UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

Tema:

"PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA PARA EL APRENDIZAJE INTERACTIVO EN LA UNIDAD EDUCATIVA "TIRSO DE MOLINA", DE LA CIUDAD DE AMBATO."

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniera en Sistemas Computacionales e Informáticos.

SUBLÍNEA DE INVESTIGACIÓN: Ingeniería Computacional

AUTORA: Caguana Tibán Jimena del Rocío

TUTOR: Ing. Mg. Clay Fernando Aldás Flores

Ambato – Ecuador

Diciembre – 2015

APROBACIÓN DEL TUTOR

En mi calidad de tutor del Trabajo de Investigación sobre tema: **"Pizarra Virtual usando Realidad Aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato"**, de la señorita Jimena del Rocío Caguana Tibán, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, Diciembre 2015

TUTOR

Ing. Mg. Clay Fernando Aldás Flores

AUTORÍA

El presente Proyecto de Investigación titulado: **"Pizarra Virtual usando Realidad Aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato"**, es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad de la autora.

Ambato, Diciembre 2015.

AUTORA

Jimena del Rocío Caguana Tibán

C.I.: 180442270-5

DERECHOS DE AUTORA

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, Diciembre 2015.

AUTORA

Jimena del Rocío Caguana Tibán

C.I.: 180442270-5

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes **Ing. Oswaldo Paredes** e **Ing. Alba Miranda**, revisó y aprobó el Informe Final del Proyecto de Investigación titulado **"Pizarra Virtual usando Realidad Aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato"**, presentado por la señorita Jimena del Rocío Caguana Tibán de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, Diciembre 2015

Ing. Mg. Vicente Morales Lozada
PRESIDENTE DEL TRIBUNAL

Ing. Mg. Oswaldo Paredes
DOCENTE CALIFICADOR

Ing. Mg. Alba Miranda
DOCENTE CALIFICADOR

DEDICATORIA

A Dios, por haberme permitido llegar hasta este punto y haberme dado sabiduría, salud e inteligencia para lograr mis objetivos.

A mis Padres Jaime Caguana(+) y Narciza Tibán, por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo en todo momento, gracias a ellos escribo estas palabras. A mi hermano Christian Caguana, en quien deposito mis esperanzas para perdurar en el tiempo. A mis abuelitos, tíos y a todos aquellos que participaron directa o indirectamente en la elaboración de esta tesis.

A mis maestros, quienes marcaron cada etapa de nuestro camino universitario, con su gran apoyo y motivación para la culminación de nuestros estudios y para la elaboración de esta tesis; al Ing. Clay Aldas por su asesoría para resolver las dudas presentadas en el desarrollo de la tesis; Al Rvdo. Luis Abad y Lic. Luzmila Fiallos por su apoyo ofrecido en este trabajo.

A mis amigos, nos apoyamos mutuamente en nuestra formación profesional y que hasta ahora, seguimos siendo amigos.

¡Gracias a ustedes!

Jimena

AGRADECIMIENTO

El presente trabajo de tesis primeramente me gustaría agradecerte a ti Dios por bendecirme para llegar hasta donde he llegado, porque hiciste realidad este sueño anhelado.

En segundo lugar a mi padre Jaime Caguana (+), mi madre Narciza Tibán, por siempre haberme dado su fuerza y apoyo incondicional que me ha ayudado y llevado hasta donde estoy ahora.

A la UNIVERSIDAD TÉCNICA DE AMBATO por darme la oportunidad de estudiar y ser un profesional.

A mi tutor de tesis, Ing. Clay Aldas, quien con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado en mí que pueda terminar mis estudios con éxito.

De igual manera agradecer a mis profesores durante toda mi carrera profesional porque han apoyado con un granito de arena a mi formación.

Son muchas las personas que han formado parte de mi vida universitaria a quienes agradezco su amistad, concejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Algunas están aquí conmigo y otras en mi recuerdo y corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han dado y por sus bendiciones que he recibido.

Jimena

ÍNDICE GENERAL

CONTENIDOS **PÁGINAS**

CAPÍTULO II

MARCO TEÓRICO

CAPÍTULO III

METODOLOGÍA

CAPÍTULO IV

DESARROLLO DE LA PROPUESTA

ÍNDICE DE FIGURAS

FIGURA **PÁGINAS**

ÍNDICE DE TABLAS

TABLA **PÁGINAS**

RESUMEN

El presente Trabajo de Graduación, está orientado al desarrollo e implementación de una Pizarra Virtual usando Realidad Aumentada (PIVRA). Tanto el diseño del software, así como el sistema PIVRA; están dirigidos a niños y jóvenes en etapa de Educación Básica, particularmente para el aprendizaje de la asignatura de Ciencias Naturales. El propósito de PIVRA, es ser una herramienta didáctica de apoyo, que ilustra de forma interactiva el reconocimiento de modelos 3D con técnicas de Realidad Aumentada (RA); con el fin que docentes y alumnos puedan visualizar de manera minuciosa las partes biológicas del ser humano, para así; interiorizar el conocimiento y resolver problemas referentes al tema expuesto en esta cátedra.

La categoría de ingeniería informática en PIVRA, hace uso de la RA, además de técnicas de reconocimiento de patrones, visión artificial y procesamiento de imágenes; como objeto para mejorar los procesos de enseñanza-aprendizaje. PIVRA ofrece al usuario una interacción con elementos reales y virtuales en un mismo entorno. El lenguaje base de programación empleado para su desarrollo es C Sharp; la parte de captura y procesamiento de imágenes en movimiento se llevó a cabo con OpenCV y su envoltorio para .NET, EmguCV. A la hora de acercar todas estas tecnologías al usuario, se empleó las librerías OpenCVSharp y AForge.NET para crear una interfaz gráfica de usuario, sencilla y fácil de usar; pero robusta en su componente de visión computacional y RA. Todas estas soluciones permiten su utilización de forma GNU y se adaptan correctamente a las necesidades del sistema PIVRA.

Incorporar las tecnologías emergentes al entorno educativo, vinculan la emotividad de los educandos al uso de la técnica y la ciencia; convirtiéndose en una herramienta indispensable para su enriquecimiento intelectual. Con PIVRA se propone cubrir aspectos específicos de la educación como son la contextualización de contenidos, o el seguimiento y evaluación del aprendizaje de los alumnos. La experiencia con la RA se realiza a través de una aplicación de escritorio denominada PIVRA, en la cual los estudiantes interactúan con la RA y con el recurso virtual aumentado para beneficio de su conocimiento integral.

SUMMARY

This Work Graduation, it focuses on the development and implementation of a virtual whiteboard using Augmented Reality (PIVRA). The design of the software and the system PIVRA; they are directed at children and young people in Basic Education stage, particularly for learning the subject of Natural Sciences. The purpose of PIVRA, it is to be a teaching tool support, illustrating interactively 3D models recognition techniques Augmented Reality (AR); so that teachers and students can visualize minutely biological parts of the human being, so; internalize the knowledge and solve problems concerning the issue exposed in this professorship.

The category of computer engineering at PIVRA, it makes use of the AR, besides pattern recognition techniques, machine vision and image processing; as an object to improve the teaching-learning. PIVRA offers users interact with real and virtual elements in the same environment. The basic programming language used for their development was C Sharp; the capture and processing of moving images was performed with OpenCV and its wrapper for .NET, EmguCV. When bringing these technologies to the user, AForge.NET the OpenCVSharp and libraries are used to create a graphical user interface, simple and easy to use; but robust in its component of computer vision and AR. All these solutions allow such use and adapt GNU correctly system needs PIVRA.

Incorporate emerging technologies into the educational environment, binding activity of students the use of technology and science; becoming an indispensable tool for intellectual enrichment. PIVRA proposes to cover specific aspects of education such as the contextualization of contents, or monitoring and evaluation of student learning. Experience with AR is done through a desktop application called PIVRA, in which students interact with the AR and the increased virtual resource for the benefit of its comprehensive knowledge.

GLOSARIO

Algoritmo: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Binarizar: Proceso para obtener una imagen en blanco y negro.

Cognoscitivo: Que es capaz de conocer o comprender.

Fotograma: Frame o cuadro, es una imagen particular dentro de una sucesión de imágenes que componen un video.

Fotogramas Por Segundo: Es la medida de la frecuencia a la cual un reproductor de imágenes genera distintos fotogramas. Estos fotogramas están constituidos por un número de pixeles que se distribuyen a lo largo de una red de texturas.

Ingeniería de Software: Es la disciplina dentro de la informática encargada de la creación de software de calidad.

Ingeniería de Usabilidad: Conjunto de técnicas para el desarrollo de software en la que se especifican previamente niveles cuantitativos de usabilidad.

Interfaz de usuario: Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

Lenguaje: Se llama lenguaje a cualquier tipo de código semiótico estructurado, para el que existe un contexto de uso y ciertos principios combinatorios formales.

Librería: Es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

Marcador: También conocido como patrón, es una imagen en blanco y negro que sirve de guía con la que se puede enfocar la cámara y ver la Realidad Aumentada.

Programación Orientada a Objetos: Es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo, y encapsulamiento.

Pixel: Un píxel o pixel, plural píxeles, es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

Pizarra interactiva: Es un panel físico que puede funcionar como una pizarra común, adicionalmente puede servir como una pantalla de proyección donde la imagen de la computadora puede ser controlada tocando o escribiendo sobre la superficie del panel.

Prototipo: Es un ejemplar o primer molde en que se fabrica una figura u otra cosa. En informática es la primera versión de un software.

Realidad Virtual: Es una tecnología que permite al usuario sumergirse en una simulación grafica 3D generada por computador y navegar e interactuar en ella en tiempo real.

Realidad Aumentada: Es una tecnología que combina el entorno físico del mundo real con elementos virtuales, permitiendo al usuario estar en un entorno real aumentado, esta interacción se logra con un conjunto de dispositivos que añaden información virtual a la física ya existente.

Reconocimiento: Distinción de una persona o cosa entre las demás por sus rasgos o características.

Tracking: Es el seguimiento de ojos; es decir, es el proceso de evaluar, bien el punto donde se fija la mirada, o el movimiento del ojo en relación con la cabeza.

Usabilidad: Se dice que un sistema tiene buena usabilidad cuando este presenta una interfaz gráfica amigable y sencilla; es decir, el usuario no necesita ser experto en el área para poder utilizar el sistema.

Virtual: Que tiene existencia aparente y no real. En computación se utiliza para designar a todo aquello que tiene existencia dentro de una simulación informática.

Visión por computador: También conocida como visión artificial, es el proceso que permite tomar imágenes del mundo real para extraer de ellas información necesaria para una aplicación específica.

ACRÓNIMOS

ARML: Augmented Reality Markup Language. Lenguaje de Marcas para la Realidad Aumentada.

CASE: Compute Aided Software Engineering. Ingeniería de Software Asistida por Computador.

FPS: Frames Per Second. Fotogramas Por Segundo.

GPL: General Public License. Licencia Pública General.

GUI: Graphics User Interface. Interfaz Gráfica de Usuario.

HCI: Human Computer Interaction. Interacción Persona Computador.

HDM: Head Display Mounted. Display montado sobre la cabeza.

IU: Usability Engineering. Ingeniería de Usabilidad.

OpenSource: Open Source. Código abierto.

OpenCV: Open source Computer Vision. Código abierto para Visión por Computadora.

PIVRA: Blackboard Virtual using Augmented Reality. Pizarra Virtual usando Realidad Aumentada.

OOP: Object Oriented Programming. Programación Orientada a Objetos.

QUIS: Questionnaire for User Interaction Satisfaction. Cuestionario para satisfacción Interacción con el Usuario.

AR: Augmented Reality. Realidad Aumentada.

RAM: Random Access Memory. Memoria de Acceso Aleatorio.

RGB: Red Green Blue. Rojo Verde Azul.

SO: Operating System. Sistema Operativo.

SUMI: Software Usability Measuring Inventory. Inventario de medición de Usabilidad de Software.

TIC: Information and Communication Technologies. Tecnologías de la Información y Comunicación.

UML: Unified Modeling Languaje. Lenguaje Unificado de Modelado.

VRML: Virtual Reality Modeling Language. Lenguaje para Modelado de Realidad Virtual.

WAMMI: Website Analysis and Measure Ment Inventory. Análisis de Sitio Web y Medida mentada de Inventario.

XML: Extended Markup Language. Lenguaje de Marcas Extensible.

INTRODUCCIÓN

La tecnología de la información y comunicación, tiene como agente de desarrollo el internet, proporcionando apertura a la evolución del software y hardware; de tal forma que han aparecido nuevos paradigmas de interacción como la visión artificial, la realidad virtual, la realidad aumentada o la computación ubicua. El impacto de estos paradigmas de vanguardia, se están haciendo notar en distintos sectores entre los que se incluye la educación, donde se está transformando el modo de enseñanza y aprendizaje tradicional. De igual forma, están evolucionando la manera de impartir las clases magistrales; a través de la introducción en el aula de herramientas y servicios como reproductores de audio, proyectores multimedia y pizarras electrónicas.

La Realidad Aumentada (RA) permite la convergencia tecnológica en el procesamiento de imágenes de video en tiempo real y la visión artificial, que hacen posible la inmersión óptica de una imagen virtual en el entorno tridimensional que rodea al usuario. La RA permite la multiplexación de una escena real con otra virtual generada por un computador, fomentada y complementada con información digital y algoritmos matemáticos programados. El prototipo de este Trabajo de Graduación pretende dar un paso hacia una técnica innovadora en los procesos de enseñanza y aprendizaje, para comprobar la utilidad interactiva que puede tener una herramienta de software, al tomar ventaja de la RA.

La Pizarra Virtual usando Realidad Aumentada (PIVRA) en su entorno de herramienta educativa emplea, por un lado; técnicas de RA como tecnología base para dotar a la realidad, de contenidos digitales interactivos, y por otro lado; una pizarra y software de escritorio para la visualización e interacción con estos contenidos digitales. En este entorno se da la posibilidad de interactuar y preguntar al alumno sobre las cuestiones planteadas en los recursos digitales empleados; de esta forma no sólo la visualización y representación de éstos, ayuda a entender de una mejor manera aquello que se pretende enseñar sino que se refuerza la comprensión y el aprendizaje de una forma más amena y transparente.

PIVRA focaliza su operatividad en el análisis de una imagen mediante un sistema de visión artificial, para identificar objetos y regiones de interés dibujado o impreso por el usuario. El sistema PIVRA comprueba si existe algún dibujo u objeto relevante a detectar; si existiera algún dibujo u objeto, éste es cambiado por una imagen digital o modelo 3D almacenado en el sistema; a este dibujo u objeto se le conoce como marcador para RA. Un marcador para RA en el sistema PIVRA, es una imagen dibujada con un rotulador en una pizarra ordinaria de tiza líquida o una imagen impresa en una hoja; la característica distintiva de esta imagen dibujada o impresa, es la combinación de color blanco y negro; y que representa la marca de reconocimiento de PIVRA para la respectiva proyección del modelo 3D u imagen digital.

El contexto de este Trabajo de Graduación es el de los sistemas de información y aplicaciones utilizadas para mejorar la experiencia de los procesos de enseñanza y aprendizaje, con la RA. Entre las ventajas que se destacan sobre esta modalidad de educación está la presentación visual de información digital, la comunicación del conocimiento conceptual e incluso valores de motivación, aparte del contenido cognitivo. Permitiendo una solución a la heterogeneidad que tienen los estudiantes presentes en un curso académico, los conocimientos iniciales, las inquietudes, etc. Un estudiante con dificultades en la comprensión de un determinado tema dispondrá de recursos digitales que le ayuden a alcanzar las competencias deseadas y por otro lado un alumno con conocimientos previos del tema, puede profundizar en él de una forma más exhaustiva.

Para constituir en forma general el contenido de esta tesis; se encuentra comprendida en cinco capítulos:

En el CAPÍTULO I se argumenta el Problema, la delimitación, la justificación y los objetivos del estudio.

En el CAPÍTULO II se diverge acerca del Marco Teórico, los antecedentes investigativos, la fundamentación teórica y la propuesta solución.

En el CAPÍTULO III se expone la Metodología utilizada en el trabajo de investigación, su modalidad, los métodos y técnicas para la recolección de información, el

procesamiento y análisis de datos; para en última instancia, concebir el desarrollo del proyecto.

