit
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry App World
Actualizaciones automáticas del S.O.	Si	Depende del fabricante	Depende del fabricante	Si
Soporte memoria externa	No	Si	No	Si
Tipo de pantalla	Capacitiva	Capacitiva/ Resistiva	Capacitiva	Capacitiva/ Resistiva

Cuadro 3.1: Tabla comparativa de los S.O. móviles

3.5. OpenCV.

Para el reconocimiento y extracción de características de objetos del mundo real, se utilizó la librería libre de visión artificial **OpenCV**, ya que en ella se tienen algoritmos que nos ayudan a realizar el reconocimiento de las figuras básicas localizadas en el entorno real en tiempo real. Esta librería posee así características que nos sirven para desarrollar el sistema. Las características principales son citadas a continuación:

- Su publicación se da bajo la licencia BSD, es decir, permite que sea usada para fines comerciales y de investigación bajo las condiciones.
- Es multiplataforma (Windows, GNU/Linux, Mac OS X).
- Su programación está basada en C y C++.
- Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica.
- Su código puede ser implementado para dispositivos móviles iOS y Android.
- Tiene más de 2500 algoritmos optimizados.
- Tiene interfaces en C, C++, Python y Java.
- Su uso principalmente es en aplicaciones de visión en tiempo real y toma ventaja de instrucciones MMX y SSE cuando estén disponibles.

SDK o Biblioteca	ARviewer SDK	Vuforia
Características	▪ Permite etiquetar y visualizar en diferentes alturas. ▪ Permite integrarlo en aplicaciones de una manera sencilla. ▪ Permite pintar etiquetas asociadas a objetos de la realidad utilizando la posición GPS y su altitud. ▪ El sistema funciona tanto en exteriores como en interiores. ▪ Muestra toda la información multimedia en una vista de Realidad Aumentada. ▪ Posee un avanzado modo de navegación y búsqueda.	▪ Tiene soporte para iOS, Android y Unity 3D. ▪ Para instalarlo en Android se necesita de Eclipse IDE y de todo el entorno de desarrollo para Android. ▪ Para instalarlo en iOS, necesitamos XCode y una cuenta de desarrollador. ▪ Permite trabajar con tres tipos de trakeables diferentes: image targets, multi targets y frame markers.
Ventajas	▪ Puede ser usado para crear y subir contenido, posibilitando infinidad de aplicaciones sociales o juegos donde se deseen crear y asociar etiquetas con una altitud, longitud y latitud desde el propio dispositivo. ▪ El SDK es gratuito.	▪ Tiene soporte para iOS, Android y Unity 3D. ▪ El SDK es gratuito. ▪ Hace uso de código nativo de Android.
Desventajas	▪ Sólo funciona para sistemas Android. ▪ En interiores solo funciona con códigos QR.	El reconocimiento que hace sólo se aplica para marcadores que previamente están precargados en el sistema.

Cuadro 3.2: Tabla comparativa de las API's de Realidad Aumentada

SDK o Biblioteca	**Metaio SDK**	**ARToolKit**
Características	▪ Desarrollo nativo para iOS, Android y Windows. ▪ Posee un potente motor de renderizado 3D. ▪ Incluye plugin para características avanzadas de juego Unity3D.	▪ Trabaja con smartphones y tabletas Android que tengan versión 2.2 o mayor. ▪ Seguimiento en cada cuadro de la cámara. ▪ Posee todas las funciones de render usando OpenSceneGraph 3.x. ▪ Ajusta el tamaños de la imagen y puede cambiar a la cámara frontal. ▪ Soporta la función de seguimiento de objetos.
Ventajas	▪ El SDK es gratuito. ▪ Multiplataforma.	▪ El SDK es gratuito. ▪ Compatible para Android e iOS. ▪ Multiplataforma.
Desventajas	▪ Solo soporta dispositivos Android.	▪ Tiene bugs con algunos sistemas operativos.

Cuadro 3.3: Tabla comparativa de las API's de Realidad Aumentada

3.6. Diagrama de bloques.

En esta sección se muestran los diagramas a bloques del sistema en sus niveles 0 y 1, explicando cada uno de los módulos con los que el sistema cuenta.

En la figura 3.2 se tiene el diagrama a bloques del sistema en su nivel 0, que muestra el sistema como una caja negra, además de las entradas y salidas que lo constituyen.

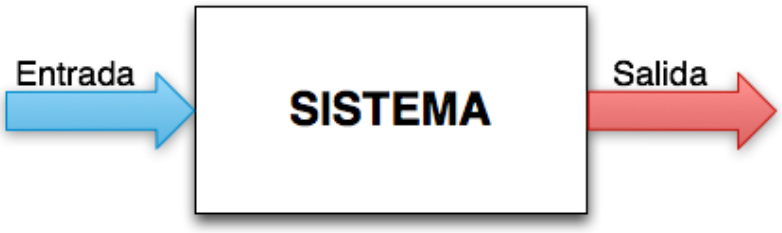


Figura 3.2: Diagrama de Bloques del Sistema Nivel 0

La figura 3.3 muestra el diagrama a bloques del sistema de forma más detallada y cada uno de los módulos que sigue desde su entrada hasta su salida. A su entrada se pueden observar los frames del entorno tomados en tiempo real por la cámara del dispositivo. Mientras son tomados los frames se realiza un procesamiento de ellos, esto con el fin de detectar las figuras geométricas que se encuentren en nuestro entorno y obtener los puntos de interés de cada una de ellas. Finalizado este proceso, la siguiente etapa es generar el modelo con realidad aumentada y presentarlo en la pantalla del móvil. Una vez desplegado el modelo final, el usuario puede manipularlo usando operaciones básicas de traslación, rotación y escalamiento.

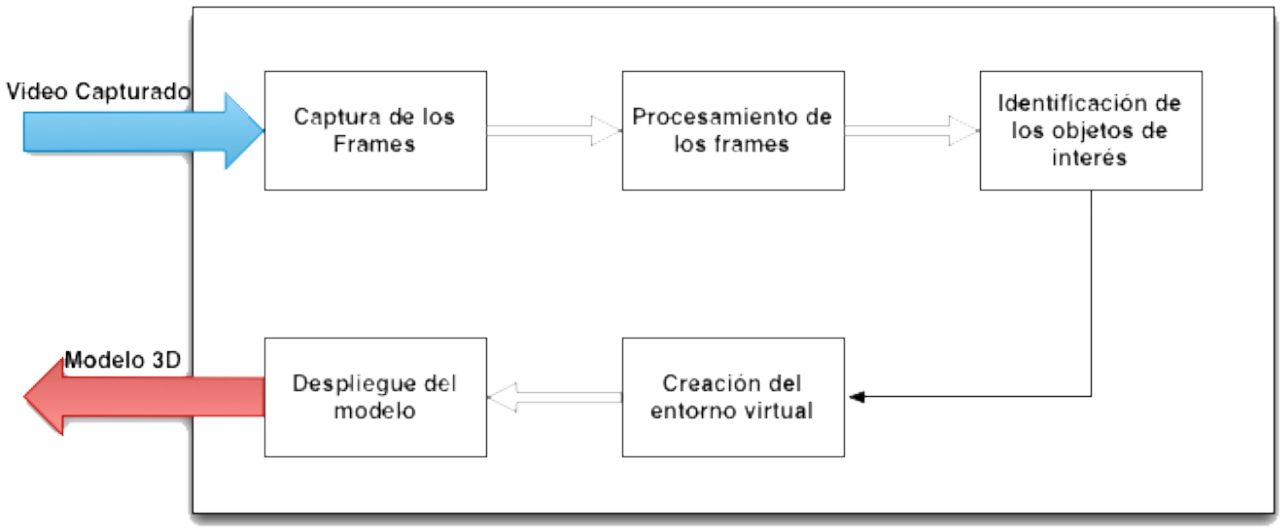


Figura 3.3: Diagrama de Bloques del Sistema Nivel 1

3.7. Metodología

La realización de este proyecto utiliza una metodología de prototipado incremental, que se basa en la generación de varios modelos parciales ejecutables del sistema antes de proceder a la implementación con el fin de evaluar sus características y poder obtener como resultado final el sistema implementado.

Las fases que se dan en la construcción de los distintos prototipos de un desarrollo son:

1. Identificación de requisitos que debe de cumplir el prototipo.
2. Diseñar e implementar el prototipo.
3. Utilizar el prototipo con el fin de probar que cumple los requisitos para los que fue diseñado.
4. Revisar y mejorar el prototipo.

La figura 3.4 ejemplifica la metodología a utilizar; en él, se aprecia de manera esquemática las fases llevadas a cabo para el desarrollo de software mediante prototipado incremental.

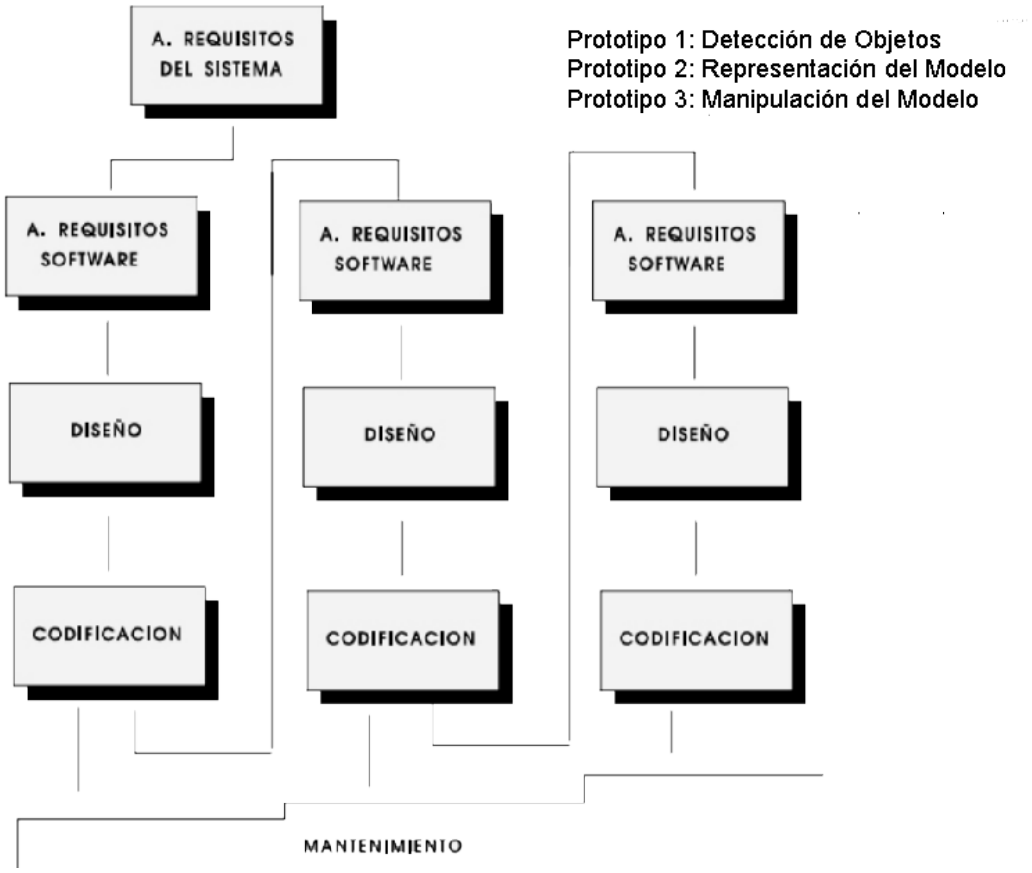


Figura 3.4: Prototipado Incremental

3.8. Entorno del sistema.

El entorno del sistema es básico y se muestra en la figura 3.5. Se puede observar que consta de una pantalla que muestra lo que la cámara del dispositivo móvil toma, además de un menú que se despliega

al dar doble tapping en esta misma pantalla, así como una barra en la parte inferior que contiene botones para la funcionalidad del sistema

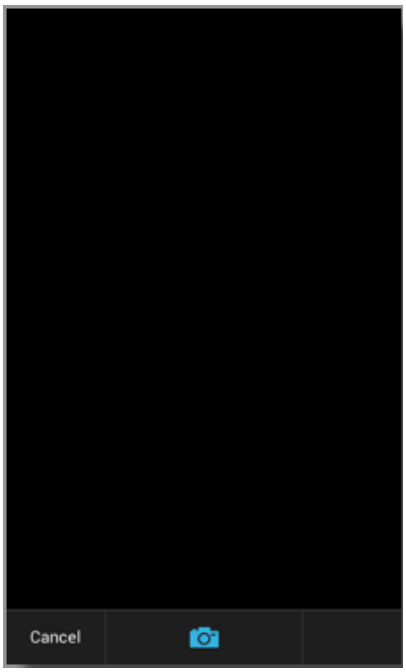


Figura 3.5: Entorno del sistema

3.8.1. Pantalla de inicio.

Al iniciar el sistema, se mostrará la Pantalla de Inicio (figura 3.6), la cual solo muestra el nombre del Trabajo Terminal junto con los logotipos del Instituto Politécnico Nacional (IPN) y de la Escuela Superior de Cómputo (ESCOM), así como un temporizador para cargar la siguiente pantalla del sistema.

3.8.2. Pantalla de instrucciones.

Una vez que el temporizador de la Pantalla de Inicio haya concluido, se muestra una pantalla con las instrucciones (ver figura 3.7) para interactuar con el sistema. También contiene un botón en la parte inferior para cerrar la pantalla y poder visualizar la Pantalla principal.

3.8.3. Pantalla principal.

Una vez que se hayan cerrado las instrucciones, se muestra la pantalla principal del sistema, en ella se tiene una vista donde se muestran los frames que la cámara del dispositivo obtiene. En los botones de funcionalidad se encuentra el botón cancelar, el cual cierra la aplicación, y el botón con el ícono de la cámara, que muestra las figuras básicas con realidad aumentada que hayan sido previamente reconocidas.

Esta pantalla reconoce figuras básicas como cuadriláteros y triángulos (figura 3.8), así como círculos (figura 3.9). Esta pantalla posee un menú que se muestra cuando damos doble tapping sobre ella.