En el CAPÍTULO IV se presenta el Desarrollo de la Propuesta, sus antecedentes, la justificación y factibilidad, el análisis de requisitos, el diseño del software, la codificación, el diseño de la pizarra; las respectivas pruebas, verificación y mantenimiento fehacientes a la implementación de la Pizarra Virtual usando Realidad Aumentada.

Finalmente en el CAPÍTULO V se redactan las Conclusiones y Recomendaciones; se aglutina la Bibliografía y los Anexos.

CAPÍTULO I

EL PROBLEMA

1.1. Tema

"PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA PARA EL APRENDIZAJE INTERACTIVO EN LA UNIDAD EDUCATIVA "TIRSO DE MOLINA", DE LA CIUDAD DE AMBATO".

1.2. Planteamiento del Problema

El conocimiento es receptado por el ser humano en un 90 %, a través de imágenes, pero que sucede si las imágenes presentan degradación o distorsión en el proceso de educación; este deja de ser eficiente e interactivo entre educador, educando, entorno, materiales y herramientas virtuales; con ello el aprendizaje se convierte en un tipo de metodología monótona e inverosímil.

Muchos investigadores y educadores coinciden que el uso de nuevas tecnologías hace que el interés de los alumnos y la participación activa de éstos aumenten. La Realidad Aumentada (RA) y su uso en el aprendizaje persigue varios objetivos: desarrollar sistemas para aprender de manera más rápida conceptos a partir de interacciones que puedan realizar los propios alumnos con su máxima concentración; conseguir un entendimiento más claro y profundo del proceso de aprendizaje humano; crear aplicaciones que permitan acelerar el proceso de asimilación de conocimientos, etc.

A nivel mundial la tecnología presenta grandes avances en el ámbito de la Inteligencia Artificial, la Visión por Computadora y el Procesamiento Digital de Imágenes; pero la técnica conocida como RA ha mejorado las actividades de ingeniería, medicina, educación, seguridad, industria, entretenimiento digital, robótica, cultura, publicidad,

arte, arquitectura, etc. Actualmente existen muchos proyectos enfocados al conocimiento y desarrollo de habilidades para niños, jóvenes y adultos, entre ellos se mencionan los siguientes:

Magic Book (Doreen, 2003) del grupo activo HIT de Nueva Zelanda. Este proyecto permite al alumno leer un libro real a través de un visualizador de mano y ve sobre las páginas reales contenidos virtuales. De esta manera cuando el alumno ve una escena de RA que le guste, puede introducirse dentro de la escena y experimentarla en un entorno virtual inmersivo. **[1]**

Al mismo tiempo, existen muchos proyectos desarrollados por Institutos y Universidades de Estados Unidos; con el objeto de uso a nivel secundario, universitario y científico: Un ejemplo concreto es HANDHELD AUGMENTED REALITY PROJECT (HARP, 2012), con el financiamiento del Departamento de Educación de EE.UU. Star Schools Program, los investigadores de la Escuela de Educación de Harvard, junto a la Universidad de Wisconsin de Madison, y el programa de formación al docente en el MIT han desarrollado un juego basado en RA, diseñado para enseñar matemáticas y ciencias. **[1]**

En el Ecuador, instituciones de educación superior como la Escuela Politécnica Nacional (Quito), la Escuela Superior Politécnica de Litoral (Guayaquil) y la Universidad Técnica Particular de Loja (Loja) han desarrollado trabajos de investigación y tesis, en la ingeniería de la RA. Las tres universidades mencionadas se han inmerso en los diversos tipos de RA; la RA basada en el reconocimiento de marcadores, la RA basada en el reconocimiento de objetos y la RA de reconocimiento basado en geolocalización o RA móvil.

En la Provincia de Tungurahua, en particular en la ciudad de Ambato, la RA es un aspecto de la ingeniería y la inteligencia artificial, que ha pasado desatendida y sin atracción para su pertinente investigación por parte de las instituciones de educación superior, empresas privadas o públicas. La Universidad Técnica de Ambato y la Facultad de Ingeniería en Sistemas, Electrónica e Industrial en vinculación con la Unidad Educativa “Tirso de Molina” se propone incursionar en la tecnología de RA con este proyecto de investigación.

La problemática que se suscita al prescindir de la RA como tecnología de enseñanza en la Unidad Educativa "Tirso de Molina", se recopila de la siguiente manera: existe deficiencia en la percepción visual de la realidad física (problemas visuales de niños y jóvenes) para la educación básica; la comprensión de contenidos es desacertada, debido a una interacción indirecta con conceptos, situaciones y estructuras (gráficos en 2D mal diseñados, impresos erróneamente o de pésima calidad); deficiente registro de imágenes en libros y presentaciones educativas; no uso de imágenes virtuales 3D en RA, no existe tecnología de vanguardia para el aprendizaje y formación de los alumnos, docentes y padres de familia.

Una pedagogía y metodología monótona en los procesos de enseñanza y aprendizaje, puede cuantificar la escaza concentración, el bajo aprovechamiento, la limitada motivación al aprendizaje y hasta la mala conducta en los alumnos de instituciones de educativas. En el mundo tan tecnificado en el que vivimos actualmente, la mejor estrategia de desarrollo y calidad educativa es la consecución de proyectos de ciencia y tecnología que estimulen la creatividad y la participación del alumno dentro del conocimiento.

1.3. Delimitación

1.3.1 Delimitación de Contenido

- Área académica: Hardware y Redes.
- Línea de investigación: Tecnologías de la Información.
- Sublínea de investigación: Ingeniería Computacional.

1.3.2 Delimitación Espacial

El proyecto de investigación se desarrolló para el Laboratorio de Informática de la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato.

1.3.3 Delimitación Temporal

La presente investigación se ejecutó en seis meses, a partir de su aprobación por el Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

1.4. Justificación

La Realidad Aumentada (RA) es una tecnología desarrollada desde hace varios años, implementada principalmente en el cine para la realización de efectos especiales; pero que ha evolucionado hasta la actualidad, dejando de ser exclusiva para los efectos especiales y cubriendo todos los campos de la ciencia.

Una pizarra virtual usando Realidad Aumentada para el aprendizaje interactivo de la Unidad Educativa "Tirso de Molina", tiene múltiples beneficios para los procesos de enseñanza y aprendizaje; no solo por impulsar la concentración y motivación del alumno a la ciencia y la tecnología, o la trascendencia del contenido real de una asignatura a una asimilación de conocimientos con entornos virtuales 3D en tiempo real. La RA también permite convertir una simple pizarra de marcador, o una pantalla para proyección con INFOCUS, en una pizarra virtual en la que se recepte contenidos de un libro o del mundo real en forma virtual 3D, con el cual el alumno tendrá una perspectiva global y en todos los ángulos del tema que la asignatura plantee para su aprendizaje diario.

Ver la disección de una rana en RA; las texturas, colores, contornos de una pintura de un artista famoso en RA; las notas de las composiciones célebres de la música clásica en RA, los movimientos físicos (movimiento rectilíneo uniforme, movimiento armónico simple, el tiro de proyectil, la caída libre, las leyes de Newton, etc.) con ejemplos de la vida real en RA, o simulaciones virtuales en RA; y muchos otros aspectos que incurren en la educación, hacen de la RA una tecnología ideal para los métodos de enseñanza y aprendizaje.

La RA al estar derivada de la ciencia del procesamiento digital de imágenes y video, además de la Visión Artificial; cuenta con el acierto técnico de muchos programas de computadora, robustos en el manejo de algoritmos de procesamientos de imágenes a través de cámaras, cálculos geométricos y matriciales para la proyección de imágenes en 3D, reconocimiento de objetos o marcadores; entre otras funcionalidades coherentes a la RA. Razón que justifica el desarrollo de este Trabajo de Investigación.

Los beneficiarios directos del proyecto de investigación es la Unidad Educativa "Tirso de Molina" aglomerada entre los estudiantes, docentes y padres de familia, que se educan y capacitan en los laboratorios de informática de esta institución. La sociedad ambateña y ecuatoriana sumida en el estudio del campo de la Visión Artificial. A su vez; los beneficiarios indirectos, son los lectores de este documento científico-técnico, la Universidad Técnica de Ambato, la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI); los estudiantes de la carrera de Ingeniería en Sistemas Computacionales e Informáticos, que se motiven por la RA para sus trabajos de graduación.

La elaboración de este proyecto cuenta con el aval y la ayuda de profesionales catedráticos de la FISEI. Las fuentes de información primaria y secundaria son variadas y están accesibles en repositorios virtuales y en documentos impresos a disposición de la investigadora, permitiéndole obtener parámetros de ingeniería y criterios científicos relevantes; para analizar, desarrollar e implementar una pizarra virtual usando Realidad Aumentada; eficiente, que mejore la calidad metodológica en la educación integral de los alumnos de la Unidad Educativa "Tirso de Molina" con el uso de herramientas informáticas y tecnológicas.

1.5. Objetivos

1.5.1 Objetivo General

- Implementar una pizarra virtual usando realidad aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato.

1.5.2 Objetivos Específicos

- Analizar la situación tecnológica de la Realidad Aumentada destinada a la educación interactiva en instituciones de formación académica.
- Argumentar en forma computacional el método óptico y matemático; para la exposición de la Realidad Aumentada en tiempo real, sobre una pizarra.
- Diseñar una pizarra virtual usando Realidad Aumentada para el aprendizaje interactivo de la asignatura de Ciencias Naturales en la Unidad Educativa "Tirso de Molina".

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes Investigativos

En la Universidad Técnica Particular de Loja (Ecuador), el autor Rodrigo Alexander Saraguro Bravo, desarrolló como tema tesis la implementación de una aplicación Android basada en Realidad Aumentada aplicada a puntos de interés de la UTPL; en el año 2012, obteniendo como conclusión que uno de los principales desafíos del sistema de RA UTPLRA es el registro preciso de los objetos virtuales con el entorno real. Esta funcionalidad tiene una exactitud entre el 90% y depende de factores como la señal de red disponible, y la exactitud del GPS. **[2]**

En la Universidad Católica de Pereira (Colombia), el autor Juan Sebastián Duque Álvarez, realizó como tema de tesis la Investigación y desarrollo de una aplicación en Realidad Aumentada para la empresa PLUGAR; en el año 2011, obteniendo como conclusión principal la entrega de un prototipo de prueba de implementación Web con la cual la empresa se guiará para la muestra de los productos en su página Web. **[3]**

En la Universidad Politécnica de Valencia (España), el autor Carlos Alcarria Izquierdo, implementó como tema de tesis un sistema de Realidad Aumentada en dispositivos móviles; en el año 2010, obteniendo como conclusión que la RA se vuelve móvil con el avance del hardware y del software de los dispositivos portátiles. **[4]**

En el libro con el título Realidad Aumentada: Un Enfoque Práctico con ARToolKit y Blender; de los autores Carlos González Morcillo, David Vallejo Fernández, Javier A. Albusac Jiménez y José Jesús Castro Sánchez; publicado en el año 2012, exponen

contextos científicos sobre la realidad aumentada desarrolladas con herramientas informáticas como ARToolKit y Blender. **[5]**

En el libro Realidad Aumentada del Autor Bruno Nievas; publicado en el año 2011, ofrece un contenido científico técnico sobre proyectos de realidad aumentada y desarrollo de sistemas innovadores con este tipo de alta tecnología. **[6]**

En el libro Realidad Aumentada basada en características naturales: Un enfoque práctico, de los autores Germán Ros y Gin S Garc; publicado en el año 2012, describen contenidos prácticos sobre la Realidad Aumentada en campos de la visión por computadora. **[7]**

En el paper con el tema Realidad Aumentada en la Educación: una tecnología emergente, de los autores X. Basogain, M. Olabe, K. Espinosa, C. Rouéche y J. C. Olabe, de la Escuela Superior de Ingeniería de Bilbao; publicado en el año 2012, contextualizan la realidad aumentada, sus parámetros técnicos, diagramas y otras características. **[8]**

En el paper con el tema Realidad Aumentada en el tratamiento de las enfermedades mentales y las adicciones, de los autores Jorge Gaviria, Guillermo Castaño, Byron Rosero y José Sierra de la Universidad de San Buenaventura (Medellín Colombia); publicado en el año 2013, redactan argumentación técnica sobre el uso de la Realidad Aumentada y la Realidad Virtual en enfermedades mentales y adictivas, dando una síntesis comparativa de Realidad Aumentada vs Realidad Virtual. **[9]**

En el paper con el tema Uso de Realidad Aumentada para Enseñanza de Conceptos Básicos de Física Mecánica, de los autores Juan Marino y Ingrid De León de la Universidad Libre Barranquilla; publicado en el año 2012, narra la importancia de la Realidad Aumentada y las Tecnologías de la información y las comunicaciones en el entorno educativo para el desarrollo integral de los alumnos, de los aspectos cognoscitivos, constructivos y Construccionistas. **[10]**

Los antecedentes investigativos acreditados a la tecnología denominada realidad aumentada trascienden en contexto y en material informativo-práctico; la descripción detallada en los párrafos anteriores corrobora la importancia de esta técnica informática.

2.2. Fundamentación Teórica

2.2.1 Educación Interactiva

El actual estilo tradicional de educación, no estimula la participación del alumno en la construcción del conocimiento. Si la enseñanza se centra solamente, en la emisión docente-libro; el receptor, es decir, al alumno, sólo ocupa un espacio receptivo-pasivo. Los docentes deben entender que el aprendizaje es un proceso de construcción del alumno, que es él que elabora los saberes gracias y a través de las interacciones con el educador y con los demás alumnos. La educación interactiva en la era digital puede ser definida como modos de vida y de comportamientos asimilados y transmitidos en la vivencia cotidiana, marcados por las tecnologías digitales mediante la comunicación y la información e interfiriendo en el imaginario del sujeto. **[11]**

Respecto a la definición de la palabra interactividad, el diccionario de la Real Academia de la Lengua, define interactivo (*adj. Que procede por interacción*); habiendo una segunda acepción (*Inform. Dicho Pde un programa o ambiente tecnológico: Que permite una interacción, a modo de diálogo, entre el ordenador y el usuario*). **[11]**

De esta forma las nuevas tecnologías aportan un nuevo reto a la enseñanza que consiste en evolucionar de un modelo unidireccional de formación, donde los conocimientos recaen en el docente o en el libro de texto, a modelos más abiertos donde la información se guarda en formato digital y multimedia, disponible para todos los alumnos. **[11]**

2.2.2 Uso de las TIC en procesos de aprendizaje

Las TIC han llegado a ser uno de los pilares básicos de la sociedad y hoy es necesario proporcionar a estudiantes una educación que sea acorde con esta realidad. Las TIC se relacionan en dos aspectos con la educación; uno es el conocimiento de las mismas y otro es su uso pedagógico. **[11]**

El conocimiento de las TIC es la consecuencia de la sociedad actual, no se puede entender el mundo de hoy sin un mínimo de conocimiento de las mismas. Es preciso entender cómo se genera, cómo se almacena, cómo se transforma, cómo se transmite y

cómo se accede a la información en sus múltiples formas (texto y contenido multimedia). **[11]**

El segundo aspecto, corresponde al uso de las TIC en la educación, relacionando con el conocimiento de éstas, pero en un ámbito más técnico. Se debe usar las TIC para aprender y para enseñar. Es decir, el aprendizaje de cualquier asignatura o habilidad se puede facilitar mediante el uso de cualquier TIC y en particular de la RA. **[11]**

2.2.3 Representación de imágenes digitales

Es necesario contextualizar contenidos básicos de la representación de imágenes.

Píxel: Es la abreviatura de las palabras inglesas *"picture element".* Es el menor de los elementos de una imagen, al que se puede aplicar individualmente un color o una intensidad; o que se puede diferenciar de los otros mediante un determinado procedimiento. La Fig. 2.1 muestra la representación de un píxel. **[12]**

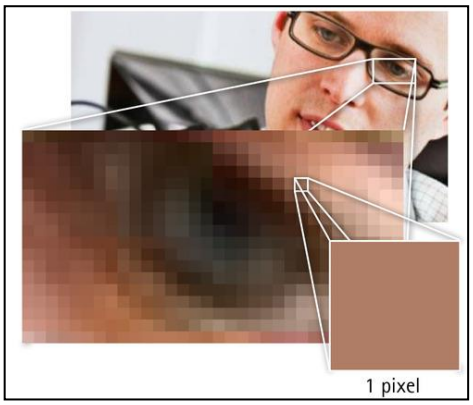


Fig. 2.1: Representación de un pixel
Fuente. http://www.functionx.com/vcsharp/illustrations/pixel.jpg - [12].