Figura 3.6: Pantalla de inicio

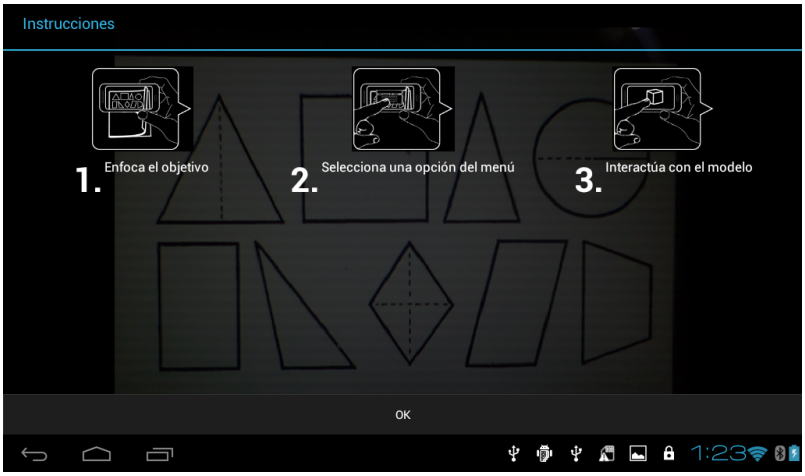


Figura 3.7: Pantalla con las instrucciones del sistema

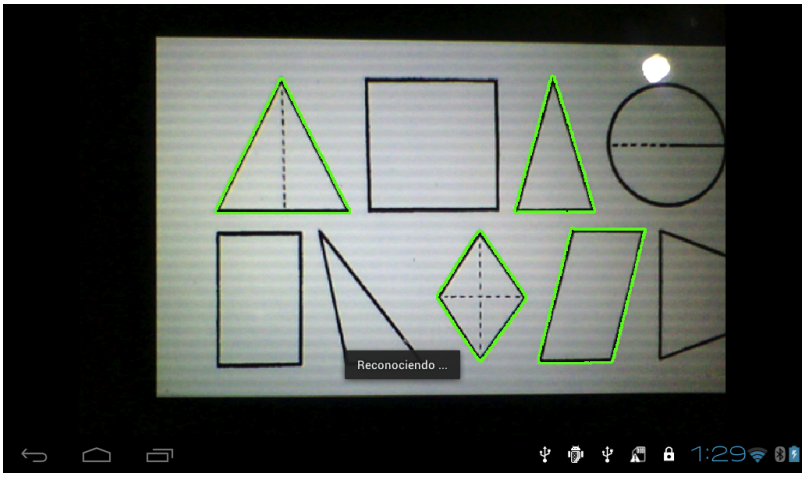


Figura 3.8: Pantalla Principal. Reconocimiento de cuadriláteros y triángulos

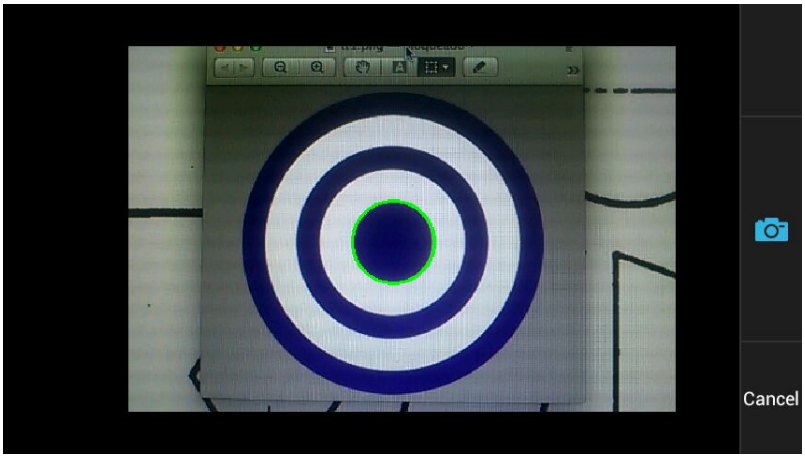


Figura 3.9: Pantalla Principal. Reconocimiento de círculos

3.8.4. Menú.

Este menú contiene opciones para especificar el tipo de figura que se desea reconocer, es decir si serán círculos o cuadriláteros y triángulos. En la figura figura 3.10 se puede observar el menú descrito.

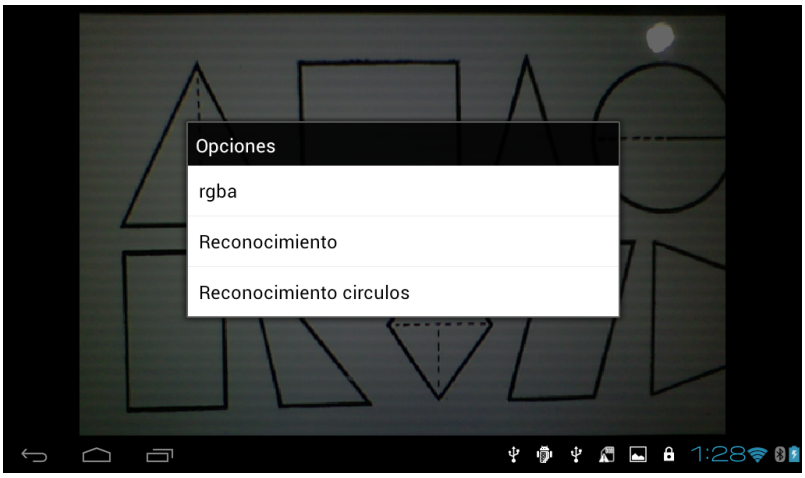


Figura 3.10: Menú

Capítulo 4

Prototipo I. Detección de figuras básicas.

Este capítulo describe el análisis y diseño correspondiente al prototipo I, cuya función principal es procesar los frames recibidos por el dispositivo móvil, además de reconocer y mostrar las figuras básicas que haya encontrado.

4.1. Análisis de tecnología.

Para el desarrollo del primer prototipo, la API de la que se hizo uso fue OpenCV. A continuación se presentan las características principales de esta API, además se describen los algoritmos empleados por el sistema.

4.1.1. OpenCV.

OpenCV es una librería basada en licencia BSD que incluye más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica. Su programación está basada en C y C++.

OpenCV se basa en una estructura modular, lo cual significa que el paquete incluye varias bibliotecas compartidas o estáticas. Los siguientes módulos están disponibles:

- **core** - un módulo compacto que define las estructuras de datos básicas, incluyendo la matriz multidimensional densa "Mat" y las funciones básicas utilizadas por todos los demás módulos.
- **imgproc** - un módulo de procesamiento de imágenes que incluye filtrado de imágenes lineales y no lineales, transformaciones de imágenes geométricas, conversión de espacio de color, histogramas, etc.
- **video** - un módulo de análisis de video que incluye la estimación de movimiento, la sustracción de fondo, y los algoritmos de seguimiento de objetos.
- **calib3d** - contiene los algoritmos de la geometría básica de múltiples vistas, calibración simple y estéreo de la cámara y elementos de reconstrucción en 3D.
- **features2d** - posee detectores de características sobresalientes y descriptores.
- **objdetect** - detección de objetos e instancias de las clases predefinidas (por ejemplo, caras, ojos, personas, coches, etc.).

- **highgui** - una interfaz fácil de usar para la captura de vídeo, imagen y codecs de vídeo, así como las capacidades simples de interfaz de usuario.
- **gpu** - Algoritmos acelerados por GPU de diferentes módulos OpenCV.

4.1.2. Umbralización.

Es el método más simple de segmentación. Separa regiones externas de una imagen correspondientes a objetos que queremos analizar. Esta separación está basada en la variación de intensidad entre los pixeles del objeto y los pixeles del fondo.

Para diferenciar los pixeles que nos interesan del resto, llevamos a cabo una comparación de cada valor de intensidad de los pixeles con respecto a un umbral acorde al problema que se quiera resolver.

Una vez que hemos separado correctamente los pixeles de importancia, podemos establecerles un valor determinado para identificarlos, es decir, podemos asignarles un valor de 0 (negro), 255 (blanco) o cualquier valor que se adapte a nuestras necesidades. En la figura 4.1 se puede observar una imagen al pasar por la técnica de umbralización.

Figura 4.1: Técnica de umbralización

4.1.3. Tipos de umbralización.

OpenCV nos ofrece la función **threshold** para poder efectuar operaciones de umbralización. Se pueden efectuar 5 tipos de operaciones de umbralización con esta función. En la figura 4.2 se ilustra cómo funciona este proceso de umbralización, vamos a considerar que tenemos una imagen con pixeles que tienen valores de intensidad src(x, y). La linea azul de la imagen representa el umbralizado **thresh**.

Umbralización binaria.

Esta operación puede ser expresada de la siguiente forma:

$$dst(x,y) = \begin{cases} maxVal & \text{si } src(x,y) > umbral \\ 0 & \text{en otro caso} \end{cases}$$

Entonces si la intensidad del pixel $src(x, y)$ es mayor que el umbral, entonces la nueva intensidad del pixel toma el valor de maxVal, de lo contrario tomará el valor de 0. La figura 4.3 muestra la gráfica correspondiente a la umbralización binaria.

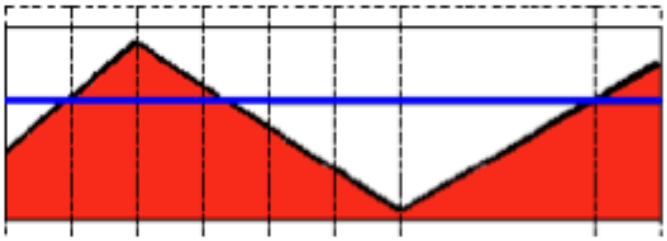


Figura 4.2: Gráfica que muestra la técnica de umbralización

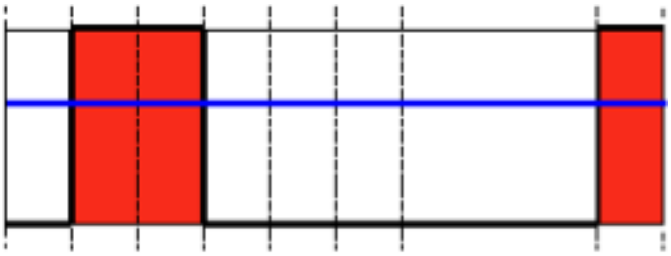


Figura 4.3: Gráfica de umbralización binaria

Umbralización binaria invertida.

Esta operación puede ser expresada de la siguiente forma:

$$dst(x,y) = \begin{cases} 0 & \text{si } src(x,y) > umbral \\ maxVal & \text{en otro caso} \end{cases}$$

Este tipo de umbralización es muy parecida a la umbralización binaria, la diferencia está en que si la intensidad del pixel $src(x,y)$ es mayor que el umbral, entonces el nuevo valor de la intensidad del pixel será 0, de lo contrario tomará el valor de maxVal. La figura 4.4 muestra la gráfica correspondiente a la umbralización binaria invertida.

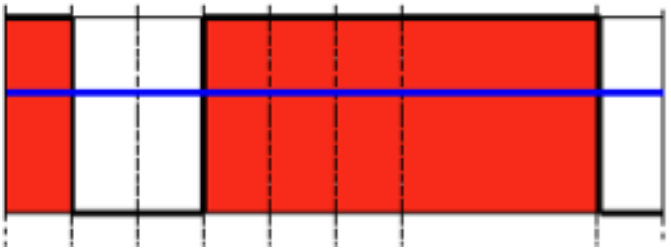


Figura 4.4: Gráfica de umbralización binaria invertida

Truncado.

Esta operación de umbralización puede ser expresada de la siguiente manera:

$$dst(x,y) = \begin{cases} umbral & \text{si } src(x,y) > umbral \\ src(x,y) & \text{en otro caso} \end{cases}$$

Este tipo de umbralización trabaja con un umbral, el cual será nuestro valor de intensidad máximo. Si src(x, y) es mayor que nuestro umbral, entonces el valor de intensidad del pixel es truncado; en otro caso, el valor de la intensidad del pixel se mantiene. La figura 4.5 muestra la gráfica correspondiente a la umbralización binaria invertida.

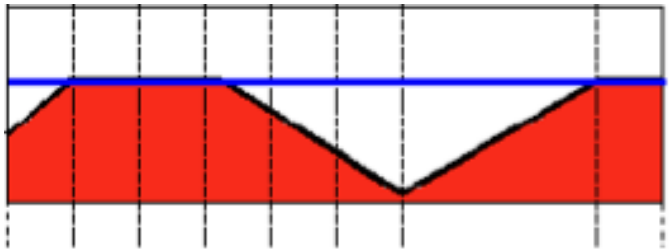


Figura 4.5: Gráfica de la umbralización truncada

Umbralización a cero.

Esta operación de umbralización puede ser expresada de la siguiente manera:

$$dst(x,y) = \begin{cases} src(x,y) & \text{si } src(x,y) > umbral \\ 0 & \text{en otro caso} \end{cases}$$

Esto nos dice que si $src(x,y)$ es menor que nuestro umbral, el nuevo valor de nuestro pixel será 0. En la figura 4.6 se muestra la gráfica de cómo funciona este tipo de umbralización.

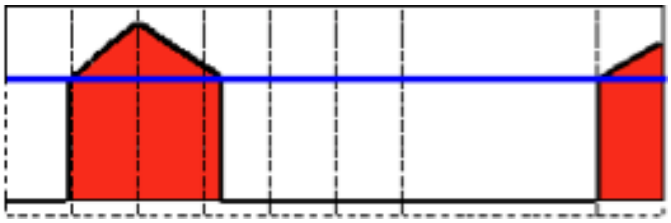


Figura 4.6: Gráfica de la umbralización a cero

Umbralización a cero invertida.

Esta operación de umbralización puede ser expresada de la siguiente manera:

$$dst(x,y) = \begin{cases} 0 & \text{si } src(x,y) > umbral \\ src(x,y) & \text{en otro caso} \end{cases}$$

Es el mismo proceso que la umbralización a cero, con el cambio de que si $src(x,y)$ es mayor al valor del umbral, el valor que tomará este pixel será 0. En la figura 4.7 se muestra la gráfica de cómo funciona este tipo de umbralización.

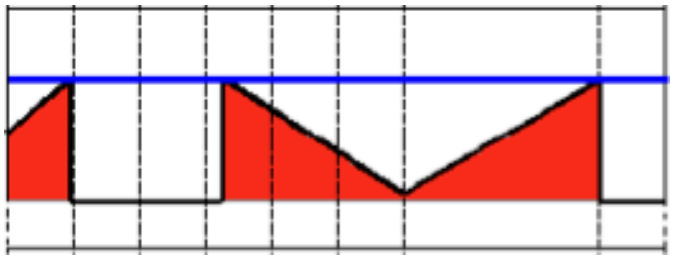


Figura 4.7: Gráfica de la umbralización a cero invertida

4.1.4. Algoritmo de Canny.

El detector de bordes Canny fue desarrollado por John. F Canny en 1986. También es conocido como el detector óptimo, el algoritmo de Canny pretende satisfacer tres criterios principales:

- **Baja tasa de error:** Presenta una buena detección de bordes existentes.
- **Buena localización:** La distancia entre los pixeles de los bordes detectados y los bordes reales tiene que ser mínima.
- **Respuesta mínima:** Solo un detector responde por borde.

Los pasos que lleva a cabo este algoritmo son los siguientes:

1. Filtrar cualquier ruido. El filtro Gaussiano es usado para este propósito. Un ejemplo de un kernel Gaussiano de $tamaño = 5$ que puede ser usado se muestra a continuación:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Encontrar el gradiente de intensidad de la imagen. Para esto, se sigue un procedimiento análogo a Sobel:
   - Aplicar un par de máscaras de convolución en las direcciones x y y:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

   - Encontrar la fuerza y la dirección del gradiente con:

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = arctan(\frac{G_y}{G_x})$$

3. Se aplica la no supresión máxima. Estos quitan los pixeles que no son considerados parte de un borde. Por lo tanto, sólo las líneas finas se mantendrán.

4. Histéresis. Canny hace uso de dos umbrales (superior e inferior):

   - Si el gradiente de un pixel es mayor que el umbral superior, el pixel es aceptado como un borde.
   - Si el gradiente de un pixel se encuentra abajo del umbral inferior, entonces el pixel es rechazado.
   - Si el gradiente del pixel está entre los dos umbrales, entonces será aceptado sólo si se encuentra conectado con un pixel que esté por encima del umbral superior.