Imagen Digital: Una imagen digital se compone de una agrupación de píxeles, cada uno con un valor de intensidad o brillo asociado. Una imagen digital se representa mediante una matriz bidimensional, de forma que cada elemento de la matriz se corresponde con cada píxel en la imagen. La Fig. 2.2 muestra la estructura matricial de una imagen digital. **[12]**

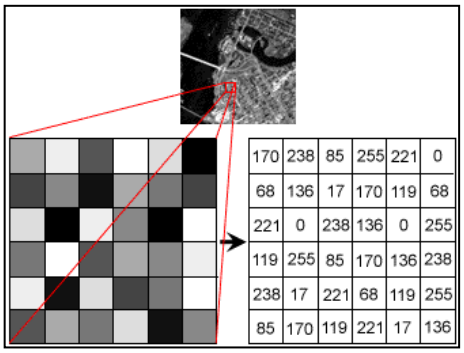


Fig. 2.2: Estructura matricial de una imagen digital
Fuente. http://www.invaringenieria.com/imagend48.jpg - [12].

<u>Clasificación de las Imágenes Digitales</u>: Dependiendo del rango de los valores que pueda tomar cada píxel, se puede distinguir los siguientes tipos de imágenes:

- *Imagen digital en blanco y negro:* El rango está formado por los valores negro y blanco [0 1]. En la Fig. 2.3 se muestra una imagen digital en blanco y negro. **[12]**

Fig. 2.3: Imagen digital en blanco y negro
Fuente. http://www.invaringenieria.com/imagendbn23.jpg - [12].

- *Imagen Digital de Intensidad:* También conocidas como imágenes en escala de grises, existen hasta 256 niveles de grises, por lo que su rango se encuentra entre [0, 255]. La Fig. 2.4 muestra una imagen digital en escala de grises. **[12]**

Fig. 2.4: Imagen digital en Escala de Grises
Fuente. http://www.invaringenieria.com/imagendint14.jpg - [12].

- *Imagen Digital de Color:* Todo color se puede componer a partir de tres componentes básicos RGB (Red, Green y Blue). El contenido de cada píxel de la imagen, es una terna de valores, un valor por cada componente de color básico. La Fig. 2.5 muestra una imagen digital de color. **[12]**

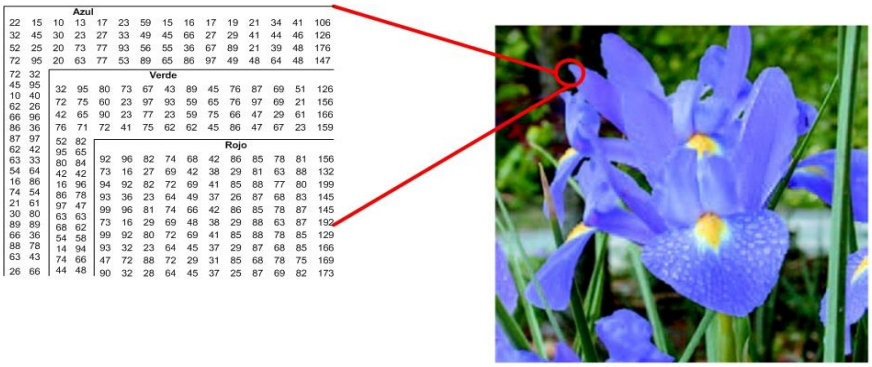


Fig. 2.5: Imagen digital de Color
Fuente. http://www.invaringenieria.com/imagendcol76.jpg - [12].

<u>Espacio de color (Modelo RGB)</u>: Es uno modelos más utilizados por los sistemas informáticos para reproducir los colores en el monitor y en el escáner. Está basado en la síntesis aditiva de las intensidades de luz relativas al rojo, al verde y al azul; para conseguir los distintos colores, incluyendo el negro y blanco. **[12]**

El nombre del modelo RGB viene de las iniciales en inglés, de los tres colores: Red, Green y Blue. La Fig. 2.6 muestra la representación gráfica del modelo de color RGB.

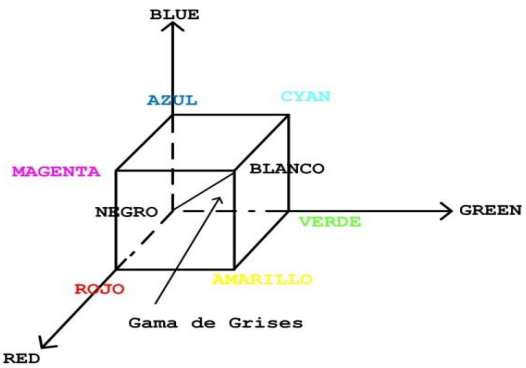


Fig. 2.6: Representación gráfica del modelo de color RGB
Fuente. http://www.invaringenieria.com/imagemodelRGB.jpg - [12].

La representación gráfica del modelo RGB se realiza mediante un cubo unitario con los ejes R, G y B (Red, Green y Blue). El origen (0, 0, 0) representa el negro y las coordenadas (1, 1, 1) el blanco. Los vértices del cubo en cada eje R, G y B, de

coordenadas (1, 0, 0), (0, 1, 0) y (0, 0, 1) representan los colores primarios rojo, verde y azul. Los restantes tres vértices (1, 0, 1), (0, 1, 1) y (1, 1, 0) al magenta, cian y amarillo respectivamente, colores secundarios y respectivamente complementarios del verde, rojo y azul. La diagonal del cubo representa la gama de grises desde el negro al blanco. En esta diagonal cada punto o color se caracteriza por tener la misma cantidad de cada color primario. **[12]**

Las imágenes con modelo RGB contienen tres planos de imágenes independientes, uno para cada color primario. Lo descrito sobre el modelo RGB, tiene gran importancia para el procesamiento digital de imágenes, a pesar de que no deriva de un proceso intuitivo para determinadas aplicaciones como por ejemplo: comparar colores, o extraer rasgos; es complementario de la Visión Artificial. **[12]**

2.2.4 Visión Artificial

La Visión Artificial es una ciencia que tiene como finalidad la extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador; se trata de un campo de la Inteligencia Artificial. Un sistema de Visión Artificial actúa sobre una representación de una realidad que le proporciona información sobre brillo, colores, formas, texturas, etc. estas representaciones se pueden agrupar principalmente en imágenes estáticas, escenas tridimensionales e imágenes en movimiento. **[13]**

La Visión Artificial, también llamada Visión Computacional pretende con determinadas restricciones, reproducir el comportamiento del ser humano en el sentido de la vista. En un sistema de Visión Artificial se definen las siguientes cuatro fases bien diferenciadas en el proceso de imágenes. La Fig. 2.7 muestra las etapas típicas de un sistema de Visión Artificial. **[13]**

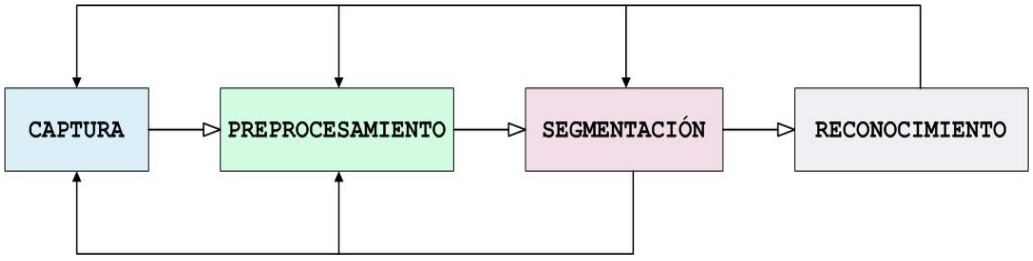


Fig. 2.7: Etapas típicas en un sistema de Visión Artificial
Fuente. Documento. "Visión Artificial" - [13].

Primera Fase: Captura o Adquisición de las imágenes digitales mediante algún tipo de sensor, fundamentalmente una cámara. Una imagen que es capturada a través de un sensor, se guarda en la memoria de la computadora como un arreglo o matriz de puntos, llamados píxeles. **[13]**

Dependiendo del tipo de imágenes que se traten, cada punto representará la intensidad o brillo de la imagen; si se trabaja con imágenes en blanco y negro. Tres matrices de puntos, uno para rojo, otro para verde y otro para azul si se trabaja con imágenes a color; su combinación representa los diferentes colores. **[13]**

Segundo Fase: Tiene por objeto el tratamiento digital de imágenes para facilitar las fases posteriores. En esta fase, conocida como Preprocesamiento, mediante filtros y transformaciones geométricas se tratan de eliminar partes indeseables de la imagen y realizar los detalles de interés para su posterior análisis. **[13]**

Segmentación: Es la siguiente fase, y consiste en particionar una imagen en regiones homogéneas con el fin de extraer características de dichas regiones (color, textura, forma, contorno, etc.) para la posterior fase de reconocimiento. **[13]**

Reconocimiento: También conocida como Clasificación. Emplea el análisis de ciertas características que se establecen previamente para diferenciar los objetos segmentados, se pretende discernir entre unos y otros. **[13]**

Diferentes aspectos han ayudado a llevar a la Visión Artificial al campo de las aplicaciones industriales como: el aumento de las prestaciones de los ordenadores, desarrollo de técnicas algorítmicas eficientes para el abordaje de los problemas, etc. Estos aspectos han permitido realizar el tratamiento de imágenes en tiempo real. **[13]**

La Visión Artificial incluye, entre otras; las siguientes aplicaciones:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes. **[13]**
- Seguimiento de objetos en secuencias de imágenes. **[13]**
- Mapeo de una escena para generar un modelo tridimensional de la escena. **[13]**
- Estimación de las posturas tridimensionales de humanos. **[13]**

- Búsqueda de imágenes digitales por su contenido. **[13]**
- Segmentación y detección de bordes. **[13]**
- Extracción de Rasgos. **[13]**
- Realidad Virtual. **[13]**
- Realidad Aumentada. **[13]**

2.2.5 Realidad Aumentada (RA)

La Realidad Aumentada es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. **[14]**

En definitiva, la RA lleva la información dentro del mundo real del usuario en vez de llevar al usuario dentro del mundo virtual del ordenador. Un sistema de RA tiene las siguientes características:

- Combinación de elementos virtuales y reales. La información se combina con la realidad. **[14]**
- Procesamiento en tiempo real. Tanto los objetos que deben ser rastreados como la información sobre estos deben proporcionarse a tiempo real. **[14]**
- Registro 3D. Los objetos reales y virtuales son registrados y alineados geométricamente entre ellos y dentro del espacio para darles coherencia espacial. **[14]**

Por objeto virtual se refiere a cualquier imagen que se genere en el dispositivo de visualización y se presente como parte de la RA, esto incluye texto, figuras, imágenes bidimensionales y modelos tridimensionales. Para desarrollar aplicaciones de RA se necesita distinguir que hardware y software es necesario:

- *Hardware:* Se utiliza un dispositivo capturador de video como una cámara y un ordenador que procese los datos. **[14]**
- *Software:* Se necesita un programa capacitado para realizar la fusión coherente del mundo real con elementos virtuales en 3D. **[14]**

El funcionamiento de esta tecnología, se puede resumir: En tiempo real, la cámara realiza un visionado de nuestra realidad buscando patrones de realidad aumentada definidos por el usuario. Cuando la cámara encuentra este patrón, la computadora procesa la perspectiva en la que el sujeto ve las cosas, y calcula e inserta elementos virtuales predeterminados en ella, haciéndolos parte de su realidad. La Fig. 2.8 muestra una estructura simplificada del funcionamiento de una aplicación con tecnología de Realidad Aumentada. **[14]**

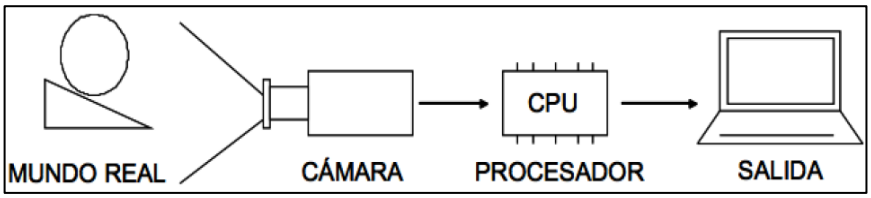


Fig. 2.8: Funcionamiento de la RA
Fuente. http://i3a.ua.es/img/proyecto_realidad_aumentada/Esquema.JPG - [14].

La Realidad Aumentada (del inglés Augmented Reality), es el término que se usa para definir una visión del mundo real combinada con elementos virtuales para la creación de una realidad a tiempo real. La Fig. 2.9 muestran ejemplos aplicativos de la RA. **[15]**

Fig. 2.9: Ejemplo de RA con el uso de gafas HDM.
Fuente. A Survey of Augmented Reality – [15].

Una de las definiciones más concretas que se han dado sobre RA, fue la descrita por Ronald Azuma en 1997, quien la define con tres características básicas: combina elementos reales y virtuales, es interactiva en tiempo real y está registrada en 3D. En forma general los elementos esenciales para la composición de un sistema de RA, son:

- Elemento captor de las imágenes reales. **[15]**

- Procesador que mezcla la información real con la creada virtualmente. **[15]**
- Proyector de las imágenes tanto reales como virtuales. Normalmente es el mismo que el elemento captor (ordenadores, teléfonos móviles, etc.). **[15]**

La definición de RA, por tanto, debe incluir siempre tres características claves; la combinación de elementos reales y virtuales, la interacción a tiempo real y la visualización de imágenes 3D. La RA puede ser usada en varios dispositivos, desde computadoras hasta dispositivos móviles. **[15]**

2.2.6 Aplicaciones de la RA

Existen muchas aplicaciones de distintos ámbitos. A continuación se describen algunas de ellas:

- *Educación:* En este campo se destaca la empresa alemana Metaio, quienes sacaron en 2010 libros educativos basados en RA. Apuntando con ellos a la cámara web de un ordenador, se superponen objetos 3D, mezclando en la pantalla ambas cosas. **[16]**
- *Medicina:* Esta tecnología podría marcar una diferencia significativa en los procedimientos quirúrgicos para extirpar los tumores de los pacientes que padezcan cáncer, pues facilitaría el trabajo de los cirujanos al brindarles la oportunidad de operar con más precisión sobre las áreas más afectadas por cierta enfermedad. **[16]**
- *Turismo:* A través de un teléfono de última generación, Wikitude muestra sobre la pantalla de éste la información sobre edificios, montañas, accidentes geográficos, etc., a las que se apunta con la cámara del móvil. Haciendo uso del GPS, la brújula y la cámara; se obtiene la información de donde se encuentra el usuario y se proporciona los datos disponibles. **[16]**

2.2.7 Tipos de RA

Es importante tener en cuenta que hay tres tipos diferentes de RA: la conocida como RA móvil, la llamada RA de escritorio y la RA mediante reconocimiento de objetos. **[17]**

Realidad Aumentada de Escritorio: Funciona con una cámara web. Sobre lo que captura la cámara, es decir; marcadores de RA, una aplicación muestra un objeto animado en 2D. **[17]**

Realidad Aumentada Móvil: Utiliza servicios basados en localización (GPS y similares) que proporciona la posición geográfica del dispositivo móvil. El problema de este modo es la poca precisión de los GPS existentes actualmente; de manera que si nuestra posición varía, aunque sea un poco, podemos ver saltar los objetos de un punto a otro. **[17]**

Realidad Aumentada mediante reconocimiento de objetos: Esta opción se basa en el reconocimiento de objetos conocidos, por ejemplo, la forma de un edificio histórico o un objeto determinado. Al reconocerlo, nos informaría sobre su perfil, su historia y sus enlaces de interés. **[17]**

2.2.8 Técnicas de Visualización

Al momento de diseñar un sistema de RA, la decisión importante que hay que tomar, es la técnica de cómo se va a llevar a cabo la combinación de lo real con la realidad virtual. Existen dos opciones básicas: ópticas y de video. **[18]**

Optical See-Through HDM: Trabaja mediante la colocación de combinadores ópticos frente a los ojos del usuario. Estos combinadores son parcialmente transmisivos, de modo que el usuario puede mirar directamente a través de ellos al ver el mundo real. Los combinadores son también parcialmente reflectante, tal que el usuario ve las imágenes virtuales rebotar en los combinadores de los monitores colocados en la cabeza. Este enfoque es similar en naturaleza a Head-Up Displays (HUD) de uso común en aeronaves militares, salvo que los combinadores se adjuntan a la cabeza. La Fig. 2.10 representa un diagrama conceptual del Optical See-Through HDM. **[18]**

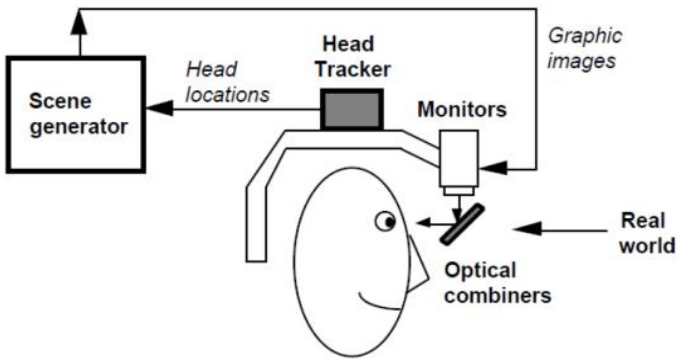


Fig. 2.10: Diagrama conceptual, Optical See-Through HDM.
Fuente. La Realidad Aumentada: Una nueva lente para ver el mundo – [18].

Video See-Through HDM: Consiste en un casco con una o dos cámaras, estas proporcionan al usuario la visión del mundo real. El video de estas cámaras se combina con las imágenes gráficas creadas por el generador de la escena, mezclando lo real y lo virtual. El resultado se envía a los monitores delante de los ojos del usuario en el HDM. La Fig. 2.11 representa un diagrama conceptual del Video See-Through HDM. **[18]**

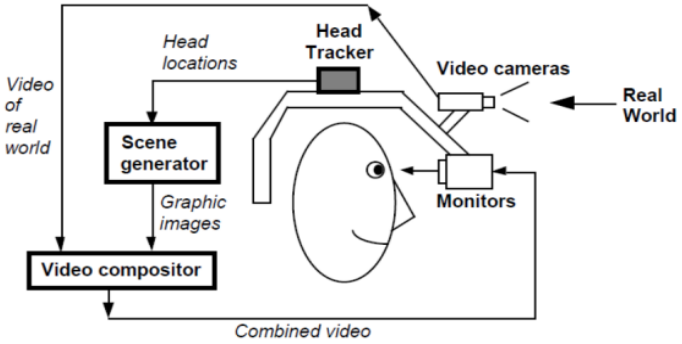


Fig. 2.11: Diagrama conceptual Video See-Through HDM.
Fuente. La Realidad Aumentada: Una nueva lente para ver el mundo – [18].

Monitor basado en RA: Los sistemas RA también pueden ser construidos con configuraciones basadas en vigilar, en lugar de ver a través de estos visores. En este caso, una o dos cámaras de video para mirar el medio ambiente. Las cámaras pueden ser fijas o móviles. En el caso de móviles, las cámaras podrían moverse por la atadura a un robot, con sus lugares de seguimiento. El vídeo del mundo real y las imágenes gráficas generadas por un generador de escena se combinan, al igual que en el caso de video HMD See-Through, y se muestran en un monitor en frente del usuario. El usuario no usa el dispositivo de visualización. Opcionalmente, las imágenes se pueden mostrar en estéreo en el monitor, el cual requiere que el usuario use un par de gafas estéreo. La Fig. 2.12 visualiza un diagrama conceptual del monitor basado en RA. **[18]**

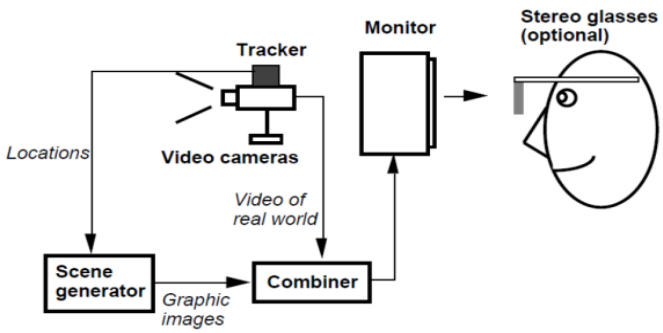


Fig. 2.12: Diagrama conceptual Monitor basado en RA.
Fuente. La Realidad Aumentada: Una nueva lente para ver el mundo – [18].

2.2.9 Software de RA

Actualmente hay una gran cantidad de software especializado para la creación de Realidad Aumentada a continuación se describen los más importantes:

BuildAR Pro: Es un programa que permite crear tus propios escenarios 3D de RA. Utiliza el seguimiento de marcadores base, lo que significa que los modelos 3D parecen adjuntados a la integridad física. Al crear un conjunto de estos marcadores y algunos modelos 3D puede crear fácilmente su propia escena de RA. Los modelos 3D se pueden hacer usando casi cualquier programa de modelado o descargar desde muchos sitios web en Internet. **[19]**

ARToolKit: Esta es una biblioteca que se usa para la creación de la RA, en donde se sobrepone imágenes virtuales en el mundo y el tiempo real, para lograr esto usa las capacidades de seguimiento de video para tener un cálculo exacto de los marcadores. Permite fácil desarrollo de una alta gama de aplicaciones de RA. **[20]**

Librerías para la creación de RA: En 1999 cuando Hirokazu Kato creó la librería ARToolKit, se han derivado diversas librerías en distintos lenguajes de programación basados en C. La mayoría de estas librerías son de tipo freeware y están diseñadas para realizar las tareas necesarias de registro, así como la composición de la escena en tiempo real. **[21]**

En las tablas desde la Tabla 2.1 hasta la Tabla 2.6 se presentan características importantes sobre las librerías creadas para realidad aumentada:

Tabla 2.1: ARToolKit

ARToolKit	Emplea técnicas de visión por computadora para calcular de manera precisa la posición y orientación de la cámara, relativa a un marcador en tiempo real.
Lenguajes	C/C++
Sistema Operativo	Windows, Mac OS X, Linux
Renderización	Soporta OpenGL
Licencia	GNU GPL
Autor	Hitokazu Kato

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

Tabla 2.2: FLARToolKit

FLARToolKit	Es una librería SDK API para la creación de Realidad Aumentada en Flash, utiliza FLARManager como wrapper.
Lenguajes	Flash, Action Script 3.0
Sistema Operativo	Windows, Mac OS X
Renderización	Paper Vision
Licencia	GNU GPL
Autor	Sagoosha

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

Tabla 2.3: ARTag

ARTag	Utiliza un procesamiento de imagen más complejo que ARToolKit y el proceso de símbolo digital para la confiabilidad y la resistencia a la luz.
Lenguajes	CSharp, C/C++
Sistema Operativo	Windows, Mac OS X, Linux
Renderización	Soporta OpenGL
Licencia	GNU GPL
Autor	Mark Fiala, IIT, NRC Canadá

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

Tabla 2.4: NyARToolKit

NyARToolKit	Puerto de ARToolKit que se ejecuta en varias plataformas entre las cuales sobresale Java, con la posibilidad de ejecutarlas en plataformas móviles.
Lenguajes	CSharp, C/C++, Java, JavaScript
Sistema Operativo	Windows, Mac OS X, Linux, Android
Renderización	Soporta Processing, Soporta OpenGL
Licencia	GNU GPL
Autor	PukiWiki Developers Team, Japón

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

NOTA: Como ARToolKit, NyARToolKit proporciona seguimiento basado en marcador. Sin embargo, el software ha sido optimizado para facilitar la portabilidad a través de diferentes lenguajes de programación. Si desea desarrollar una aplicación RA que puede ejecutarse en diferentes plataformas y sistemas operativos, NyARToolKit es

una opción ideal; NyARToolKit Professional está disponible sólo bajo licencia comercial. Sin embargo; el puerto de origen abierto original, ha sido actualizado a la versión 4.x de ARToolKit y puede ser descargado y utilizado bajo la licencia GPLv2 de libertad. **[21]**

Tabla 2.5: JARToolKit

JARToolKit	Se trata de la librería de Java que busca cubrir las mismas necesidades de ARToolKit y se encuentran escritas en lenguaje nativo C.
Lenguajes	Java, JavaScript
Sistema Operativo	Windows, Mac OS X
Renderización	Soporta Processing, Soporta OpenGL
Licencia	GNU GPL
Autor	Jorg Stocklein Tim Schmidt

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

Tabla 2.6: SLARToolKit

SLARToolKit	Es una librería para llevar aplicaciones para SilverLigth, está basado en NyARToolKit y ARToolKit.
Lenguajes	CSharp, C/C++, Silverlight
Sistema Operativo	Windows
Renderización	Soporta Processing
Licencia	GNU GPL
Autor	René Schule

Fuente. J. V. Rivadeneira Herrera, Tesis ESPE – [21].
Elaborado por. Jimena Caguana

2.2.10 Importancia en la Educación

Los alumnos aprenden directamente interactuando con los elementos virtuales. Las aplicaciones más comunes de la RA en educación son los libros basados en RA. En los libros se tiene la posibilidad de ver en 3D los elementos sobre los que se está estudiando e interactuar y modificarlos, viendo su evolución, cambiándolos, etc., y en general aprendiendo de ellos. Varios ejemplos son la visualización de un volcán en erupción, los planetas o los dinosaurios. Las posibilidades en ese sentido son infinitas. La Fig. 2.13 representa gráficamente unos ejemplos de los libros con RA. **[22]**

Fig. 2.13: Libro educativo interactivo basado en Realidad Aumentada.
Fuente. R. Gómez. Realidad Aumentada y las Pizarras virtuales – [22].

Con los libros que usan Realidad Aumentada se consigue "aumentar" la realidad consiguiendo que el alumno se implique más en el aprendizaje del contenido debido a la interactividad que tiene el mismo. **[22]**

Los contenidos que se puede encontrar en un libro aumentado son de diferentes tipos:

- 2D, con modelos estáticos y dinámicos: imágenes (fotografías, pinturas, dibujos, ilustraciones), esquemas, texto, videos y animaciones. **[22]**
- 3D, con modelos estáticos y dinámicos: puede incluir objetos y diferentes entornos. **[12]**
- Sonido: música ambiente, sonidos interactivos. **[22]**

El funcionamiento de la RA, se la puede representar de la siguiente manera: se detecta la marca, posteriormente se calcula su posición y orientación, se identifica la marca que ha detectado y por último se modela el objeto 3D. La Fig. 2.14 muestra el proceso graficado del funcionamiento de ARTolKit. **[22]**

Fig. 2.14: Funcionamiento de ARTolKit
Fuente. R. Gómez. Realidad Aumentada y las Pizarras virtuales – [22].

2.2.11 Pizarras digitales interactivas

Se puede definir la pizarra virtual como un sistema tecnológico, generalmente integrado por un ordenador, un video proyector y un dispositivo de control de puntero, que permite proyectar en una superficie interactiva contenidos digitales en un formato idóneo para visualización en grupo. Se puede interactuar directamente sobre la superficie de proyección. **[23]**

Los beneficios a la práctica docente se pueden resumir en:

- El profesor presta más atención tanto a los alumnos como a sus posibles preguntas de la asignatura que se esté aprendiendo. **[23]**
- Fomenta la espontaneidad y la flexibilidad, facilitando a los profesores una gran oferta de recursos en texto, en gráficos, en sonidos y en imágenes. Así las clases pueden ser mucho más atractivas y documentadas. **[23]**
- El profesorado aumenta su autoestima profesional al utilizar eficazmente las tecnologías avanzadas, mejorando el quehacer docente y la formación del alumnado; su actualización de conocimientos se motiva a un mejoramiento. **[23]**
- Facilita a los profesores el uso de las TIC integrándolas en su diseño curricular de aula mientras se dirigen a toda la clase manteniendo el contacto visual. **[23]**

Los beneficios para los alumnos son los siguientes:

- Aumenta la participación de los alumnos facilitando el debate. **[23]**
- Aumenta la atención y retentiva de los alumnos. **[23]**
- Es motivadora para el alumno. El compromiso del estudiante en el proceso de aprendizaje se incrementa durante el uso de una pizarra virtual. **[23]**
- Los alumnos pueden comprender conceptos más complejos gracias a las presentaciones, más claras, más dinámicas y más eficientes. **[23]**

Las pizarras virtuales resultan útiles en todas las asignaturas y niveles educativos, proporcionando muchos recursos visuales y nuevas posibilidades metodológicas que facilitan presentación y comprensión de los contenidos, el tratamiento de la diversidad, el aprovechamiento educativo de Internet, la realización de actividades más dinámicas y una mayor motivación y participación de los estudiantes. De hecho, una pizarra virtual

es una de las herramientas que permite interactuar con los alumnos en mayor medida que cualquier otro dispositivo. **[23]**

2.2.12 Realidad Aumentada y pizarras virtuales

Una vez que se ha descrito ambas tecnologías, es necesario; reconocer su integración: La pizarra virtual como tal por su capacidad de procesamiento al incluir un ordenador, permite utilizar la realidad aumentada. Así sólo se necesita un dispositivo con cámara para poder interactuar con la imagen y la aumentación de la realidad. **[23]**

2.3. Propuesta de Solución

La implementación de una pizarra virtual usando realidad aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato; favorece la educación y los procesos de enseñanza y aprendizaje con instrumentos tecnológicos e informáticos, en esta institución de formación básica y secundaria. Proporciona una respuesta positiva, exigida por los estándares de educación vanguardista de los países desarrollados tecnológicamente.

CAPÍTULO III

METODOLOGÍA

3.1. Modalidad de la Investigación

El presente Trabajo de Graduación es un proyecto científico-técnico que se guió en los siguientes tipos de investigación.

3.1.1 Investigación Aplicada

El proyecto de investigación parte de una problemática que requiere ser intervenida y mejorada, a través de una propuesta factible a la ingeniería en sistemas computacionales e informática; por tal razón la investigación es de tipo Aplicada, porque incurrió de manera tecnificada con métodos, técnicas, conceptos, principios, algoritmos, programación y prototipos concordantes con la performance de la Realidad Aumentada.

3.1.2 Investigación de Campo

La investigadora efectuó un estudio de campo en la Unidad Educativa "Tirso de Molina", para conseguir información y datos eficaces relacionados a los objetivos; por ello la investigación tuvo una modalidad de campo; también porque permitió adquirir nuevos conocimientos en el campo de la Realidad Aumentada, el Procesamiento Digital de Imágenes y Video, la Informática y la Inteligencia Artificial. Adicionalmente, porque facilitó la indagación del uso de esta tecnología para diagnosticar requisitos y solucionar problemas.

3.1.3 Investigación Bibliográfica Documental

La investigación tuvo una modalidad bibliográfica porque argumentó de manera óptima, el contenido teórico a través de diferentes fuentes primarias: repositorios digitales,

libros, revistas, publicaciones, internet y otros documentos confiables que permitieron sustentar científicamente el desarrollo de la propuesta.

La investigación es también de modo documental porque se estructuró metodológicamente por medio de fuentes primarias y secundarias. Además, porque tuvo como propósito profundizar y deducir diferentes enfoques teóricos, algoritmos técnicos, lenguajes de programación de Realidad Aumentada, criterios de diversos autores sobre la RA y sus aplicaciones en la educación interactiva con medios de presentación visual.

3.2. Población y Muestra

La presente investigación por su característica no requirió población y muestra.

3.3. Recolección de Información

La recolección de información utilizó un procedimiento integral, enfatizado en instrumentos para el registro de datos como: documentos y la observación. Desplegó una coherencia metodológica en: libros, papers, algoritmos y lenguajes de programación para Realidad Aumentada.

Al identificar un problema suscitado en los procesos de enseñanza y aprendizaje con instrumentos tecnológicos de visualización en tiempo real; y cuando la calidad educativa exige parámetros científicos de desarrollo, se dispuso implementar una pizarra virtual usando Realidad Aumentada; para con ello, evolucionar la educación a estándares de excelencia mundial.