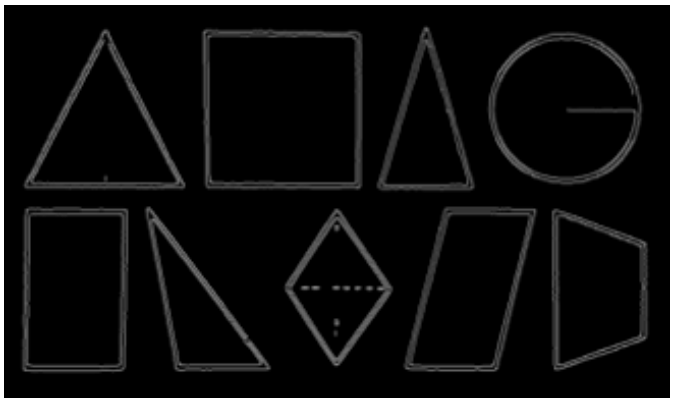


Figura 4.8: Representación de una imagen al efectuar Canny

4.1.5. Algoritmo de Ramer-Douglas-Peucker.

El algoritmo de Ramer-Douglas-Peucker es un algoritmo que reduce el número de puntos en una curva aproximada por una serie de puntos. Esto lo hace mediante el "pensamiento" de una linea entre el primer y último punto en un conjunto de puntos que forman una curva. Verifica cuál de los puntos intermedios es el más lejano de esta línea. Si el punto está más cerca que una distancia "epsilon" dada, este se remueve. Si por el otro lado este punto esta más lejos que epsilon, la curva se divide en dos partes:

1. Del primer punto al punto intermedio.

2. Del punto intermedio y los puntos restantes al punto final.

La función es repulsivamente llamada para ambas curvas. En las figuras 4.9, 4.10, 4.11 y 4.12 se muestra el proceso que sigue el algoritmo.

4.1.6. Algoritmo de Hough.

El objetivo de este algoritmo es el reconocimiento de figuras circulares en una imagen. Esta tarea puede ser dividida en diferentes procedimientos. Primero, la imagen es suavizada con el fin de reducir la cantidad de ruido. Posteriormente, una tarea de reconocimiento de bordes es implementada. Para esto se hace uso de infiltro que utiliza la segunda derivada y que en conjunto con el filtro de Sobel, nos ofrece una mejor reconocimiento de bordes. Después aplicamos la transformada de Hough sobre el borde del mapa. Al espacio (a, b) resultante se le aplica una convolución con un filtro llamado "Mexican hat" con el fin de

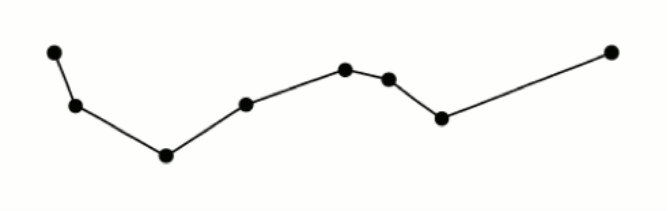


Figura 4.9: Recta inicial del algoritmo de Ramer-Douglas-Peucker

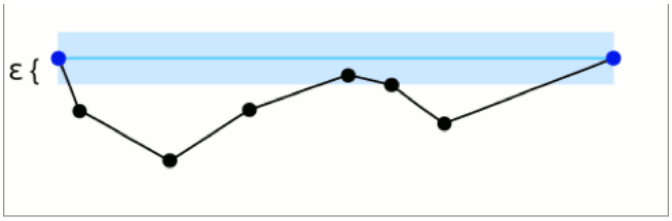


Figura 4.10: Toma de la distancia epsilon

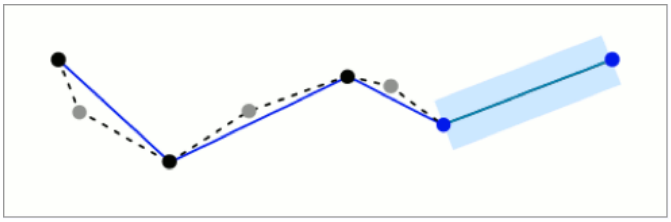


Figura 4.11: Algoritmo de Ramer-Douglas-Peucker

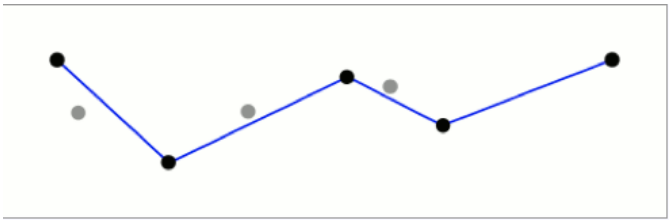


Figura 4.12: Recta final del algoritmo de Ramer-Douglas-Peucker

mejorar los puntos de interés. Posteriormente el espacio (a, b) pasa por un proceso de umbralización y los centros de los puntos de interés son encontrados. El último paso es encontrar la mayor parte de círculos observables con ayuda de los centros obtenidos. En la figura 4.13 se puede observar la transformada de Hough en uso para encontrar círculos.

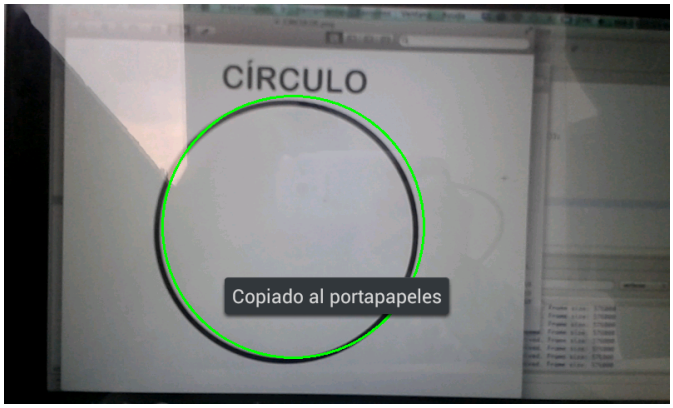


Figura 4.13: Algoritmo de Hough

4.2. Requerimientos.

En esta sección se listan los requerimientos funcionales y no funcionales correspondientes a nuestro prototipo I.

4.2.1. Requerimientos Funcionales.

- El sistema obtiene frames del entorno real mediante la cámara del dispositivo móvil en el que se ejecute.
- El sistema verifica si se reconocen figuras básicas haciendo uso de los algoritmos necesarios para el procesamiento de los frames.
- El sistema reconoce figuras básicas y las marca.
- El sistema localiza atributos de interés, dependiendo de las figuras reconocidas:
  - radio y centro (x, y) para círculos.
  - aristas para cuadriláteros y triángulos.

4.2.2. Requerimientos No Funcionales.

- La cámara del dispositivo debe tener una resolución mínima de 5 megapixeles.
- El sistema puede ejecutarse sobre un sistema móvil Android con la versión 2.3 como mínimo.
- El dispositivo móvil debe tener instalado OpenCV Manager.
- El ambiente de pruebas de este prototipo debe ser controlado.

4.3. Diagrama de Casos de Uso.

En este apartado se describen los casos de uso para el funcionamiento del prototipo I. En ellos se describe la trayectoria principal que debe seguir el proceso normalmente, de igual manera se muestran y se describen las trayectorias alternativas que puede seguir el proceso en caso de haber algún error al realizar la interacción entre el usuario y la aplicación.

La figura 4.3.1 muestra el diagrama general de casos de uso para el prototipo 1 del sistema. En él se observan los casos de uso que abarca este primer prototipo y la dependencia entre éstos y la interacción del usuario.

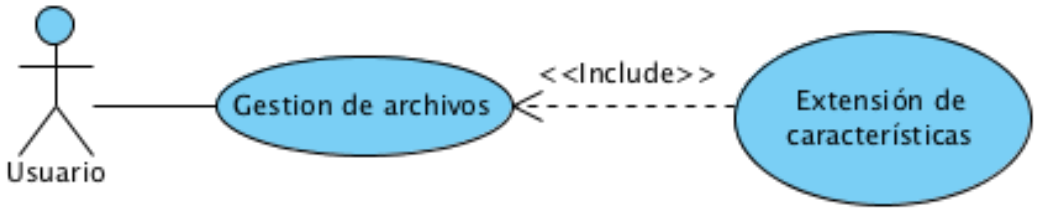


Figura 4.14: Diagrama General de Casos de Uso

4.3.1. Caso de Uso 1. Gestión de archivos

id	CU1: Gestión de archivos
Objetivo	Obtener frames del entorno al usar la cámara del dispositivo para su procesamiento.
Descripción	El sistema obtiene frames del entorno para su posterior procesamiento.
Actor	Usuario
Entradas	Imagen en tiempo real mostrada a través de la cámara del dispositivo móvil.
Salidas	Frame a procesar
Pre-condiciones	▪ La cámara del dispositivo debe estar en funcionamiento.
Post-condiciones	▪ El sistema debe obtener frames del entorno
Errores:	▪ Error al inicializar la cámara ▪ Error al obtener frames ▪ Error al no tener instalado el OpenCV Manager para el móvil.

Trayectoria principal

1. El actor inicia la aplicación.
2. El sistema inicia la cámara del dispositivo móvil. [T.A.-A]

3. El actor enfoca un objeto.
4. El sistema obtiene el frame de la imagen mostrada por la cámara. [T.A.-B]
5. Fin.

Trayectorias alternativas

- T.A.-A
  Condición: El sistema no inicia la cámara del dispositivo.
  1. El sistema muestra un mensaje de error del sistema operativo.
  2. Regresa al paso 1.
  3. Fin.
- T.A.-B
  Condición: El sistema no obtiene la imagen del entorno.
  1. Regresa al paso 3.
  2. Fin.

4.3.2. Caso de Uso 2. Extracción de características

id	CU2: Reconocimiento de las figuras básicas.
Objetivo	Extraer características por medio de los frames del entorno real y reconocer las figuras básicas que en él se encuentran.
Descripción	El sistema realiza un procesamiento a los frames obtenidos en el 4.3.1 para identificar las figuras básicas que se presenten en el entorno real.
Actor	-
Entradas	Imagen obtenidas en el 4.3.1.
Salidas	Identificación de las figuras básicas y obtención de sus atributos principales.
Pre-condiciones	▪ La aplicación debe estar en funcionamiento.
Post-condiciones	▪ El sistema debe procesar los frames. ▪ El sistema debe identificar las figuras básicas de interés. ▪ El sistema debe obtener los atributos de interés de cada una de las figuras encontradas.
Errores:	▪ Extracción inadecuada de las características. ▪ Error en el reconocimiento de las figuras ocasionadas por el ruido que pueda haber en el ambiente.
Viene de:	Caso de uso 1: Gestión de archivos.

Trayectoria principal

1. El actor da doble tap en la pantalla del dispositivo.

2. El sistema muestra el menú correspondiente de la aplicación.

3. El actor selecciona la opción para procesar los frames e identificar figuras geométricas.

4. El sistema procesa los frames obtenidos en el caso de uso 1 en el apartado 4.3.1 con el fin de identificar figuras básicas en el entorno.

5. El sistema identifica las figuras básicas de interés. [T.A.-A]

6. El sistema muestra al usuario que ha identificado las figuras de interés marcando su contorno de color verde.

7. El sistema obtiene los atributos de interés para cada una de las figuras identificadas.

8. Fin.

Trayectorias alternativas

- T.A.-A
  Condición: El sistema identifica otro tipo de figuras que no sean círculos, cuadriláteros o triángulos.

  1. Regresa al paso 4 de la trayectoria principal.

  2. Fin.

4.4. Diagrama de Actividades.

La figura 4.15 muestra el diagrama de actividades correspondiente al prototipo número uno del sistema. En él podemos observar el comportamiento del sistema con el usuario.

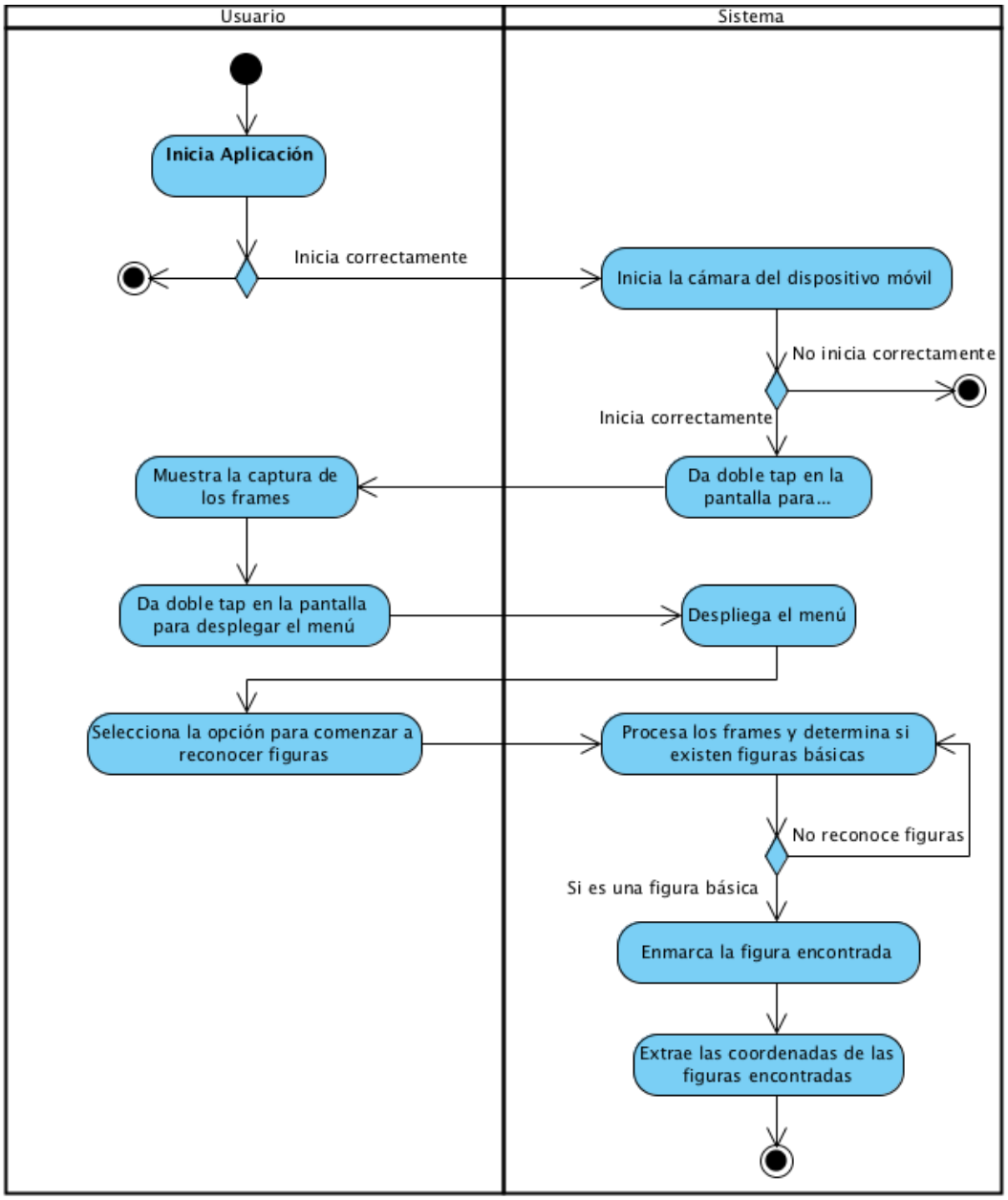


Figura 4.15: Diagrama de Actividades

4.5. Diagramas de Secuencia.

En esta sección se muestra el modelo dinámico correspondiente al prototipo 1 del sistema. Los diagramas a continuación muestran la secuencia para hacer uso de la aplicación.

4.5.1. Gestión de archivos

La figura 4.16 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 1. Gestión de archivos descrito en el apartado 4.3.1; en él, se observa como el sistema obtiene una imagen en tiempo real de la cámara del dispositivo y la usa para su posterior procesamiento.

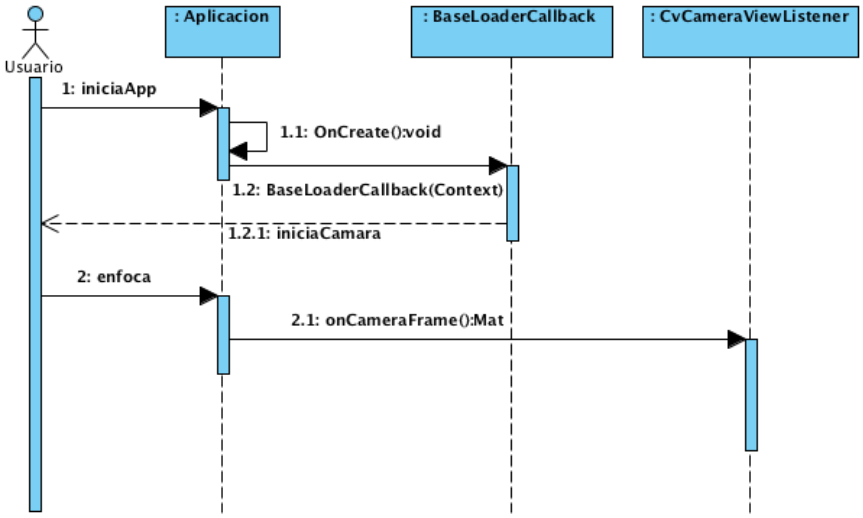


Figura 4.16: Gestión de archivos

4.5.2. Extracción de características

La figura 4.17 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 2. Extracción de características descrito en el apartado 5.3; en él, podemos observar cómo el sistema realiza una extracción de características de la imagen obtenida previamente para determinar objetos de interés en ella.

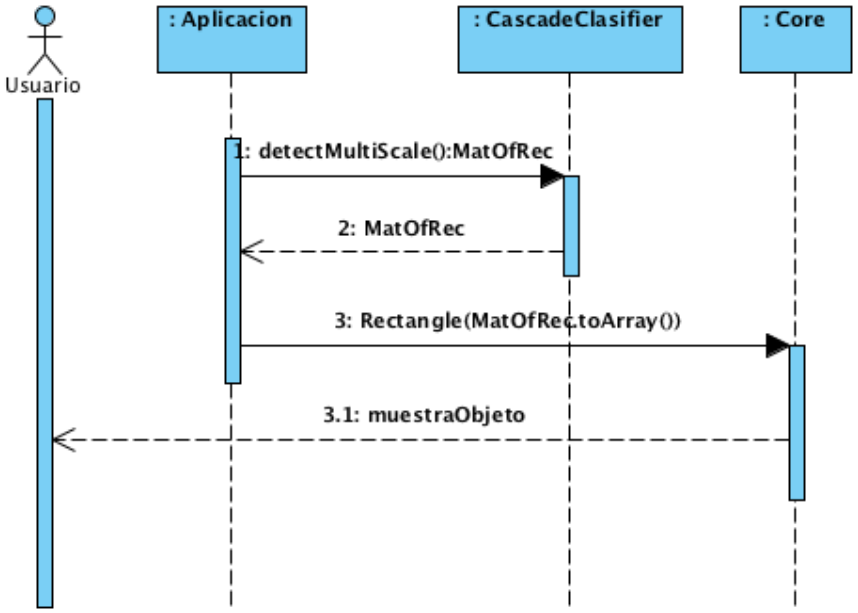


Figura 4.17: Extracción de Características

Capítulo 5

Prototipo II. Representación del Modelo.

Este capítulo expone el análisis y diseño referente al prototipo II que lleva por nombre "Representación del modelo". La función principal de este prototipo es representar en la pantalla del móvil las figuras básicas identificadas en el prototipo I, utilizando las coordenadas de interés de cada una de las figuras y haciendo uso de la realidad aumentada.

5.1. Análisis de tecnología.

Para el desarrollo del prototipo II, la API utilizada fue OpenGL ES, la cual es una versión reducida de OpenGL que utiliza Android. En esta sección se describen los aspectos principales de OpenGL ES, algunos de sus componentes y cómo funciona.

5.1.1. OpenGL ES.

OpenGL ES (OpenGL for Embedded Systems) es una variante simplificada de la API gráfica OpenGL diseñada para dispositivos integrados tales como teléfonos móviles, PDAs y consolas de videojuegos. La define y promueve el Grupo Khronos, un consorcio de empresas dedicadas a hardware y software gráfico interesadas en APIs gráficas y multimedia.

Al crear OpenGL ES 1.0, gran parte de la funcionalidad de la API OpenGL original fue eliminada. Dos de las diferencias más importantes entre OpenGL ES y OpenGL, son la eliminación de la semántica de las llamadas glBegin–glEnd para representar primitivas; y la introducción de tipos de datos en formato de punto fijo para su uso en coordenadas de vértices, lo que es útil debido a la limitada capacidad de cómputo que tienen los procesadores integrados, los cuales a menudo no disponen de una unidad aritmética en punto flotante.

Los dispositivos Android soportan diferentes versiones de la API de OpenGL ES, entre ellas están las siguientes:

- OpenGL ES 1.0 y 1.1 - Esta API es soportada a partir de Android 1.0.
- OpenGL ES 2.0 - Esta API es soportada a partir de Android 2.2 (API nivel 8).
- OpenGL ES 3.0 - Esta API es soportada a partir de Android 4.3 (API nivel 18).

Existen dos clases fundamentales en el framework de Android que nos permiten manipular y crear gráficos con la API de OpenGL ES: **GLSurfaceView** y **GLSurfaceView.Renderer**.

GLSurfaceView.

Esta clase es una vista donde puedes dibujar y manipular objetos usando las llamadas de la API de OpenGL y es muy similar con respecto a una **SurfaceView**. Se puede usar esta clase para crear una instancia de GLSurfaceView y agregar un objeto Renderer a él. Sin embargo si deseamos capturar eventos touch sobre la pantalla, debemos heredar de la clase GLSurfaceView para poder implementar los gestos touch.

GLSurfaceView.Renderer.

Esta interfaz define los métodos requeridos para dibujar los gráficos en una GLSurfaceView. Se tiene que proporcionar una implementación de esta interfaz en una clase separada a nuestra GLSurfaceView y mandar a llamar al Renderer usando la función GLSurfaceView.setRenderer().

La interfaz GLSurfaceView.Renderer requiere de la implementación de los siguientes métodos:

- onSurfaceCreated(): El sistema manda a llamar sólo una vez a este método, esto es cuando se crea el objeto GLSurfaceView. Este método es usado para realizar acciones que necesitan pasar sólo una vez, tales como inicializar objetos gráficos de OpenGL o establecer parámetros de entorno.
- onDrawFrame(): Este método es llamado por el sistema cada que se redibuja sobre el objeto GLSurfaceView. Este método es el que se usa principalmente para dibujar y redibujar los objetos.
- onSurfaceChanged(): Este método es llamado por el sistema cuando la geometría del objeto GLSurfaceView cambia, incluyendo cambios en el tamaño o en la orientación del dispositivo móvil.

5.1.2. Mapeo de coordenadas para dibujar objetos.

Uno de los problemas principales al desplegar gráficos sobre dispositivos Android es que sus pantallas pueden variar de tamaño y forma. OpenGL asume un cuadrado, un sistema de coordenadas uniforme, por defecto, dibuja esas coordenadas sobre pantallas que no son normalmente cuadradas como si fueran cuadradas.

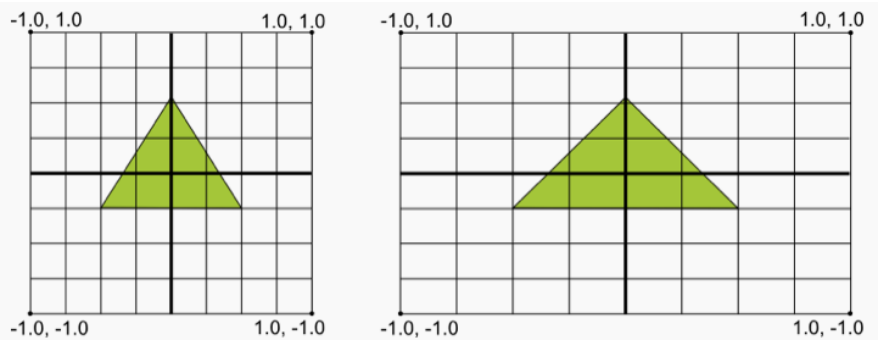


Figura 5.1: Sistema de coordenadas de OpenGL

La figura de arriba muestra del lado izquierdo el sistema de coordenadas uniforme que adopta un marco de OpenGL, y cómo esas coordenadas se mapean a una pantalla de un dispositivo en orientación horizontal del lado derecho.

5.1.3. Caras y devanado de las figuras.

En OpenGL, la cara de una figura es una superficie definida por tres o más puntos en un espacio de tres dimensiones. El conjunto de tres o más puntos tridimensionales tienen una cara frontal y una cara posterior. Para poder diferenciar esto, se hace uso de la dirección en la cual se definen los puntos o vértices de una figura.

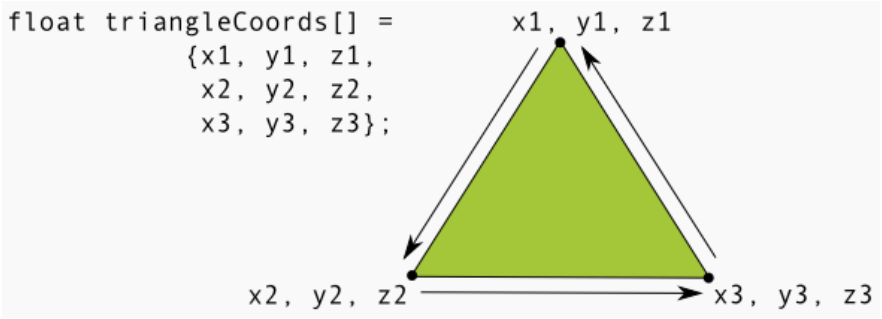


Figura 5.2: Modo en el que se dibuja la cara de un triángulo

En la figura 5.2 se puede observar que los puntos del triángulo están definidos en un orden tal que son dibujados en dirección al sentido opuesto a las manecillas del reloj. El orden en el que estas coordenadas son dibujadas definen la dirección de devanado de la figura. Por defecto, en OpenGL, la cara que es dibujada en sentido opuesto a las manecillas del reloj es la cara frontal.

5.2. Requerimientos.

En esta sección se listarán los requerimientos funcionales y no funcionales que corresponden al prototipo II.

5.2.1. Requerimientos Funcionales.

- El sistema genera un modelo de realidad aumentada coherente con las figuras reconocidas en el prototipo anterior y será mostrado al usuario.
- El sistema genera el modelo con base en la figura reconocida en el prototipo 1.

5.2.2. Requerimientos No Funcionales.

- El procesador del dispositivo móvil debe ser al menos de dos núcleos para un buen desempeño del sistema.
- El sistema no hará uso de marcadores para representar los modelos de realidad aumentada.
- El ambiente donde se encuentra el usuario debe ser controlado.

5.3. Diagrama de Casos de Uso.

En este apartado se muestra el caso de uso correspondiente al prototipo II. En él se especifica la trayectoria principal que sigue el proceso normalmente, así como las posibles trayectorias alternativas que pueda tener.

La figura 5.3 muestra el diagrama general de casos de uso abarcando los dos primero prototipos. En él se observa el caso de uso agregado del prototipo II y la dependencia que tiene con los casos de uso del prototipo I.

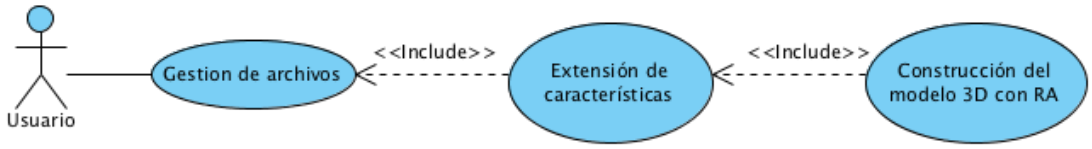


Figura 5.3: Diagrama General de Casos de Uso

5.3.1. Caso de Uso 3. Construcción de modelo 3D con realidad aumentada.

id	CU3: Construcción de Realidad Aumentada.
Objetivo	Sobreponer en la pantalla del móvil el objeto de Realidad aumentada, según la figura que haya detectado.
Descripción	El sistema sobrepondrá un objeto en 3D, ya sea una esfera, un cubo o una pirámide según sea el caso de la figura detectada.
Actor	-
Entradas	Un objeto de la clase figure.
Salidas	El objeto en 3D.
Pre-condiciones	▪ El sistema debió haber reconocido alguna figura.
Post-condiciones	▪ El sistema deberá mostrar un objeto de RA en 3D según la figura que haya detectado. ▪ El sistema seguirá mostrando el objeto, aunque la figura detectada ya no se esté enfocando.
Errores:	▪ El dispositivo no soporta el SDK de Vuforia.
Viene de:	Caso de Uso 2. Extracción de características

Trayectoria principal.

1. El sistema inicia la actividad.
2. El sistema recibe un objeto de la clase figure.
3. El sistema configura la librería de Realidad Aumentada (QCAR) con la versión de OpenGL.

4. El sistema inicializa la librería de Realidad Aumentada de forma asíncrona. [T.A.-A]

5. El sistema inicializa el renderizado.

6. El sistema trae al frente la capa donde se pintarán los objetos de Realidad Aumentada.

7. El sistema activa la bandera de "estado de listo".

8. El sistema pregunta asíncronamente, si la aplicación está en "estado de listo".

9. El sistema crea el objeto de Realidad Aumentada y lo sobrepone en la pantalla.

10. Fin.

Trayectorias alternativas.

- T.A.-A
  Condición: El dispositivo no soporta Vuforia.

  1. El sistema muestra una alerta de que el dispositivo no soporta Vuforia.
  2. El actor presiona el botón de cerrar.
  3. El sistema cierra la aplicación.
  4. Fin

5.4. Diagrama de Actividades.

La figura 5.4 muestra el diagrama de actividades correspondiente al prototipo número dos del sistema. En él podemos observar el comportamiento que tiene el sistema al crear las figuras en 3D con realidad aumentada.