Se recopiló información relevante sobre la pizarra virtual que utiliza Realidad Aumentada; se analizó el funcionamiento de esta tecnología y se aplicó ingeniería de software de Visión Artificial y hardware para la obtención de una pizarra virtual de alta performance. Se desarrolló en un lenguaje de programación de Realidad Aumentada prototipos, pruebas y depuración de errores; se compilaron algoritmos de Realidad Aumentada y se adaptaron a las necesidades de la pizarra virtual de la Unidad Educativa "Tirso de Molina". En el proceso concluyente se implementó la pizarra virtual en el Laboratorio de Informática principal de la institución y se capacitó al personal sobre la operatividad y mantenimiento de la misma.

3.4. Procesamiento y Análisis de datos

La información formó parte de un proceso de análisis, en el cual se trazó los datos más importantes, estipulados de la siguiente manera:

- Manejo de técnicas de investigación observación y entrevista.
- Recolección de información, algoritmos y datos técnicos.
- Análisis y depuración de la información recolectada.
- Procesamiento de la información.
- Optimización de datos e información.
- Diseño e implementación de la pizarra virtual usando Realidad Aumentada.
- Pruebas piloto y memorias técnicas.
- Presentación de resultados.

3.5. Desarrollo del proyecto

En el desarrollo y la implementación de la pizarra virtual con realidad aumentada para el aprendizaje interactivo de la Unidad Educativa "Tirso de Molina", se procedió según la siguiente estructura metodológica:

Analizar la situación tecnológica de la Realidad Aumentada destinada a la educación interactiva en instituciones de formación académica.

- Análisis de ingeniería.
  - Casos de Uso.
  - Identificación de requisitos.

Argumentar en forma computacional el método óptico y matemático; para la exposición de la Realidad Aumentada en tiempo real, sobre una pizarra.

- Diseño Conceptual.
  - Arquitectura de la pizarra virtual (hardware y software).
  - Descripción de algoritmos para RA.
- Diseño de la aplicación e interfaz gráfica de usuario.
  - Uso de librerías múltiples.
- Diseño de visión artificial.

  - Algoritmo de acceso a la cámara.
  - Calibración de la cámara, obtención del video y adquisición de imágenes.
  - Algoritmo de segmentación y detección de regiones de interés.
  - Descripción y seguimiento de las regiones detectadas.
  - Creación de menús de configuración.
  - Almacenamiento de datos en variables.
- Diseño de algoritmos de RA en las imágenes.
  - Uso de modelos 3D.
  - Algoritmo de reconocimiento basado en marcadores.
  - Cálculo de la matriz transformación.
  - Graficado de figuras digitales.
  - Inclusión de las figuras digitales.

Diseñar una pizarra virtual usando Realidad Aumentada para el aprendizaje interactivo de la asignatura de Ciencias Naturales en la Unidad Educativa "Tirso de Molina".

- Implementación de la pizarra virtual para la asignatura de Ciencias Naturales.
  - Detallar la realidad aumentada en tiempo real con la realidad física.
- Pruebas de la pizarra virtual con realidad aumentada.
- Implantación de la pizarra virtual con realidad aumentada.
- Documentación del hardware y software.
  - Desarrollo del manual de usuario.
  - Capacitación acerca del uso de hardware y software.
- Presentación del informe final del Trabajo de Graduación.

CAPÍTULO IV

DESARROLLO DE LA PROPUESTA

4.1. Datos Informativos

Tema: "Pizarra virtual usando realidad aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato".

Institución Ejecutora: Universidad Técnica de Ambato.

Institución Beneficiada: Unidad Educativa "Tirso de Molina".

Beneficiarios: Autoridades, Docentes, Alumnos y Padres de Familia.

Ubicación: La Unidad Educativa "Tirso de Molina" está ubica en la provincia de Tungurahua, específicamente a 22 Km al NE (Noreste) de la ciudad de Ambato, Av. Pedro Vásconez Sevilla S/N Barrio San Juan, de la parroquia Izamba.

Equipo Técnico Responsable:

- **Autora:** Jimena del Rocío Caguana Tibán
- **Tutor:** Ing. Mg. Clay Aldás
- **Departamento de Informática:** Lcdo. Rvdo. Luis Abad

4.2. Antecedentes de la Propuesta

El avance de la tecnología, en particular; la tecnología conocida como Realidad Aumentada (RA) tienen la función de revolucionar la manera de enseñar en las instituciones de educación, a través de la innovación en la visualización del mundo real en 3D, combinando las imágenes y el contenido que se esté aprendiendo según sea la malla curricular. Aplicar la RA en la enseñanza, conlleva múltiples ventajas; entre las

cuales, la retención eficiente de conocimientos y una mejor concentración son los parámetros de calidad que medirán el aprovechamiento de los alumnos y docentes, que utilicen una pizarra virtual usando RA. **[24]**

Un sistema de RA, desde su invención ha promulgado grandes aplicaciones en todos los ámbitos de Ingeniería Informática debido a que esta tecnología emplea algoritmos de procesamiento digital de imágenes, renderización en tiempo real de imágenes 3D, adquisición de video e imágenes a través de cámaras, algoritmos de creación de marcadores, etc.; directrices pertenecientes a Visión Artificial. La Visión Artificial permite el desarrollo de la pizarra visual usando RA, las prestaciones de la RA convierten la educación en interactiva; entre el docente, alumno y sistema RA (Hardware-Software). **[24]**

Si se regresara en el tiempo, ha aquel 1968; en el que Ivan Sutherland crea el primer sistema de realidad aumentada, que también es el primer sistema de realidad virtual, utiliza una interfaz óptica a través de un display montado en la cabeza. Que sigue por medio de uno de los dos diferentes 6DOF sensores (un seguidor de movimiento y un seguidor de ultrasonido). Debido a limitado poder de procesamiento de las computadoras de ese momento, solo dibujos simples podían ser mostrados en tiempo real. **[24]**

En 1992 Tom Caudell y David Mizell se acreditan el término "realidad aumentada" para referirse a la superposición de material que un computador presenta sobre el mundo real. En 1993, Fitzmaurice crea "Chameleon", un ejemplo clave de la visualización de la información situada en un ámbito real, ayudado con un dispositivo de mano. El dispositivo consta de una pantalla de 4 pulgadas conectada a una cámara de video a través de un cable; la cámara graba el contenido de una pantalla de una estación de trabajo, con el fin de que se muestre en la pantalla del dispositivo. **[24]**

En el año 1995, Junio Rekimoto y Nagao Katashi crean "NaviCam", una aplicación conectada a un computador, similar al "Chameleon" de Fitzmaurice. El NaviCam también utiliza una estación de trabajo cerca, pero tiene una cámara montada en la pantalla del dispositivo móvil que se utiliza para el seguimiento óptico. La computadora detecta código de colores en los marcadores de la imagen de la cámara en tiempo real, y

muestra información referida al contexto directamente en la parte superior del canal de video. **[24]**

En 1996 Junio Rekimoto presenta marcadores en 2D en forma de matrices (de forma cuadrada, como un código de barras), uno de los primeros sistemas de marcadores que permiten el seguimiento de la cámara con 6DOF. Ronald Azuma en 1997 presenta el primer estudio sobre Realidad Aumentada. En su publicación Azuma proporciona la definición ampliamente reconocida la RA e identifica tres características: combina lo real con lo virtual, es interactivo en tiempo real, está registrado en tiempo real. **[24]**

Uno de los hitos más importantes para la realidad aumentada se da en 1999, cuando Hirokazu Kato y Mark Billinghurst presentaron el ARToolKit, plantean una biblioteca de seguimiento con 6DOF, utilizando marcadores fiduciales (objeto que es usado en el campo de visión de un sistema de imágenes, que aparece en la imagen producida, para ser usado como un punto de referencia) y un enfoque basado en una plantilla para el reconocimiento. ARToolKit está disponible como código abierto bajo licencia GPL y sigue siendo muy popular en la comunidad de RA. **[24]**

En 2003 Ramesh Raskar presenta iLamps. En este trabajo se ha creado un primer prototipo para Realidad Aumentada con un dispositivo de mano que posee una cámara. Un mejorado proyector, puede determinar y responder a la geometría de la superficie, y puede ser utilizado en una red AD-HOC para crear una pantalla auto configurada. **[24]**

La aplicación más conocida de Realidad Aumentada aplicada en la educación es el proyecto Magic Book del grupo activo HIT de Nueva Zelanda. El usuario lee un libro real a través de un visualizador de mano y mira sobre las páginas reales contenidos virtuales. De esta manera cuando el alumno ve una escena de RA que le gusta puede introducirse dentro de la escena y experimentarla en un entorno virtual. **[24]**

Ineludiblemente el ámbito de la Realidad Aumentada en la educación, está congregado a la calidad de enseñanza con tecnología de vanguardia; en respuesta a la configuración de Visión Artificial, con el propósito de mejorar el aprendizaje y la percepción de conocimientos. La implantación de una pizarra virtual usando Realidad Aumentada corrige los puntos vulnerables y requerimientos de educación, en la etapa pedagógica de

las imágenes y el contenido de las asignaturas dictadas por los docentes de la Unidad Educativa “Tirso de Molina”.

La implantación de una pizarra virtual usando Realidad Aumentada en la Unidad Educativa “Tirso de Molina”, promoverá esporádicamente el desarrollo y la evolución de las instituciones educativas en el Ecuador, con tecnología factible a la calidad de enseñanza regida a la eficiencia internacional. El país está impulsando proyectos de crecimiento tecnológicos, y crear una pizarra virtual usando Realidad Aumentada para el servicio educativo de la institución antes mencionada, evoluciona el aprendizaje en la comunidad educativa ecuatoriana. La propuesta, en su magnitud; contempla los cambios científicos y tecnológicos propios de la Visión Artificial y la portabilidad informática.

4.3. Justificación

Una pizarra virtual usando Realidad Aumentada, es preponderante implementarlo en el Laboratorio de Informática de la Unidad Educativa “Tirso de Molina”; no solo por ser una propuesta científica; también por ser un proyecto de ingeniería, totalmente social y de desarrollo, vinculante entre la Universidad Técnica de Ambato y el establecimiento mencionado. Una pizarra virtual con RA impulsará la calidad educativa y tecnológica de este centro de estudios, mejorará la metodología pedagógica en el estudio de los alumnos y brindará la motivación del educando a la ciencia y tecnología.

Implementar una pizarra virtual usando Realidad Aumentada es la mejor solución tecnológica a las exigencias internacionales de calidad educativa, además; a la necesidad de ingeniería computacional que solicita las nuevas invenciones de hardware electrónico, para potenciar la ciencia y el conocimiento. Los algoritmos de Visión Artificial, las librerías de procesamiento digital de imágenes o video, los códigos de programación en entornos de Realidad Aumentada y las Interfaces Gráficas de Usuario (GUI), se convierte en un componente fundamental para el funcionamiento de la Realidad Aumentada y su aplicación en los procesos de enseñanza y aprendizaje consolidando una interactividad en la educación. Los alumnos consientes de la tecnología, en la actualidad buscan proyectos técnicos que se vinculen a sus equipos como: Smartphone, tabletas, pda’s, laptops y dispositivos multimedia; son temas de su interés, que les motivan a aprender, a conocer y saber; para no sentirse rezagados de la

ciencia y la técnica. Una pizarra virtual usando realidad aumentada es categoría atrayente a las diversas directrices de la Ingeniería en Sistemas Computacionales e Informáticos. Estar cohibidos a tecnologías de nivel mundial, limita el desarrollo científico de los centros de educación y sus talentosos educandos.

Produce carencia creativa la metodología pedagógica tradicionalista, en la que el educador se limita a la enseñanza obligatoria del contenido textual de su guía o libro de apoyo; las nuevas tecnologías ya no permiten esos procesos de enseñanza y aprendizaje vetustos, que solo causan el aburrimiento y el fracaso educativo en las instituciones. Una pizarra virtual usando realidad aumentada es un referente que conforta la calidad educativa en el Ecuador; su implementación provocará la imitación sistemática de otros centros de estudio en la utilidad de esta tecnología. Trascender a la RA, es una necesidad imperante y consecuente con los parámetros que presiden su invención.

4.4. Objetivos

4.4.1 Objetivo General

- ✓ Implementar una pizarra virtual usando realidad aumentada para el aprendizaje interactivo en la Unidad Educativa "Tirso de Molina", de la ciudad de Ambato.

4.4.2 Objetivos Específicos

- Puntualizar las etapas, parámetros y arquitectura necesaria para la implementación de una pizarra virtual usando Realidad Aumentada.
- Argumentar contextualmente las herramientas informáticas para solventar las necesidades pertinentes a Visión Artificial y Realidad Aumentada.
- Desarrollar una aplicación de Realidad Aumentada con bases pedagógicas que sea empleada, a través de una pizarra; como herramienta para la educación interactiva.
- Vincular software informático y hardware que se ayude de lenguajes orientados a objetos y librerías de Visión Artificial; demostrando el procesamiento digital de imágenes que la Realidad Aumentada utiliza.
- Fundamentar la creación de modelos de imágenes digitales y marcadores para la funcionalidad de la aplicación final; justificando la programación y la finalidad de establecer un software usando técnicas de Realidad Aumentada.

- Documentar la ingeniería de software y la programación informática constituyente a la pizarra virtual usando Realidad Aumentada.
- Desarrollar un Manual de Usuario para la pizarra virtual de Realidad Aumentada.
- Emprender la respectiva capacitación del Departamento de Informática, comprometido con el soporte y mantenimiento de la pizarra virtual.

4.5. Análisis de Factibilidad

4.5.1 Factibilidad Operativa

Las ventajas en educación de la pizarra virtual con Realidad Aumentada son cuantiosas, debido a que desarrolla mayor efectividad y eficacia en los procesos de enseñanza y aprendizaje, incrementa la asimilación de conocimientos en base a una mejor concentración de los alumnos, evoluciona el material multimedia, consolida la percepción visual en un ángulo de 360 grados (2D) de las imágenes 2D, sumerge al alumno en el contenido de estudio en forma virtual y en tiempo real. Innegablemente mejoran la formación integral del alumno, la creatividad y la motivación por la tecnología; refrendan calidad a las instituciones educativas.

4.5.2 Factibilidad Temporal

La factibilidad temporal estimada para la implementación de la pizarra virtual usando Realidad Aumenta, es el concordante con el levantamiento y recopilación de información en la Unidad Educativa "Tirso de Molina"; además del desarrollo de la ingeniería de software y la programación de la aplicación final. Se contempla que el tiempo invertido en el Trabajo de Graduación concuerde con el tiempo de aprobación estipulado por el Honorable Concejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial; aproximadamente seis meses, espacio suficiente para concluir con la propuesta.

4.5.3 Factibilidad Técnica

La propuesta es factible de manera técnica, debido a que la Unidad Educativa "Tirso de Molina" cuenta con el equipo informático ideal para la implementación de la pizarra virtual con Realidad Aumentada; posee un laboratorio de computación con ordenados

de vanguardia, cámaras web, proyectores multimedia; además de la infraestructura concordante a la iluminación de proyectos de Visión Artificial y proyección de video o imágenes a pizarras y pantallas.

4.5.4 Factibilidad Económica

La propuesta pizarra virtual usando Realidad Aumentada tiene apertura positiva en la factibilidad económica, porque la institución beneficiada cuenta con el presupuesto necesario y el equipo informático indispensable, adquirido en su reciente estructuración tecnológica. La Unidad Educativa "Tirso de Molina" sabe que invertir en este proyecto, es promover estándares de calidad educativa y mejorar los procesos de enseñanza y aprendizaje. El Rector de la institución educativa apoya la iniciativa, conoce el desarrollo que acompaña esta tecnología para la formación de los alumnos, y está motivado con la implementación de la misma.

4.5.5 Proyección a Futuro

A partir de la implementación de la pizarra virtual usando Realidad Aumentada denotada en Visión Artificial, la proyección a futuro; tecnifica los procesos de enseñanza y aprendizaje, facilita la educación con material multimedia, mejora la interactividad entre docente, asignatura y alumno. A mediano y a largo plazo, la Unidad Educativa "Tirso de Molina", se convertirá en una de las cuatro instituciones educativas en incursionar esta tecnología; tres de ellas a nivel de educación superior, ninguna a nivel de educación primaria y secundaria. Además, la pizarra virtual con Realidad Aumentada tipifica el bienestar educativo en su performance curricular.