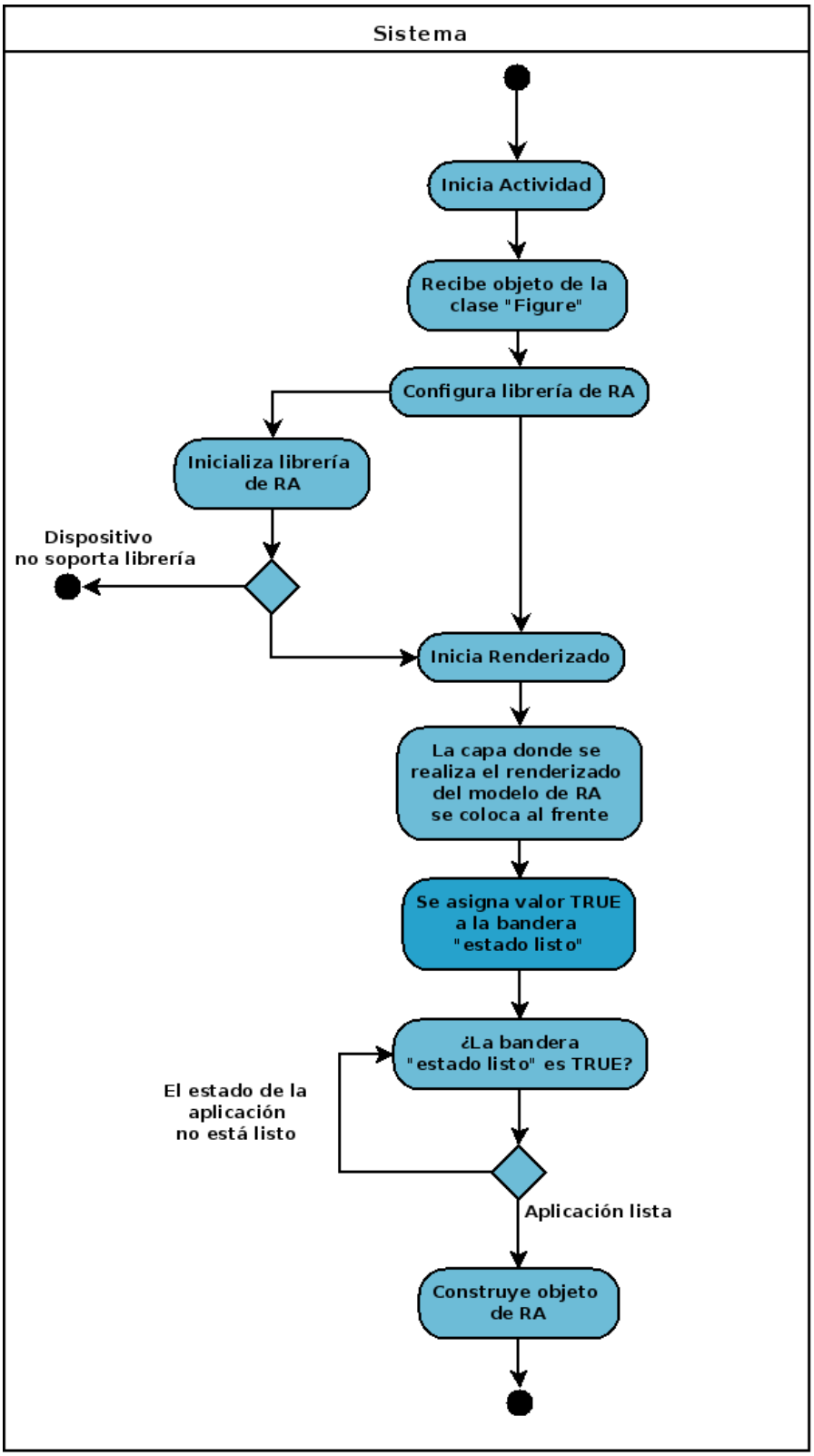


Figura 5.4: Diagrama de Actividades

5.5. Diagramas de Secuencia.

En esta sección se muestra el modelo dinámico correspondiente al prototipo 2 del sistema. El diagrama a continuación muestra la secuencia para hacer uso de la aplicación.

5.5.1. Construcción de modelo 3D con realidad aumentada.

La figura 5.5 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 3. Construcción de modelo 3D con realidad aumentada descrito en el apartado 6.1; en él, se observa cómo el sistema realiza el modelado de las figuras reconocidas en 3D con realidad aumentada.

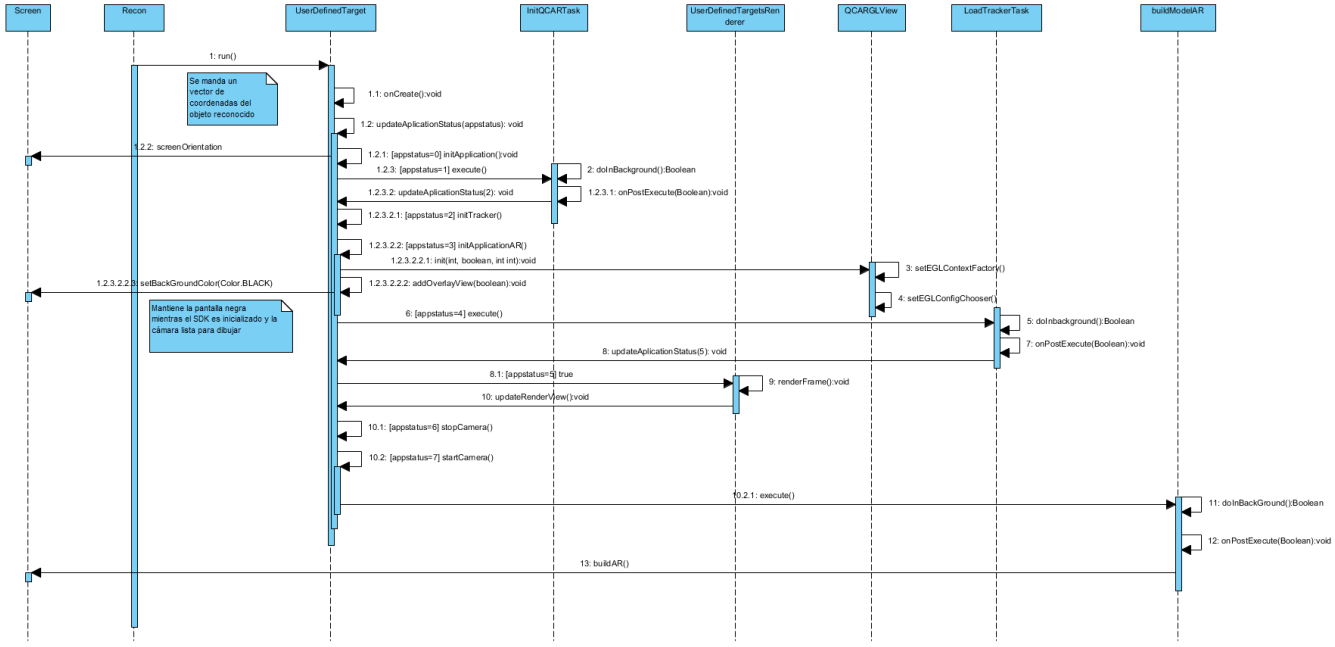


Figura 5.5: Construcción de modelo 3D con realidad aumentada.

Capítulo 6

Prototipo III. Manipulación del Modelo.

Este capítulo presenta el último prototipo del Trabajo Terminal, el cual tiene como función principal realizar la manipulación del modelo 3D generado en el prototipo II. Este prototipo constará de tres gestos diferentes que realizarán las operaciones básicas de rotación, traslación y escalamiento del modelo.

6.1. Requerimientos.

En esta sección se listan los requerimientos funcionales y no funcionales que corresponden al prototipo III.

6.1.1. Requerimientos Funcionales.

- El sistema puede realizar operaciones básicas de traslación, rotación y escalamiento sobre el modelo 3D generado en el entorno virtual.
- El sistema tiene la capacidad de alterar el modelo según la operación que el usuario le solicite.
- El sistema entiende qué tipo de operación básica realizar gracias al manejo de diferentes gestos.

6.1.2. Requerimientos No Funcionales.

- El dispositivo móvil debe soportar la versión utilizada de OpenGL.
- El ambiente donde se encuentra el usuario debe ser controlado.
- El sistema debe poseer una pantalla táctil capacitiva.

6.2. Diagrama de Casos de Uso.

Este apartado muestra el caso de uso correspondiente al prototipo III. Aquí se describe la trayectoria principal que debe sigue el proceso, así como cada una de las trayectorias alternativas que pueda tener.

La figura 6.1 muestra el diagrama general de casos de uso final. En él se observa todos los casos de uso con los que interactúa el usuario y la relación que tienen cada uno entre ellos.

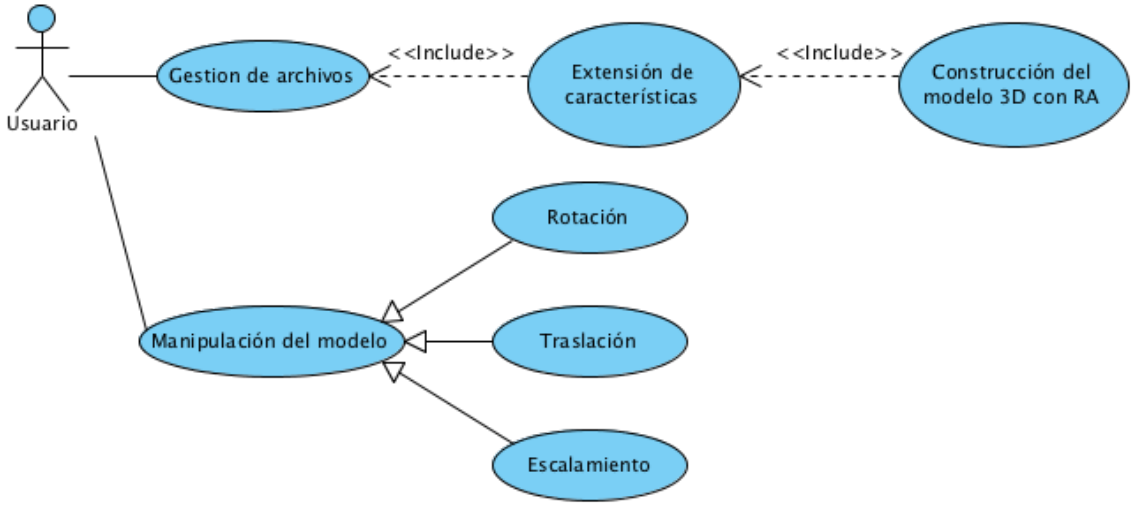


Figura 6.1: Diagrama General de Casos de Uso

6.2.1. Caso de Uso 4. Rotación del modelo.

id	CU4: Rotación del modelo.
Objetivo	Crear un gesto de rotación para el modelo representado en 3D.
Descripción	El sistema rota el modelo 3D interpretando el gesto generado por el actor.
Actor	Usuario
Entradas	▪ El modelo 3D en la pantalla del dispositivo. ▪ El gesto de rotación introducido por el actor.
Salidas	El objeto 3D modificado después de realizar la rotación.
Pre-condiciones	▪ El sistema debió haber presentado el modelo de alguna figura.
Post-condiciones	▪ El sistema debe modificar el modelo de RA en 3D según el gesto que realice el actor.
Errores:	▪ El dispositivo no soporta el SDK de Realidad Aumentada ▪ El sistema no soporta la versión de OpenGL.

Trayectoria principal.

1. El actor realiza uno de los gestos correspondiente a las tres operaciones básicas.
2. El sistema verifica si el gesto corresponde a una rotación. [T.A.-A] [T.A.-B] [T.A.-C]

3. El sistema hace la rotación correspondiente al modelo 3D.

4. El sistema muestra el modelo final en la pantalla.

5. Fin.

Trayectorias alternativas.

- T.A.-A
  Condición: El sistema no identifica un gesto válido.

  1. El sistema no reacciona al gesto realizado.
  2. Regresa al paso 1 de la trayectoria principal.
  3. Fin

- T.A.-B
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al de la traslación del modelo.

  1. Ir al paso 2 de la trayectoria principal del caso de uso 5 correspondiente a 6.2.2.
  2. Fin

- T.A.-C
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al del escalamiento del modelo.

  1. Ir al paso 2 de la trayectoria principal del caso de uso 6 correspondiente a 6.2.3.
  2. Fin

6.2.2. Caso de Uso 5. Traslación del modelo.

id	CU4: Traslación del modelo.
Objetivo	Crear un gesto de traslación para el modelo representado en 3D.
Descripción	El sistema traslada o mueve el modelo 3D interpretando el gesto generado por el actor.
Actor	Usuario
Entradas	▪ El modelo 3D en la pantalla del dispositivo. ▪ El gesto de traslación introducido por el actor.
Salidas	El objeto 3D modificado después de realizar la traslación.
Pre-condiciones	▪ El sistema debió haber presentado el modelo de alguna figura.
Post-condiciones	▪ El sistema debe modificar el modelo de RA en 3D según el gesto que realice el actor.
Errores:	▪ El dispositivo no soporta el SDK de Realidad Aumentada ▪ El sistema no soporta la versión de OpenGL.

Trayectoria principal.

1. El actor realiza uno de los gestos correspondiente a las tres operaciones básicas.
2. El sistema verifica si el gesto corresponde a una traslación. [T.A.-A] [T.A.-B] [T.A.-C]
3. El sistema hace la traslación correspondiente al modelo 3D.
4. El sistema muestra el modelo final en la pantalla.
5. Fin.

Trayectorias alternativas.

- T.A.-A
  Condición: El sistema no identifica un gesto válido.
  1. El sistema no reacciona al gesto realizado.
  2. Regresa al paso 1 de la trayectoria principal.
  3. Fin
- T.A.-B
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al de la rotación del modelo.

1. Ir al paso 2 de la trayectoria principal del caso de uso 4 correspondiente a 6.2.1.
2. Fin

- T.A.-C
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al del escalamiento del modelo.

1. Ir al paso 2 de la trayectoria principal del caso de uso 6 correspondiente a 6.2.3.
2. Fin

6.2.3. Caso de Uso 6. Escalamiento del modelo.

id	CU4: Escalamiento del modelo.
Objetivo	Crear un gesto de escalamiento para el modelo representado en 3D.
Descripción	El sistema hace más grande o más pequeño el modelo 3D interpretando el gesto generado por el actor.
Actor	Usuario
Entradas	▪ El modelo 3D en la pantalla del dispositivo. ▪ El gesto de escalamiento introducido por el actor.
Salidas	El objeto 3D modificado después de realizar el escalamiento.
Pre-condiciones	▪ El sistema debió haber presentado el modelo de alguna figura.
Post-condiciones	▪ El sistema debe modificar el modelo de RA en 3D según el gesto que realice el actor.
Errores:	▪ El dispositivo no soporta el SDK de Realidad Aumentada ▪ El sistema no soporta la versión de OpenGL.

Trayectoria principal.

1. El actor realiza uno de los gestos correspondiente a las tres operaciones básicas.
2. El sistema verifica si el gesto corresponde al de escalamiento. [T.A.-A] [T.A.-B] [T.A.-C]
3. El sistema hace la operación de escalamiento al modelo 3D.
4. El sistema muestra el modelo final en la pantalla.
5. Fin.

Trayectorias alternativas.

- T.A.-A
  Condición: El sistema no identifica un gesto válido.

  1. El sistema no reacciona al gesto realizado.
  2. Regresa al paso 1 de la trayectoria principal.
  3. Fin

- T.A.-B
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al de la rotación del modelo.

  1. Ir al paso 2 de la trayectoria principal del caso de uso 4 correspondiente a 6.2.1.
  2. Fin

- T.A.-C
  Condición: El sistema identifica un gesto válido pero el gesto corresponde al de la traslación del modelo.

  1. Ir al paso 2 de la trayectoria principal del caso de uso 5 correspondiente a 6.2.2.
  2. Fin

6.3. Diagrama de Clases.

La figura 6.2 muestra el diagrama de clases del sistema y la relación entre cada que tiene cada clase con las demás. En él se pueden observar la clase SplashScreen que pertenece a la pantalla de bienvenida, la clase Reconocimiento que es donde se realiza el procesamiento de los frames y la clase UserDefinedTargets que contiene la representación del modelo en 3D con realidad aumentada. De ésta última clase se desprenden varias clases para realizar la inicialización de objetos que son útiles para cargar el entorno.

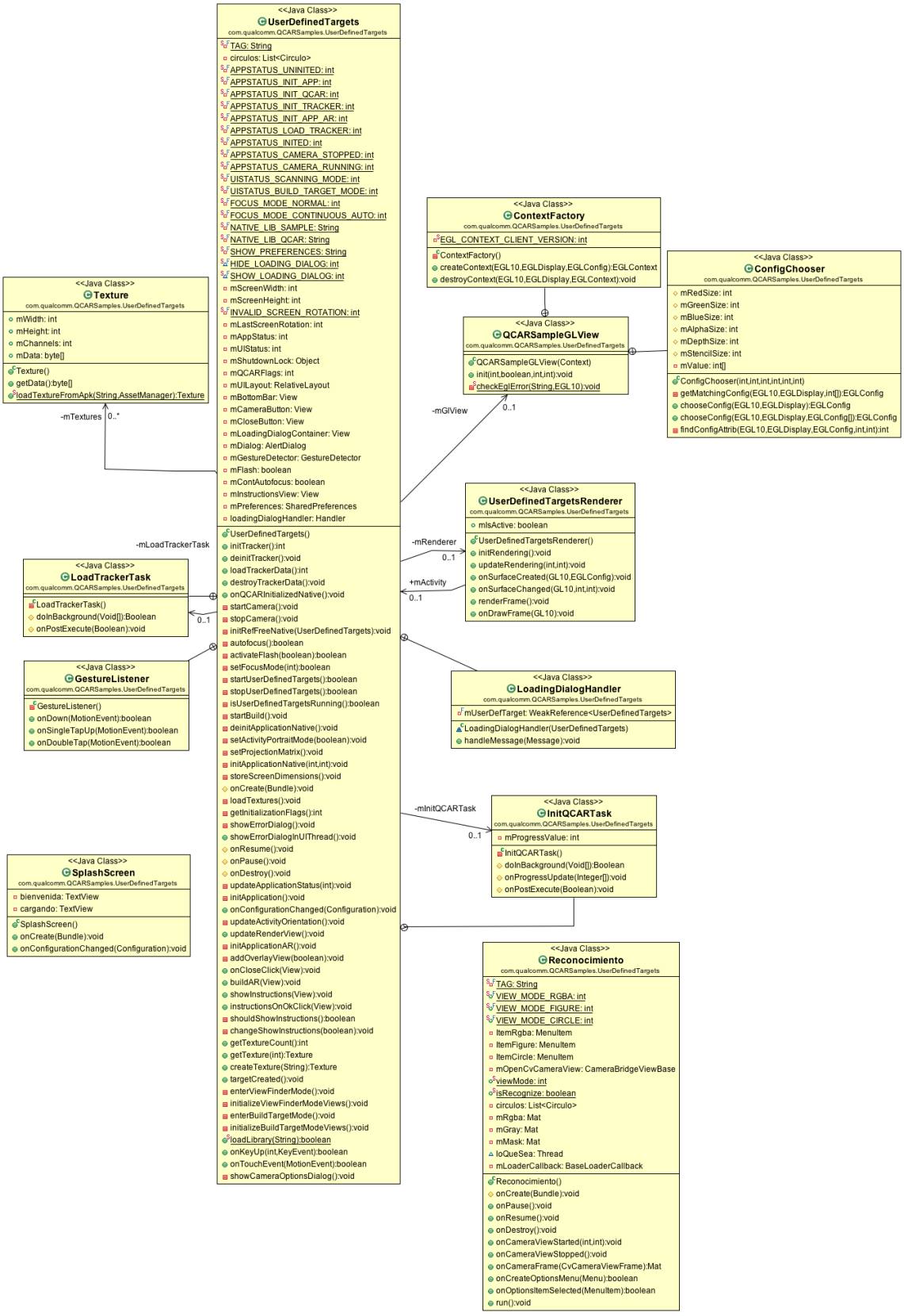


Figura 6.2: Diagrama de Clases

6.4. Diagrama de Actividades.

En esta sección se muestran el diagrama de actividades correspondiente a los casos de uso del prototipo III, el cual maneja la manipulación del modelo por medio de los gestos realizados por el usuario.

La figura 6.3 muestra el diagrama de actividades correspondiente al caso de uso 3, 4 y 5 descritos en la sección anterior. En él podemos observar el comportamiento que tiene el sistema al realizar alguna de las tres operaciones básicas para manipular el modelo 3D que haya escogido el usuario.

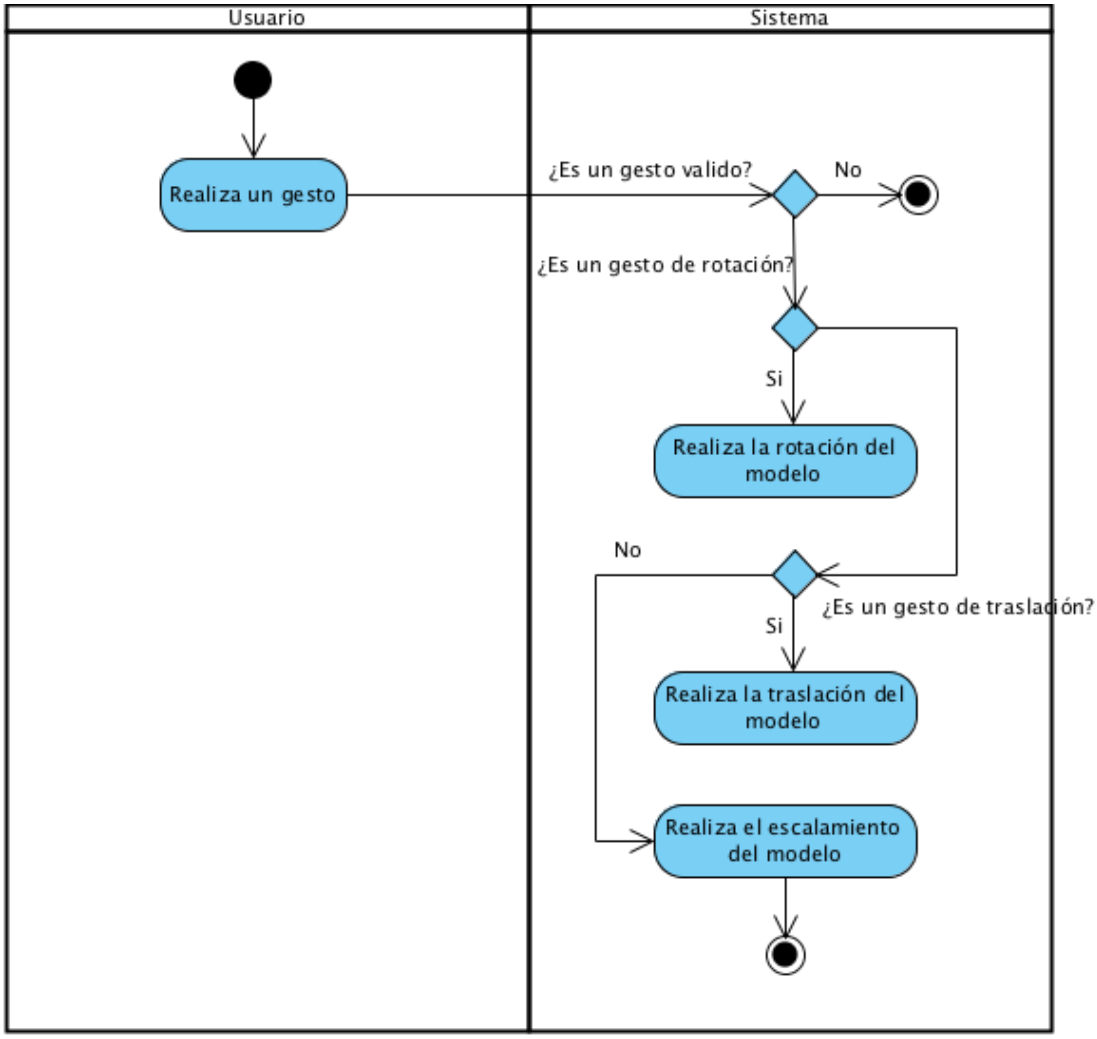


Figura 6.3: Diagrama de Actividades

6.5. Diagramas de Secuencia.

En esta sección se muestra el modelo dinámico correspondiente al prototipo 2 del sistema. El diagrama a continuación muestra la secuencia para hacer uso de la aplicación.

6.5.1. Rotación del modelo.

La figura 6.4 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 4 de nombre Rotación del modelo descrito en el apartado 6.2.1; en él, se observa cómo el sistema

realiza la operación de rotación sobre el modelo 3D generado cuando el usuario realiza un gesto sobre la pantalla.

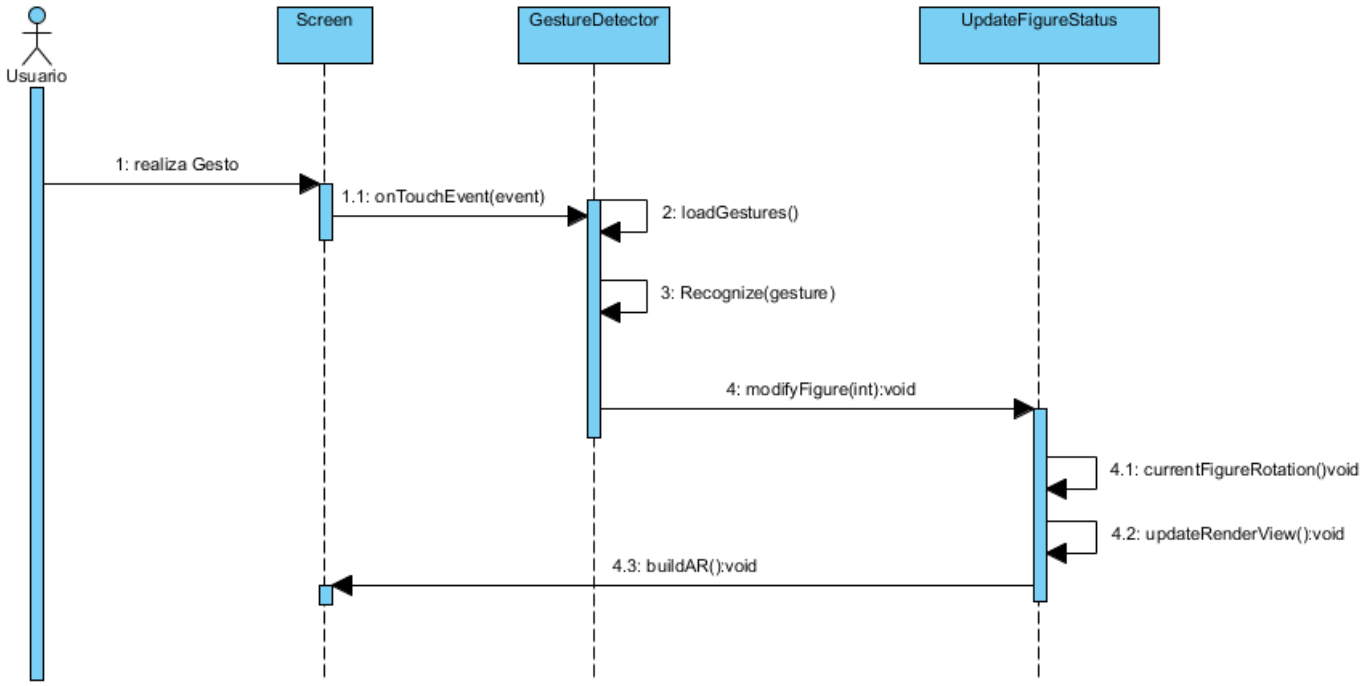


Figura 6.4: Rotación del modelo.

6.5.2. Traslación del modelo.

La figura 6.5 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 5 de nombre Traslación del modelo descrito en el apartado 6.2.2; en él, se observa cómo el sistema realiza la operación de traslación sobre el modelo 3D lanzado cuando el usuario realiza un gesto sobre la pantalla.

6.5.3. Escalamiento del modelo.

La figura 6.6 muestra el diagrama de secuencia correspondiente al caso correcto simple del Caso de Uso 6 de nombre Escalamiento del modelo descrito en el apartado 6.2.3; en él, se observa cómo el sistema realiza la operación de rotación sobre el modelo 3D lanzado cuando el usuario realiza un gesto sobre la pantalla.

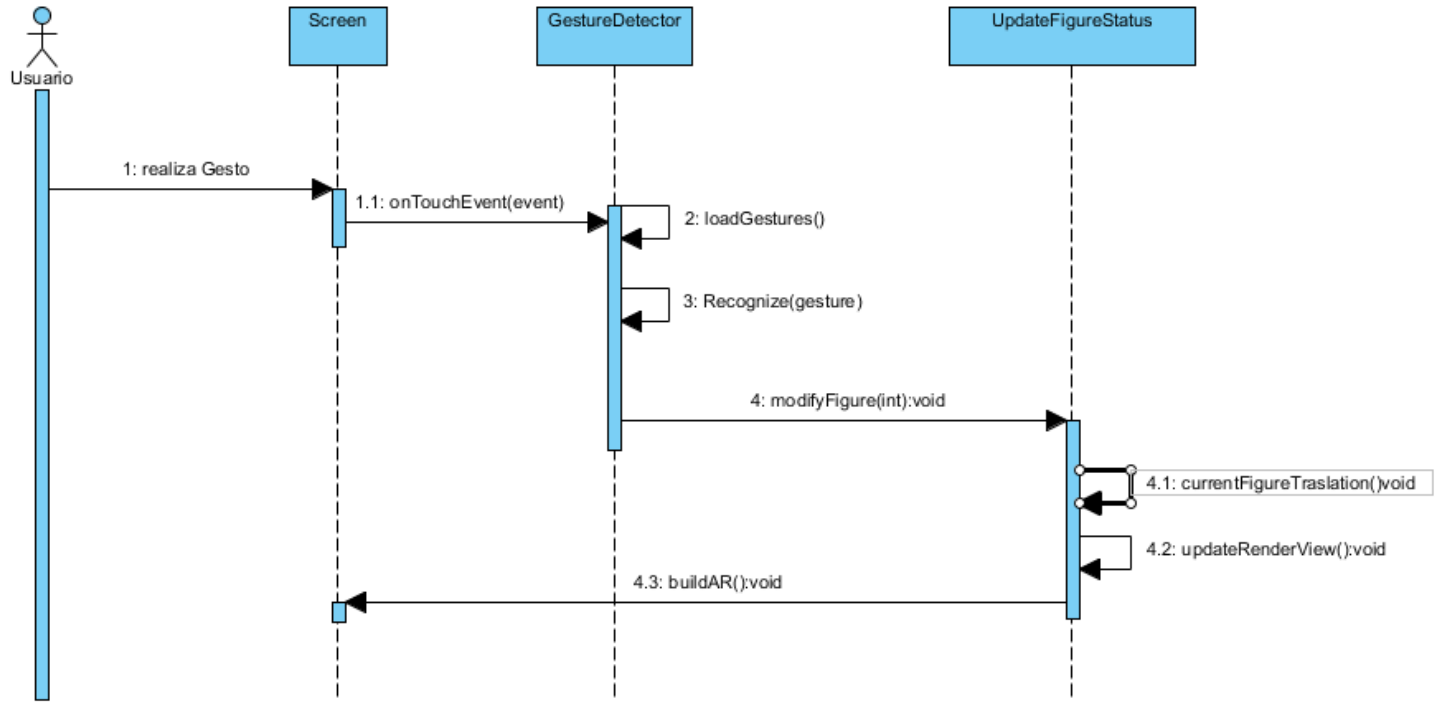


Figura 6.5: Traslación del modelo.