4.6. Fundamentación Científico-Técnica

A continuación se presentan los contextos de ingeniería, metodología, procesos, algoritmos, diagramas UML; antecedentes y argumentos técnicos útiles para la implementación de la Pizarra Virtual usando Realidad Aumentada. La Fig. 4.1 muestra un diagrama de bloques representativo al diseño de la pizarra virtual.

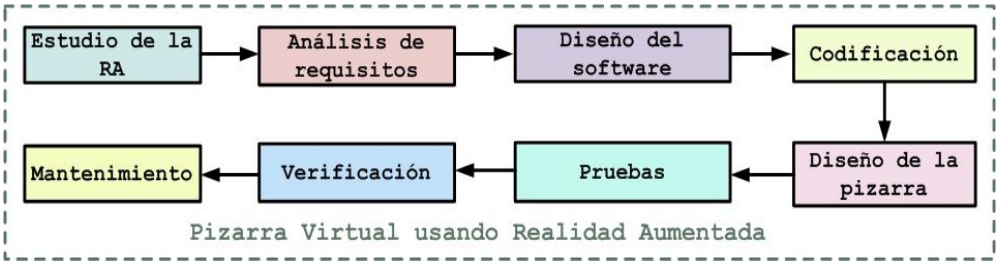


Fig. 4.1: Arquitectura de la Pizarra Virtual usando Realidad Aumentada

4.6.1 Estudio de la Realidad Aumentada

El desarrollo de la Pizarra Virtual usando Realidad Aumentada propone la disertación técnica entre el término realidad virtual y realidad aumentada para tener evidente el objeto de la Realidad Aumentada en el ámbito educativo y los procesos de enseñanza y aprendizaje.

Realidad Virtual: Es una tecnología que permite al usuario sumergirse en una simulación gráfica 3D, generada por computador y navegar e interactuar en ella en tiempo real, desde una perspectiva centrada en el usuario. La Realidad Virtual es una experiencia sintética mediante la cual se pretende que el usuario sustituya la realidad física por un entorno ficticio generado por computador. **[25]**

Realidad Aumentada: Es una tecnología que complementa la percepción e interacción con el mundo real y permite al usuario estar en un entorno real aumentado con información adicional, generada por el computador. A su vez, la RA es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, que se combinan con elementos virtuales. La principal diferencia con la Realidad Virtual es la agrupación de dispositivos que añaden información virtual a la información física ya existente, es decir; sobreimprime los datos informáticos al mundo real. **[25]**

Algunas características de la Realidad Aumentada son:

- *Combina lo real y lo virtual:* La información digital es combinada con la realidad.
- *Funciona en tiempo real:* La combinación de lo real y lo virtual se hace en tiempo real. **[25]**
- *Registra en tres dimensiones:* En general la información aumentada se localiza o registra en el espacio. Para conservar la ilusión de ubicación real y virtual, ésta

última tiende a conservar su ubicación o a moverse respecto a un punto de referencia en el mundo real. **[25]**

4.6.2 Análisis de requisitos

En la ingeniería de software el punto de partida es el análisis de requisitos; condición que orienta la metodología a desarrollar por la aplicación informática para automatizar algún tipo de proceso, en el caso de la Pizarra Virtual; la Visión Artificial y la Realidad Aumentada. El mundo computacional proporciona herramientas CASE (Ingeniería de Software Asistida por Computador) para la diagramación UML (Lenguaje Unificado de Modelado) como parte fundamental en la creación de sistemas computacionales.

Casos de uso: El siguiente apartado del trabajo de graduación, presenta los requisitos funcionales de la Pizarra Virtual usando Realidad Aumentada en forma de casos de uso y diagramas de flujo.

Caso de Uso "Identificar Marcador"

La Fig. 4.2 muestra el caso de uso "Identificar Marcador".

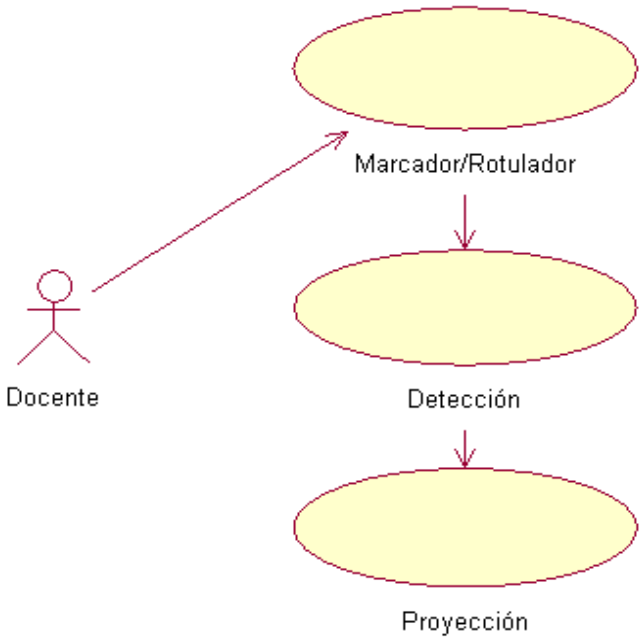


Fig. 4.2: Caso de Uso "Identificar Marcador"

La Tabla 4.1 detalla los procesos del caso de uso "Identificar Marcador".

Tabla 4.1: Procesos del Caso de Uso "Identificar Marcador"

Nombre	Identificar Marcador
Descripción	Permite identificar el arte del marcador para proyección de RA del modelo 3D.
Actores	Docente
Precondiciones	El Docente ha iniciado el sistema de RA.
Flujo Normal	El usuario dibuja con un rotulador, el arte del marcador en la pizarra virtual usando RA para reconocimiento del sistema.
Flujo Alternativo	El usuario muestra un marcador prediseñado junto a la pizarra virtual usando RA para reconocimiento del sistema.
Postcondiciones	La información que entrega el marcador es reconocido por el sistema para la proyección de RA del respectivo modelo 3D.

Elaborado por. Jimena Caguana

La Fig. 4.3 muestra el diagrama de flujo para este caso de uso.

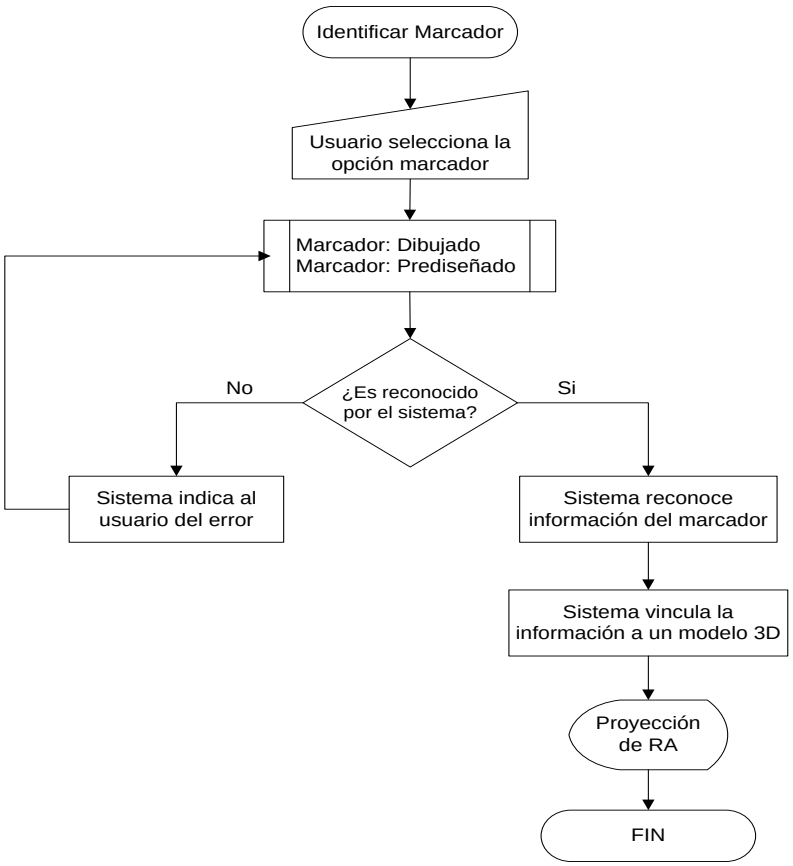


Fig. 4.3: Diagrama de flujo "Identificar Marcador"

Caso de Uso "Demandar modelo 3D"

La Fig. 4.4 muestra el caso de uso "Demandar modelo 3D".

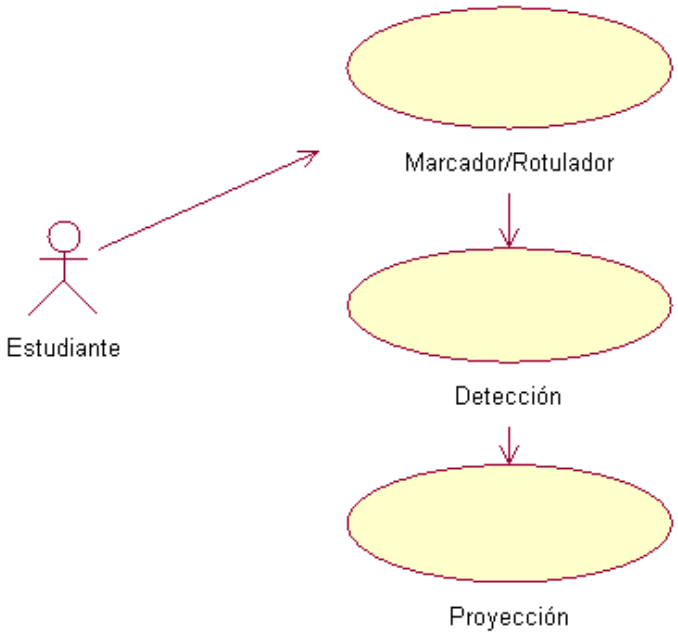


Fig. 4.4: Caso de Uso "Demandar modelo 3D"

La Tabla 4.2 detalla los procesos del caso de uso "Demandar modelo 3D".

Tabla 4.2: Procesos del Caso de Uso "Demandar modelo 3D"

Nombre	Demandar modelo 3D
Descripción	Permite demandar el respectivo marcador para la proyección de RA del modelo 3D.
Actores	Estudiante
Precondiciones	El Estudiante solicita la proyección de un modelo 3D de RA a su interés de aprendizaje.
Flujo Normal	El usuario dibuja con el roturador, el arte del marcador en la pizarra virtual usando RA para reconocimiento del sistema.
Flujo Alternativo	El usuario muestra un marcador prediseñado junto a la pizarra virtual usando RA para reconocimiento del sistema.
Postcondiciones	La información que entrega el marcador es reconocido por el sistema para la proyección de RA del respectivo modelo 3D.

Elaborado por. Jimena Caguana

NOTA: Los Casos de uso "Identificar Marcador" y "Demandar modelo 3D"; proponen el funcionamiento de la Pizarra Virtual usando Realidad Aumentada en el ámbito de la Visión Artificial y la RA explorando así estas tecnologías en la educación.

La Fig. 4.5 muestra el diagrama de flujo para este caso de uso.

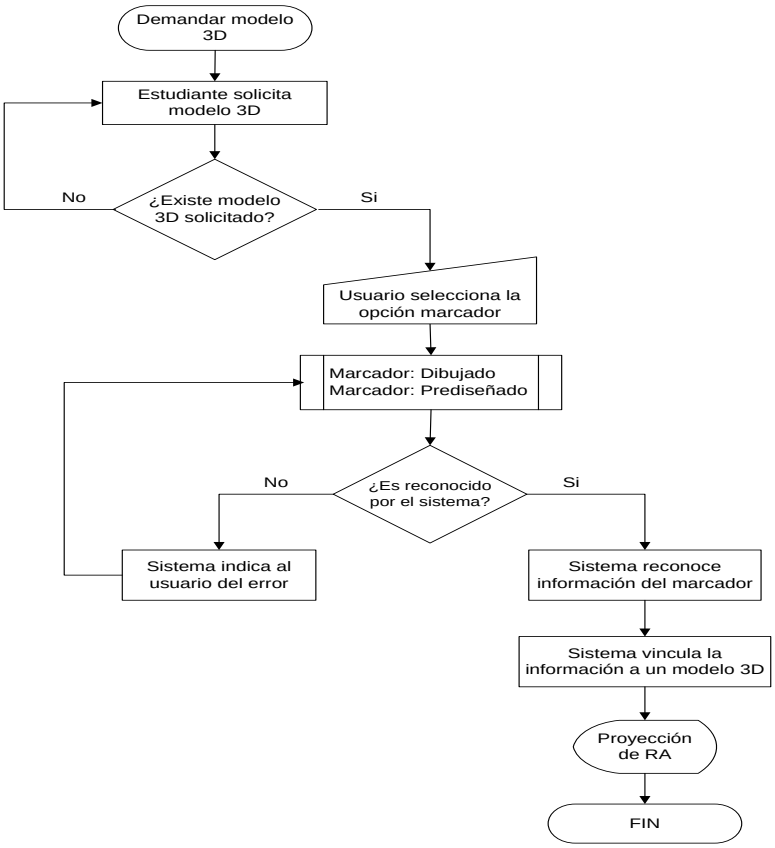


Fig. 4.5: Diagrama de flujo "Demandar modelo 3D"

Caso de Uso "Explorar Aprendizaje"

La Fig. 4.6 muestra el caso de uso "Explorar Aprendizaje".

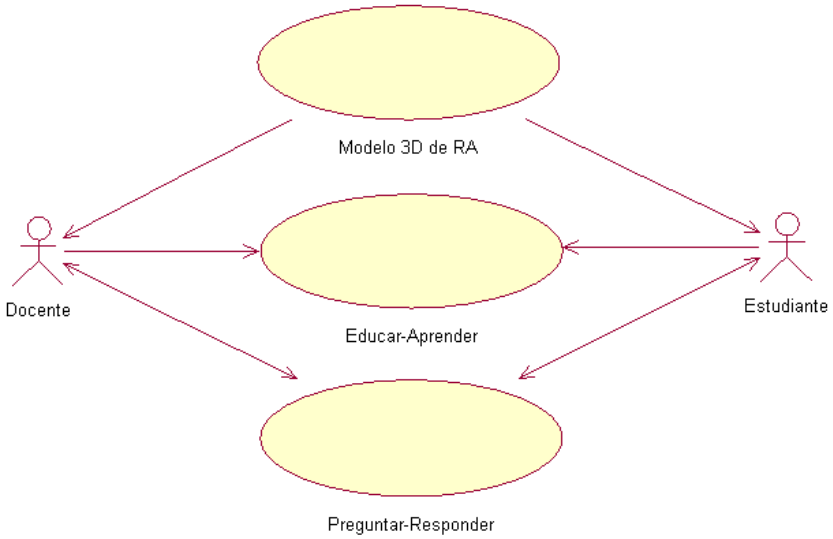


Fig. 4.6: Caso de Uso "Explorar Aprendizaje"

La Tabla 4.3 detalla los procesos del caso de uso “Explorar Aprendizaje”.

Tabla 4.3: Procesos del Caso de Uso “Explorar Aprendizaje”

Nombre	Explorar Aprendizaje
Descripción	Permite el aprendizaje interactivo a través de la proyección de RA del modelo 3D.
Actores	Docente y Estudiante
Precondiciones	Proyección de un modelo 3D de RA en ejecución.
Flujo Normal	El Docente educa el contenido de la asignatura referente al modelo 3D de RA; y el Estudiante aprende dicho contenido.
Flujo Alternativo	El Estudiante realiza preguntas exploratorias del contenido de la asignatura aprendida con el uso del modelo 3D de RA. El Docente evalúa el contenido de la asignatura aprendida con el uso del modelo 3D de RA.
Postcondiciones	Uso de un proyector multimedia para una mejor percepción visual de contenido.

Elaborado por. Jimena Caguana

La Fig. 4.7 muestra el diagrama de flujo para este caso de uso.