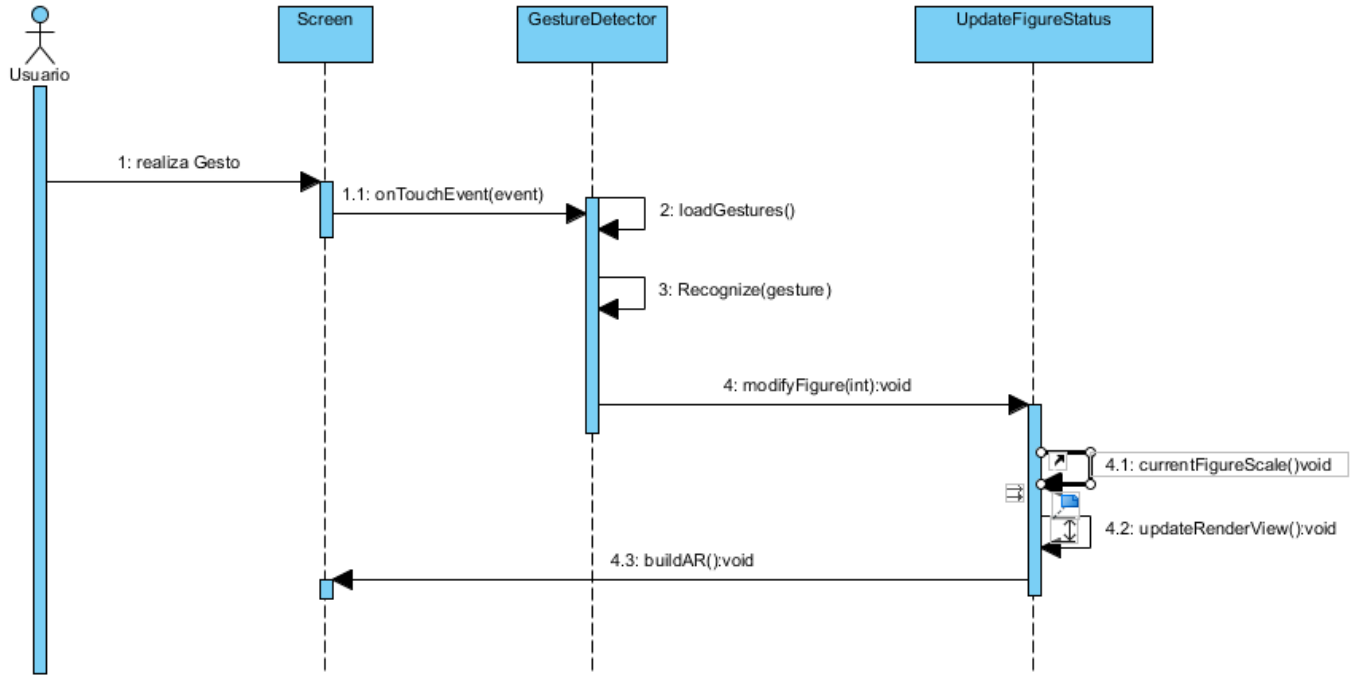


Figura 6.6: Escalamiento del modelo.

Capítulo 7

Pruebas y resultados.

7.1. Pruebas del prototipo I.

7.1.1. Entorno controlado

La aplicación esta diseñada para trabajar en un entorno controlado, para obtener los resultados esperados disminuyendo la probabilidad de encontrar falsos positivos en el momento del reconocimiento de las figuras básicas.

Los factores externos que tomamos en cuenta fueron los siguientes:

La intensidad luminosa

Se tomó un rango entre 500 a 700 luxes dentro de la habitación, esto nos ayuda a no tener pérdida de información al reconocer la figura, ya que está bien iluminada y se pueden percibir todos sus bordes sin problema, y de igual manera con el rango que consideramos no hay exceso de luz sobre las figuras que haga que no se distingan los bordes.

La distancia del dispositivo móvil a la figura

La distancia que se ocupó para reconocer la figura cambia dependiendo al tamaño de esta, sin embargo se debe de tomar la distancia necesaria para solo enfocar la figura de interés para evitar reconocer falsos positivos.

Tamaño de los bordes de la imagen

Se manejó un nivel mínimo del valor de píxeles en los bordes, para garantizar el reconocimiento correcto de la figura, utilizamos 20 píxeles de grosor sobre los bordes de la imagen, en caso de que se reconozca una imagen que se encuentre sin rellenar, en el caso de reconocer una figura con volumen se discrimina el fondo de este para obtener el objeto de interés.

7.1.2. Pruebas

Para el prototipo I (Detección de figuras), se realizarón pruebas con diferentes conjuntos de imágenes, en las condiciones mensionadas anteriormente. El proceso para realizar las pruebas fue el siguiente.

1. Se inicia la aplicación.

2. El dispositivo móvil se coloca de tal manera que la cámara de este enfoque la figura y muestre sobre la pantalla del móvil la figura que se va a reconocer.

3. Con doble tap sobre la pantalla del móvil, se abre el menú y se selecciona la opción de “Reconocimiento” (para triángulos y cuadriláteros) o la opcion “Reconocimiento de círculos”.

4. Cuando se haya reconocido una figura básica, el Log de la aplicación imprime que figura detectó (cuadrilpatero, círculo o tríangulo).

Nota: *Falsos Positivos* son aquellas figuras que el sistema clasifica como triángulos y cuadriláteros sin ser estás figuras, o aquellas que no fueron enmarcadas en su totalidad y recibieron esta clasificación. *Positivos* son aquellas figuras que fueron bien clasificadas y enmarcadas sus aristas.

Los resultados se muestran en las siguientes tablas:

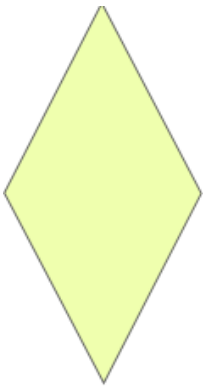
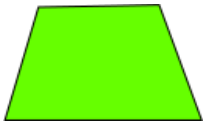
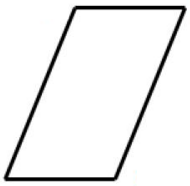
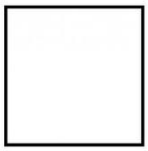
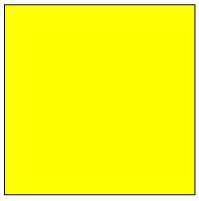
Figura	Resultado	Observaciones
	Positivo	▪ El sistema enmarcó las cuatro aristas de la figura. ▪ El sistema clasificó como un cuadrilátero.
	Falso Positivo	▪ El sistema solo enmarcó, dos aristas de los bordes superiores. ▪ Sin embargo el número de aproximación de curvas fue de cuatro. ▪ La figura se clasificó como cuadrilátero.
	Falso Positivo	▪ El sistema enmarcó sólo dos aristas de cuatro. ▪ El número de aproximación de curvas fue de cuatro. ▪ El sistema clasificó la figura como un cuadrilátero.

Figura	Resultado	Observaciones
	Positivo	▪ El sistema enmarcó las cuatro aristas de la figura. ▪ El sistema clasificó la figura como cuadrilátero.
	Falso Positivo	▪ El sistema enmarcó completamente, solo tres aristas superiores de la figura. ▪ La arista inferior no fue remarcada completamente. ▪ El número de aproximación de curvas fue de cuatro. ▪ La figura se clasificó como cuadrilpatero.
	Positivo	▪ Las cuatro aristas fueron enmarcadas. ▪ El número de aproximación de curvas de fue cuatro. ▪ El sistema clasificó la figura como cuadrilátero.
	Positivo	▪ Las cuatro aristas fueron enmarcadas. ▪ El número de aproximación de curvas fue de cuatro. ▪ El sistema clasificó la figura como cuadrilátero.

Figura	Resultado	Observaciones
	Positivo	▪ Las cuatro aristas fueron enmarcadas. ▪ El número de aproximación de curvas fue de cuatro. ▪ El sistema clasificó la figura como cuadrilátero.
	Positivo	▪ Las cuatro aristas fueron enmarcadas. ▪ El número de aproximación de curvas fue de cuatro. ▪ El sistema clasificó la figura como cuadrilátero.
	Positivo	▪ Las cuatro aristas fueron enmarcadas. ▪ El número de aproximación de curvas fue de cuatro. ▪ El sistema clasificó la figura como cuadrilátero.

Cuadro 7.1: Resultados de detección de cuadriláteros

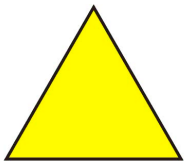
Figura	Resultado	Observaciones
	Positivo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas fue de tres. ▪ El sistema clasificó la figura como triángulo.

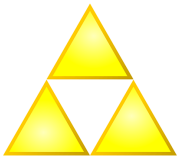
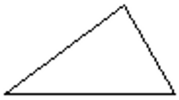
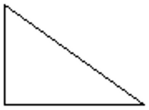
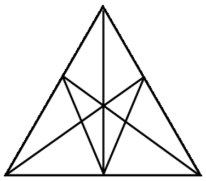
Figura	Resultado	Observaciones
	Positivo	▪ Fueron enmarcados tres triángulos ▪ El sistema clasificó las figuras como triángulos. ▪ El Log del sistema imprimió que reconocio tres triángulos diferentes.
	Falso Positivo	▪ El sistema sólo enmarcó dos de las tres aristas. ▪ El número de aproximación de curvas de cuatro. ▪ El sistema clasificó la figura como cuadrilátero.
	Positivo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas de tres. ▪ El sistema clasificó la figura como triángulo.
	Positvo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas de tres. ▪ El sistema clasificó la figura como triángulo.
	Positvo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas de tres. ▪ El sistema clasificó la figura como triángulo.

Figura	Resultado	Observaciones
	Falso Positvo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas de tres. ▪ El sistema clasificó la figura como triángulo.
	Falso Positvo	▪ Hay dos triángulos de los cuales sólo enmarcó uno, y una parte del otro. ▪ El segundo marco que dibujó la aplicación fue clasificada como un cuadrilátero. ▪ El sistema clasificó el primer borde como triángulo.
	Positvo	▪ Las tres aristas fueron enmarcadas. ▪ El número de aproximación de curvas de tres. ▪ El sistema clasificó la figura como triángulo.

Cuadro 7.2: Resultados de detección de triángulos

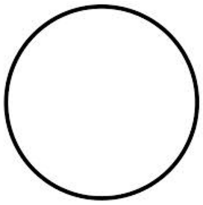
Figura	Resultado	Observaciones
	Positivo	▪ El sistema reconoció la circunferencia. ▪ El sistema se traba un poco si no se ha vizualizado en la pantalla del móvil la circunferencia a reconocer.

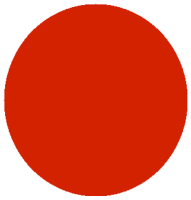
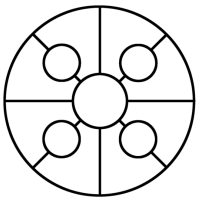
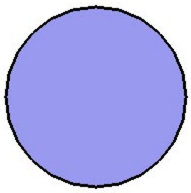
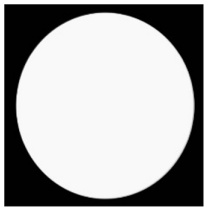
Figura	Resultado	Observaciones
	Positivo	▪ El sistema reconoció una de las tres circunferencias. ▪ Después de reconocer la primer circunferencia si se sigue vizualizando la figura reconoce el círculo que está mas al exterior de le a figura.
	Positivo	▪ El sistema reconoció la circunferencia.
	Positivo	▪ El sistema reconoció la circunferencia.
	Falso Positivo	▪ El sistema no reconoció la circunferencia. ▪ La figura se seguía vizualizando en la pantalla del móvil, después de 40 segundos, el sistema enmarcó círculos en toda la pantalla provocadas por el ruido en la imágen (iluminación).

Figura	Resultado	Observaciones
	Positivo	▪ El sistema reconoció la circunferencia. ▪ Detecó el círculo en 3 segundos, dado que el borde de la circunferencia es muy contrastante.
	Falso Positivo	▪ El sistema reconoció la circunferencia mas grande y la que está mas al centro. ▪ El sistema enmarcó 3 circunferencias que pasaban que tocaban en forma tangencial los bordes de los círculos más pequeños.
	Positivo	▪ El sistema reconoció la circunferencia.
	Positivo	▪ El sistema reconoció la circunferencia.
	Positivo	▪ El sistema reconoció la circunferencia.

Cuadro 7.3: Resultados de detección de círculos

La figura 7.1 muestra la pantalla de inicio del sistema donde se cargan todos los componentes necesarios para pasar a la actividad principal.

Figura 7.1: Pantalla de inicio.

Al cargar la siguiente pantalla, el sistema muestra una pantalla con las instrucciones de cómo usar el sistema (figura 7.2).

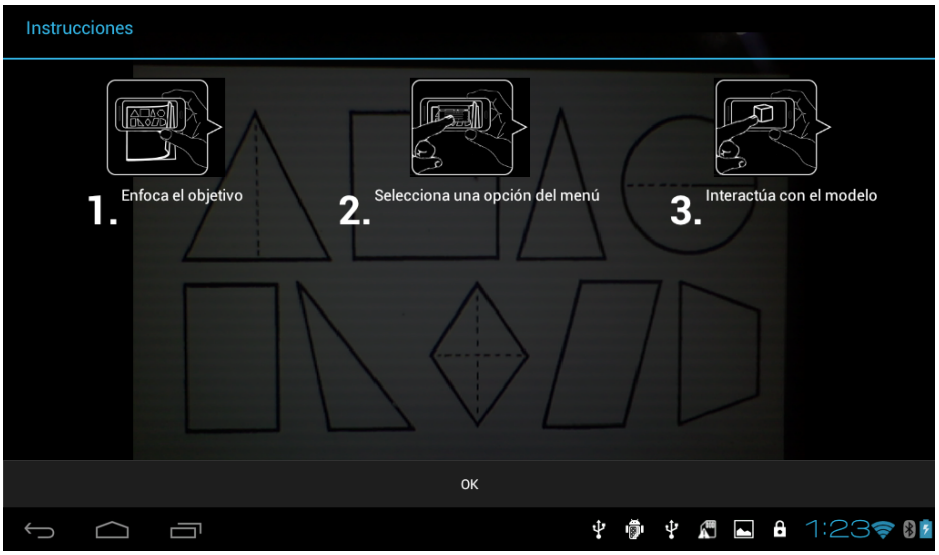


Figura 7.2: Pantalla con las instrucciones de uso

Una vez mostradas las instrucciones, se muestra la pantalla principal de la aplicación, la cual carga la cámara del dispositivo móvil. A continuación damos doble tapping en la pantalla del dispositivo y el sistema nos mostrará el menú de la figura 7.3. Para esta prueba el usuario seleccionó la opción "Reconocimiento", la cual permite hacer el análisis de los frames y reconocer figuras como triángulos y cuadriláteros.

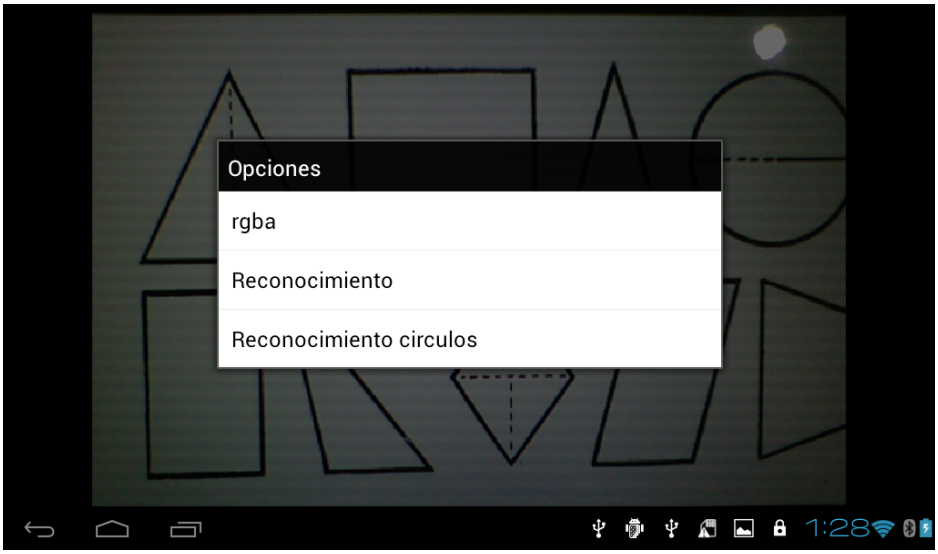


Figura 7.3: Pantalla que muestra el menú

Una vez comenzado el análisis de los frames, se obtuvieron diferentes resultados, una de las pruebas que mejores resultados arrojó es la que se muestra en la figura 7.4, haciendo el reconocimiento de 4 de 7 figuras básicas.

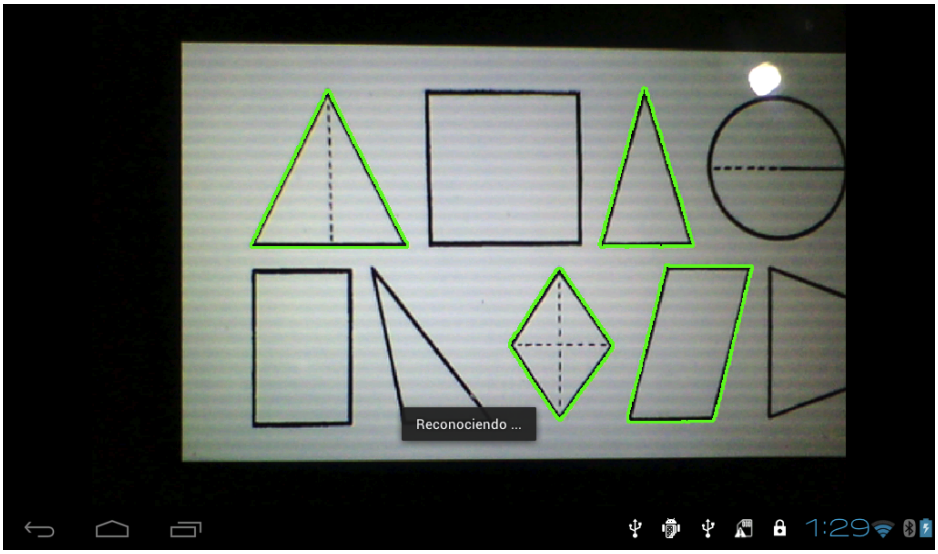


Figura 7.4: Reconocimiento de las figuras.

En la figura 7.5 se muestra el resultado de la prueba para la opción de Reconocimiento de círculos.

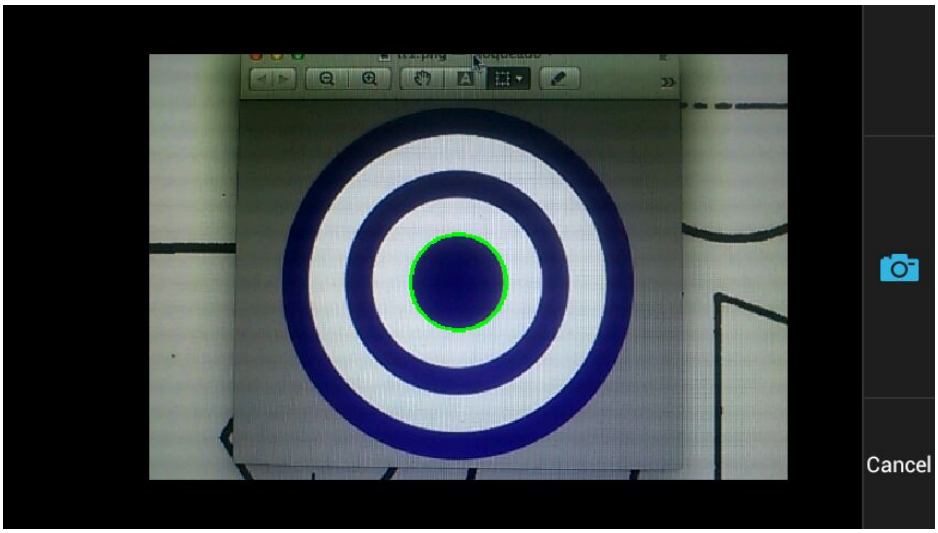


Figura 7.5: Reconocimiento de círculos.

A continuación se muestran otros resultados obtenidos durante las pruebas del prototipo 1:

7.2. Pruebas del prototipo II.

Las pruebas referentes al prototipo II representación del modelo, muestran un objeto 3D con ayuda de realidad aumentada sobre los frames previamente procesados. Esta parte aún se encuentra en pruebas, pero en las siguientes imágenes se muestran algunos de los resultados obtenidos.

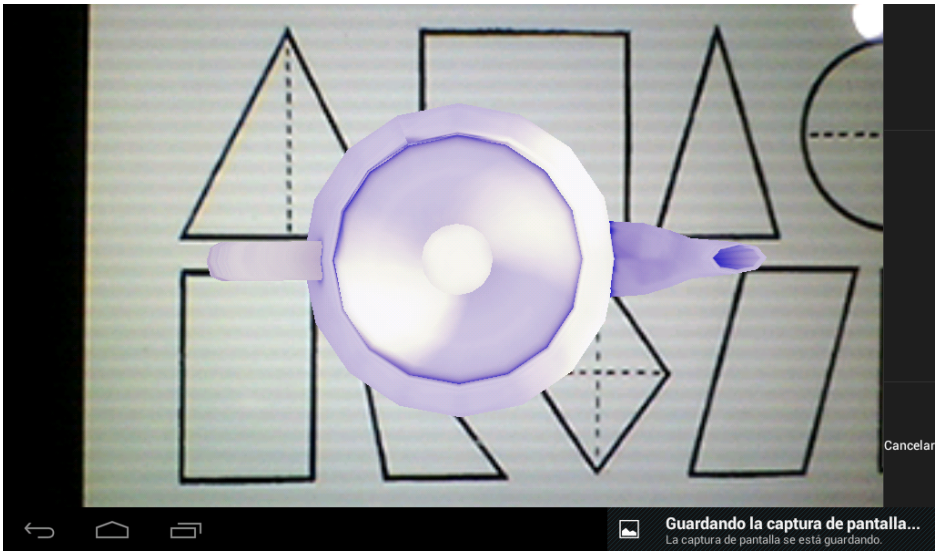


Figura 7.6: Representación del modelo.

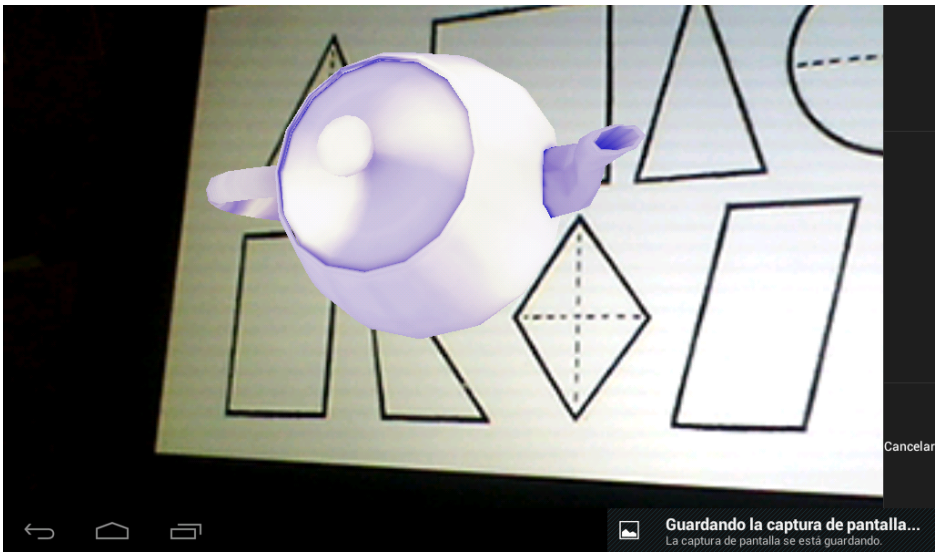


Figura 7.7: Representación del modelo.

Conclusiones y trabajo a futuro.

7.3. Conclusiones.

Con el trabajo realizado, concluimos que el uso de dispositivos móviles en la actualidad es muy demandado, por lo tanto existe mucho desarrollo de aplicaciones con diferentes tecnologías, las cuales han ido evolucionando, una de estas evoluciones ha sido la realidad aumentada que en un principio se planteó con el uso de marcadores, como fueron los códigos QR. En este trabajo realizamos el reconocimiento de figuras básicas sin el uso de marcadores, utilizando algoritmos de procesamiento de imágenes que están implementados en OpenCV, y para la representación del modelo en 3D utilizamos OpenGL, si se quisieran reconocer y representar figuras mas complejas podrían utilizar como base este trabajo, ya que con base al reconocimiento de figuras básicas se puede realizar el análisis de objetos más complejos.

En la actualidad, la realidad aumentada es una tecnología ampliamente usada en el mercado dentro de aplicaciones en las que cierta información es agregada al entorno real; permite, además, la simulación de actividades de alto riesgo. Las áreas y aplicaciones que tiene la realidad aumentada hoy en día es amplia y, por tanto, las herramientas de desarrollo de estas aplicaciones son extensas de igual forma.

Una aplicación de realidad aumentada requiere interacción con el mundo real para proporcionar información coherente con el entorno. En su mayoría las aplicaciones existentes hacen uso de elementos externos, como marcadores, para detectar dónde y cómo se debe representar la información virtual. Sin embargo, podemos aprovechar las herramientas de desarrollo que nos permiten hacer un procesamiento de lo que se observa en el entorno, como la extracción de características y el procesamiento digital, proporcionando así una interacción más natural al momento de usar la aplicación de realidad aumentada.

7.4. Trabajo a futuro.

Se contempla como trabajo a futuro mejoras en cuanto al reconocimiento de figuras más complejas en el entorno, así como reconocer estas mismas en un ambiente no controlado.

El Trabajo Terminal puede ser utilizado como base para desarrollar sistemas similares en otras plataformas móviles como iOS para iPhone o Ipad y Windows Mobile para Smartphones o Tablets.

Actualmente, se desarrollan más y mejores herramientas para los desarrolladores que permiten reconocer entornos y crear realidad aumentada con base en estos. Como trabajo a futuro, se puede hacer uso de estas API's para recrear y mejorar el concepto de este trabajo terminal, haciendo uso de hardware que permita reconocer superficies 3D y herramientas para la creación de modelos de realidad aumentada dinámicos, como lo son los últimos desarrollos de Vuforia.

Bibliografía

[1] Rosa Atzín Vázquez del ángel. Sistema de desarrollo para aplicaciones de realidad aumentada. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional,México, D. F. Octubre 2010.

[2] Gilberto Nájera Gutiérrez. Realidad Aumentada en Interfaces Hombre Maquina. Centro de investigación en computación, Junio de 2009.

[3] Brian Wanstall. Hud on the head for combat pilots. Interavia, (44):334ñ338, 1989.

[4] Charles E. Hughes, Christopher B. Stapleton, Darin E. Hug, and Eileen M. Smith. Mixed reality in education, entertainment, and training. IEEE Computer Graphics and Applications, pages 24ñ30, December 2005.

[5] B. Reitinger and D. Schmalstieg. Augmented reality scouting for interactive 3D reconstruction. Proceedings of IEEE Virtual Reality, pages 219-222, 2007.

[6] Alfredo Hernández Moreno and Lino Soberanes Pérez. Sistema para la integración de entornos Reales y Virtuales mediante Realidad Aumentada. Ingeniería en sistemas computacionales. ESCOM, Mayo 2008.

[7] Carlos Alcarria Izquierdo. Desarrollo de un sistema de Realidad Aumentada en dispositivos móviles.Universidad Politécnica de Valencia Escuela Técnica Superior de Ingeniería Informática,VALENCIA,2010.

[8] © 2013 Layar, All Rights Reserved, Layar 3D Model Converter [online]. Disponible en: http://www.layar.com/documentation/browser/3d-model-converter/

[9] ARToolworks, Inc, ARToolKit [online]. Disponible en: http://www.hitl.washington.edu/artoolkit/

[10] http://www.intel.com/research/mrl/research/opencv http://sourceforge.net/projects/opencvlibrary

[11] A. C. Carlos, L. I. Lárcher, A. I. Ruggeri, A. C. Herrera y E. M. Biason, Métodos de umbralización de imágenes digitales basados en entropía de Shannon y otros, Mecánica Computacional , vol. XXX, pp. 2785-2805, 2011.

[12] Disponible en: http://www.buildar.co.nz/home/what-is-buildar/

[13] Disponible en: http://www.sologicolibre.org/projects/atomic/en/

[14] Disponible en: http://www.sketchup.com/intl/es/

[15] Disponible en: http://www.blender.org