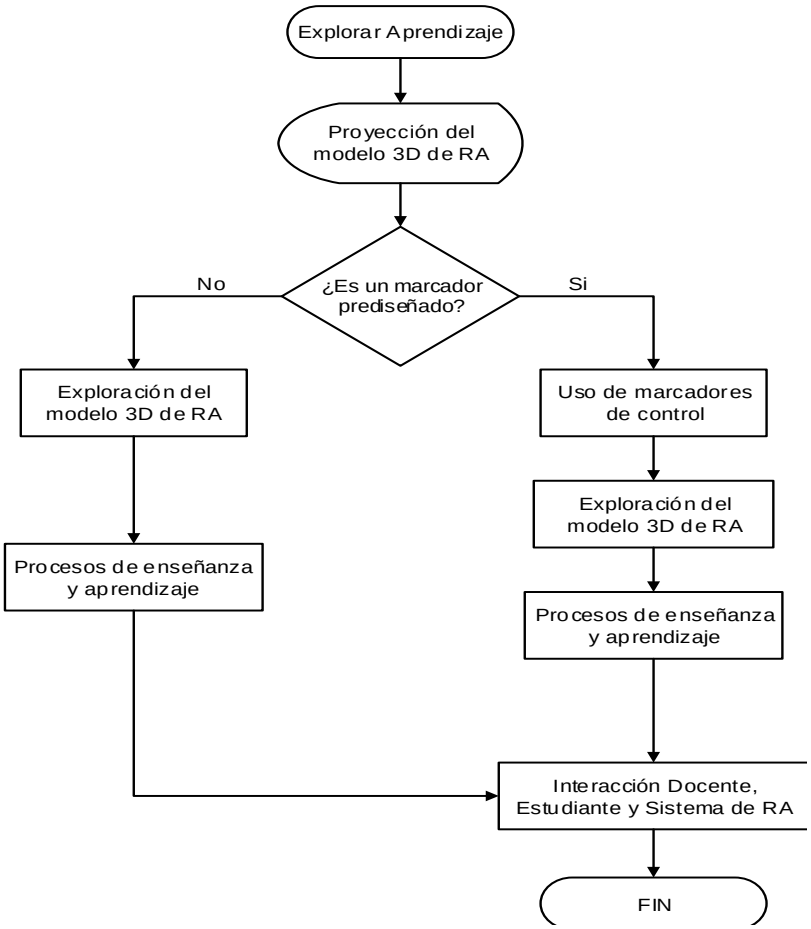


Fig. 4.7: Diagrama de flujo "Explorar Aprendizaje"

NOTA: El uso de marcadores por medio del rotulador o un marcador impreso; además del caso de uso "Explorar Aprendizaje", efectivizan la interactividad de la Pizarra Virtual usando Realidad Aumentada.

4.6.3 Diseño del software

Marcadores: Los sistemas de Realidad Aumentada (RA) generalmente cuentan con marcadores, que es una imagen impresa o mostrada en un dispositivo de salida de video. De ser impresa esta puede contener información adicional del modelo de RA en el dorso de la misma. **[26]**

Cuando se exhibe el marcador delante de un sistema de captura de imágenes que contenga la correspondiente aplicación de RA; lo que hace el software, es montar la imagen virtual al marcador. **[26]**

Para el correcto funcionamiento debe contener sistemas inerciales y ópticos que sean capaces de medir características como son la aceleración y la orientación; así como el ángulo de inclinación. De esta manera, el modelo cambiará su tamaño dependiendo de la distancia donde se ubique el marcador con respecto a la cámara, el modelo también tendrá que ser capaz de girar si el marcador gira y otras características de movimiento. La Fig. 4.8 muestra un marcador con elementos básicos. **[26]**

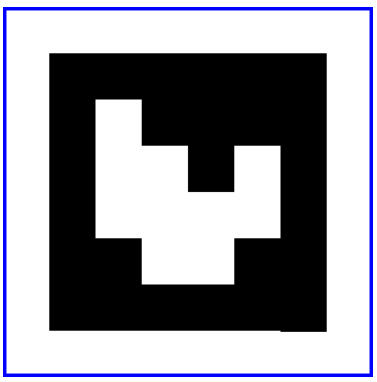


Fig. 4.8: Marcador con elementos básicos

Para desempeñar las respuestas a estas características el marcador debe ser asimétrico, tanto horizontal como verticalmente; del mismo modo, el marcador no debe ser tan

complejo, ya que esto empeora el tiempo de respuesta del sistema, por lo que se sugiere que el marcador tenga elementos básicos. **[26]**

Un marcador ideal es una figura cuadrada de alto contraste, en cuyo interior se encuentra el arte del marcador, lo que le diferencia del resto de marcadores y le proporciona la información necesaria a la aplicación para el procesamiento de la escena. La Fig. 4.9 muestra un marcador cuadrado de alto contraste. **[26]**

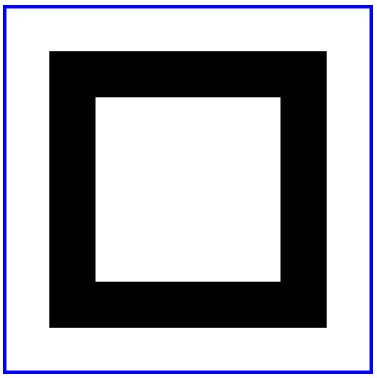


Fig. 4.9: Marcador cuadrado de alto contraste

Creación del arte del marcador: El arte del marcador consiste en figuras preferiblemente de color negro dentro del marcador ideal. El contenido del marcador, debe su nombre a que el desarrollador podrá crear a su gusto las figuras dentro del mismo y podrá dictar sus propios protocolos al momento de crear marcadores en serie. La Fig. 4.10 muestra un ejemplo de diversos marcadores creados por Nintendo para PokéDex 3D. **[26]**

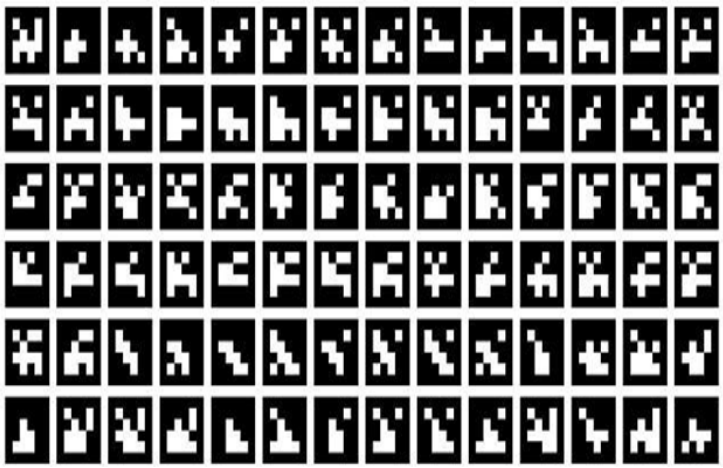


Fig. 4.10: Diversos marcadores creados por Nintendo para PokéDex 3D

Fuente. https://theworldwentaway.files.wordpress.com/2011/11/19852.jpg.

Para la creación de este arte se ha elegido el programa GIMP 2.8. Este programa es un editor de imágenes profesional de código abierto y multiplataforma (originario de Linux). Con versiones para los sistemas operativos más conocidos (Windows, Linux y Mac Os). Este programa trabaja con capas y herramientas profesionales útiles para crear el arte del marcador. La Fig. 4.11 muestra la ventana de trabajo del GIMP. **[26]**

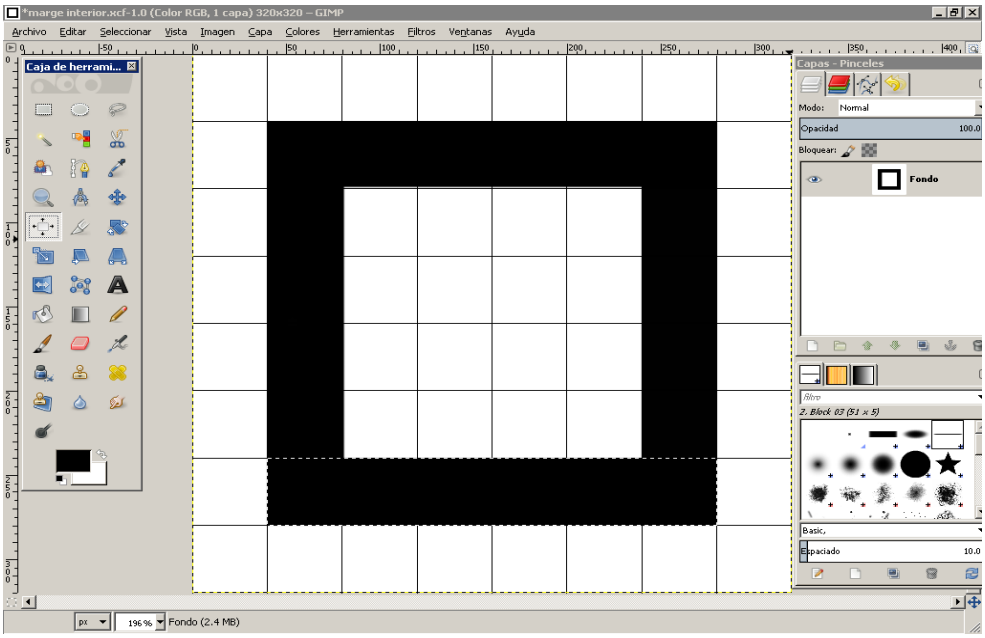


Fig. 4.11: Ventana de trabajo del GIMP 2.8

Para la creación del marcador ideal (totalmente blanco en su interior), se procede a crear una capa de 320x320 píxeles, en esta capa se crea una grilla con espacios de 40x40 píxeles, esto para construir los dos diferentes márgenes del marcador. La Fig. 4.12 muestra la grilla creada en el fondo de 320x320, con espacios de 40x40 píxeles. **[26]**

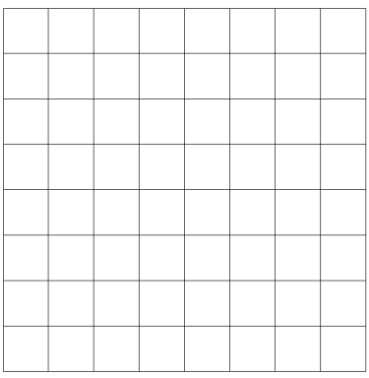


Fig. 4.12: Grilla creada en el fondo de 320x320

El primer margen corresponde a uno blanco de 40 píxeles en el contorno exterior del marcador, el segundo margen es uno de 40 píxeles al interior del primer margen, este es completamente negro. Después se procede a borrar la grilla del margen exterior, quedando el marcador identificado con su margen exterior y su margen interior. La Fig. 4.13 muestra el margen interior y margen exterior de un marcador. **[26]**

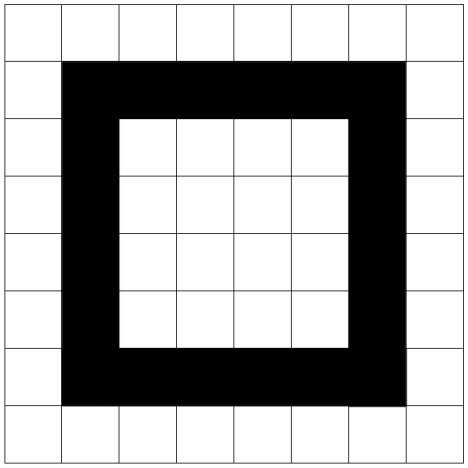


Fig. 4.13: Margen interior y margen exterior de un marcador

La Fig. 4.14 muestra el margen exterior sin grilla.

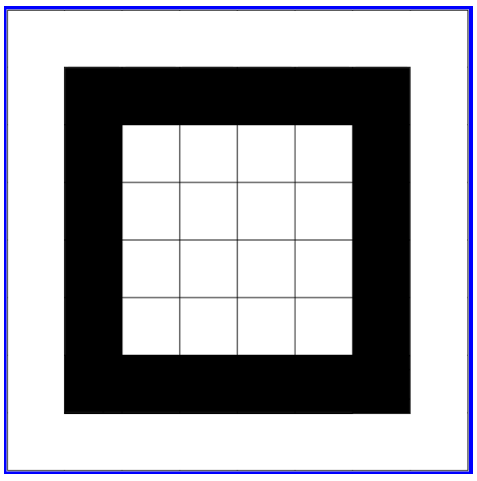


Fig. 4.14: Marcador con margen exterior sin grilla

Modelos: Los modelos son figuras en 2D o 3D en el ámbito virtual, que se sobreponen a un marcador; la mayoría de modelos para RA son en 3D y algunos muestran figuras con posibilidad de ser animadas. El modelo en 3D, es el arte de diseñar objetos en 3D, por lo general utilizando algún tipo de software de diseño. **[26]**

Es una nueva área de desarrollo tecnológico, es una manera de interactuar con el mundo real adoptando modelos muy útiles para explicar cosas que anteriormente solo se podrían ver en una imagen 2D, sin interacción alguna. Se están desarrollando en las áreas de publicidad y marketing, y en campos como la medicina, ingeniería y arquitectura; donde la manipulación de modelos en 3D puede ser muy informativa. Un modelo en 3D sobre un marcador se muestra en la Fig. 4.15, donde se aprecia la mezcla de la realidad con lo virtual. **[26]**

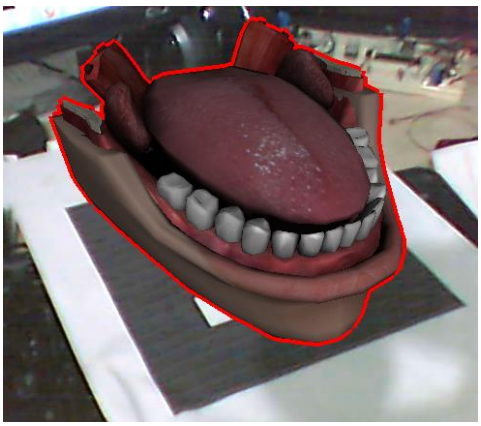


Fig. 4.15: Mezcla de la realidad con lo virtual
Fuente. https://ingenieriaaumentada.com/b5353-sintc3adtulo.png - [26].

Método del arte del modelo: El arte del modelo consiste en figuras tridimensionales que contienen información necesaria para cumplir con los objetivos de la aplicación. La mayoría de modelos son en 3D pudiendo ser estos animados o no, también existen modelos en 2D con características similares a los de 3D. Para la creación de estos modelos se ha elegido el programa Autodesk 3Ds Max. **[26]**

Autodesk proporciona software de modelado potente, integrado en 3D, animación, renderizado y herramientas que permiten a los artistas y diseñadores, centrarse más en la parte creativa; en vez de los desafíos técnicos. Ofrece conjunto de herramientas especializadas para los desarrolladores de juegos, artistas de efectos visuales y artistas gráficos en movimiento junto con otros profesionales creativos que trabajan en la industria del diseño de los medios de comunicación. **[26]**

Además ofrece potentes funciones para realizar modelos en 3D con calidad profesional. El eficaz conjunto de herramientas creativas de Autodesk 3Ds Max le ayudará a crear mejor contenido 3D en menos tiempo. La ventana de trabajo de este software muestra

cuatro visualizaciones del modelo, una vista superior, una vista frontal, una vista de la parte izquierda y una percepción realística. La Fig. 4.16 muestra la ventana de trabajo de Autodesk 3Ds Max. **[26]**

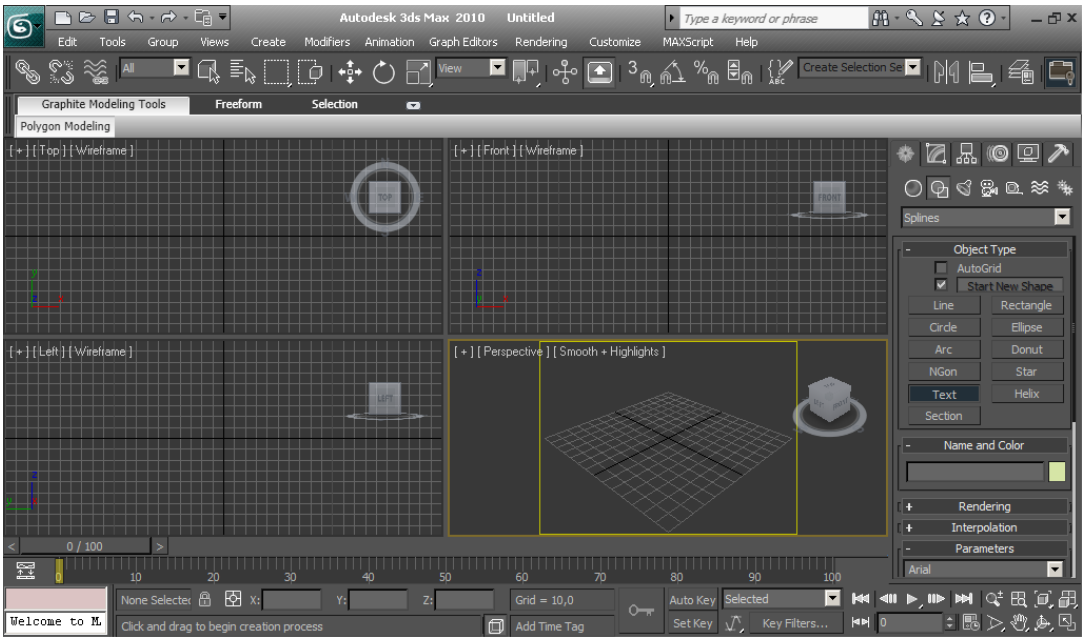


Fig. 4.16: Ventana de trabajo de Autodesk 3Ds Max

Existe diversidad de herramientas informáticas encargadas de generar modelos, se eligió este software, ya que en la página oficial se dispone de descargas gratuitas para estudiantes. Además, del manejo simple junto con todas las facilidades que el software proporciona al usuario, para enfocarse en la parte artística del modelo. **[26]**

Creación del modelo: Para explicar la creación del modelo se partirá de un modelo simple; se realizará un texto que gire en cuatro tiempos, 90 grados cada tiempo, esto da la impresión de que el texto se encontrará girando sobre su propio eje. Para esto se debe conocer las herramientas básicas del software. **[26]**

En la parte derecha de la ventana de trabajo se puede observar un frame de trabajo, con opciones como créate, modify, hierarchy, motion, display y utilities. Cuando se presiona cualquier botón, se despliega opciones propias de lo que ha sido presionado, por ejemplo en create, se despliega un frame que contiene los diferentes tipos de objetos que se puede crear, para este caso se elige shapes. Al elegir shape, el frame vuelve a cambiar para mostrar las propiedades del objeto que se ha elegido, se elige en este caso text, con

lo que el frame vuelve a cambiar, a continuación se llena el cuadro de texto etiquetado como text: con lo que el usuario prefiera para su modelo, para esto se escribe la palabra FISEI, y se da un clic en área de visualización realística, se puede observar el texto dentro del modelo. La Fig. 4.17 muestra el frame de trabajo de Autodesk 3Ds Max.

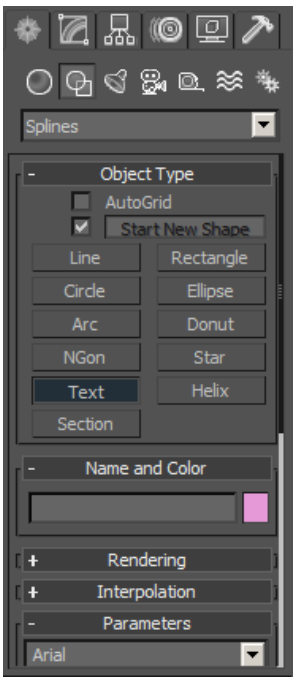


Fig. 4.17: Frame de trabajo de Autodesk 3Ds Max

La Fig. 4.18 muestra el texto en el modelo en Autodesk 3Ds Max.

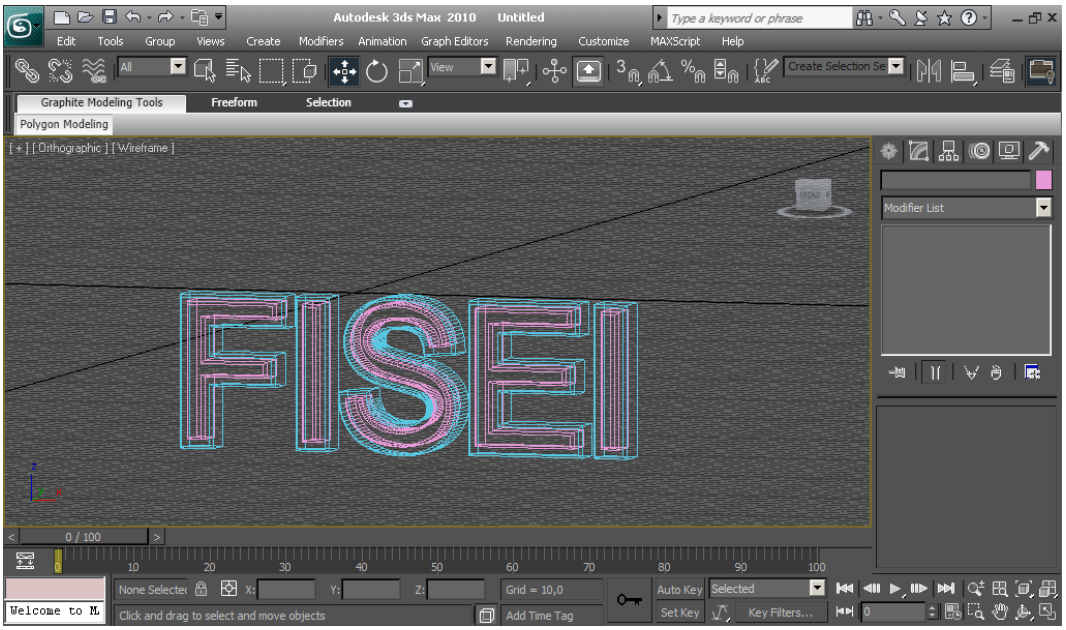


Fig. 4.18: Texto en el modelo en Autodesk 3Ds Max

Tres herramientas de la barra de herramientas principal Autodesk 3Ds Max, son necesarias para la modificación de este texto. En la Fig. 4.19 se muestra las herramientas "seleccionar y mover", "seleccionar y rotar" y "seleccionar y unificar la escala" de izquierda a derecha respectivamente.

Fig. 4.19: Herramientas básicas de modificación

Luego de ubicar el texto en una posición de 90° con respecto a la plantilla, se procede a dar espesor al texto en el frame de trabajo con la opción modify, el frame cambia una vez más, en esta ocasión muestra una barra de opciones de la cual se escoge Extrude, y se cambiará su parámetro Amount, que para este caso se ha alterado con el número 10. La Fig. 4.20 muestra la modificación básica del modelo (ubicación, giro y tamaño). **[26]**

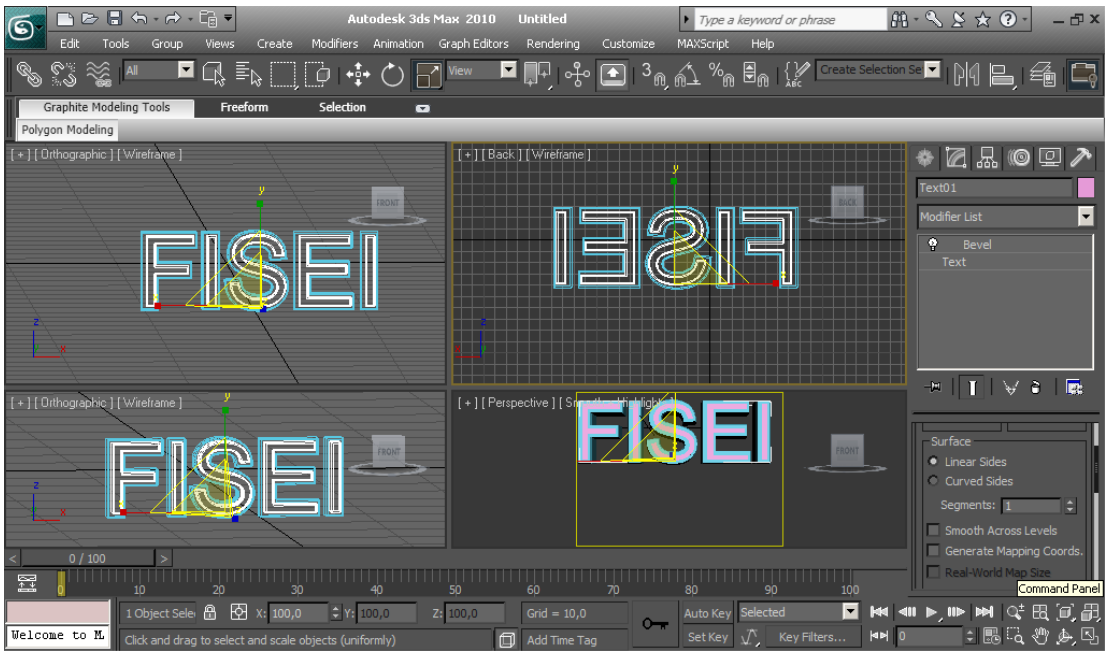


Fig. 4.20: Modificación básica del modelo

Se procede a dar color con "rendering", ubicado en la barra de opciones, se selecciona Material Editor y después compact material editor, donde se desplegará una ventana con seis muestras en parte superior, se altera las opciones de Ambient para dar el color deseado y a continuación se procede arrastrar el ejemplo para el modelo previamente

creado. La Fig. 4.21 muestra la modificación del ancho del modelo. La Fig. 4.22 muestra la opción Material Editor, que modifica el color del modelo. **[26]**

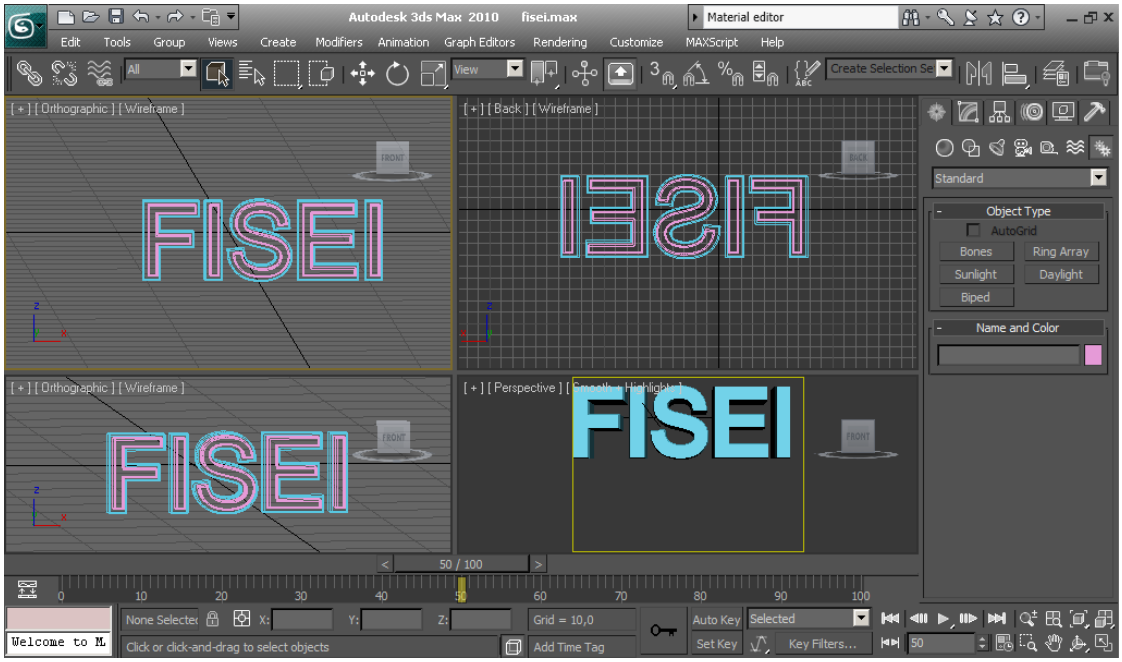


Fig. 4.21: Modificación del ancho del modelo

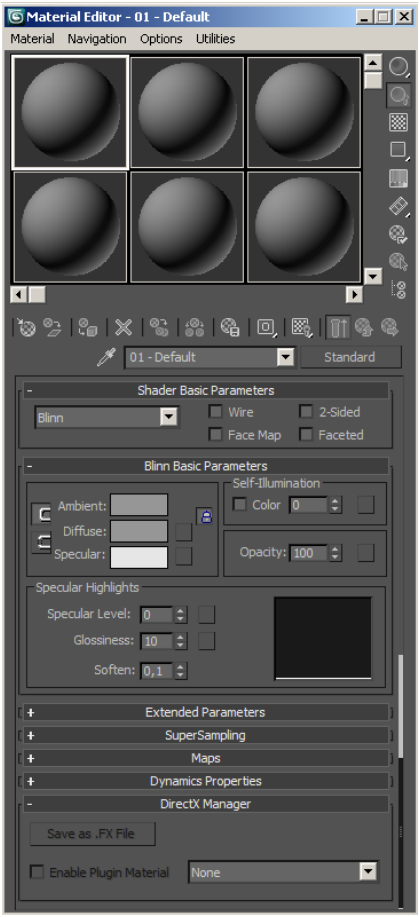


Fig. 4.22: Opción Material Editor

La Fig. 4.23 muestra la modificación del color del modelo.

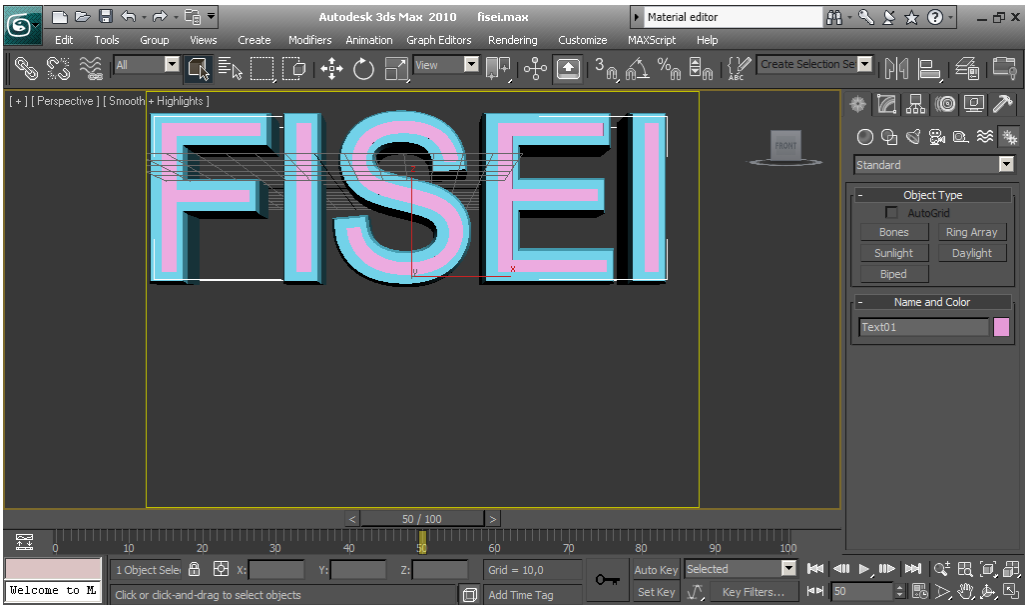


Fig. 4.23: Modificación del color del modelo

Posterior a la creación del arte del modelo, se procede a crear una animación, para esto se cuenta con una barra ubicada en la parte inferior de la ventana; donde se ubica la opción Auto Key, esta opción permite "grabar" una animación con la ayuda de la barra de tiempo, misma que se tornará de color rojo al momento que se esté captando la animación. El proceso consiste en deslizar la barra de herramientas hasta una ubicación deseada y modificar el modelo (girar o modificar su tamaño). La Fig. 4.24 muestra la barra de animación Autodesk 3Ds Max. **[26]**

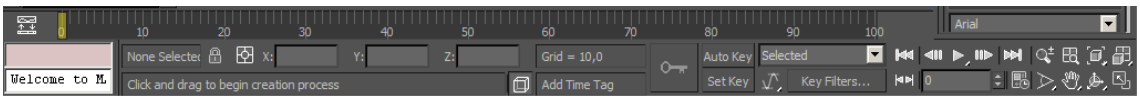


Fig. 4.24: Barra de animación Autodesk 3Ds Max

NOTA: Basta con presionar el botón play de la barra de animación para observar como el modelo parece que gira en su propio eje.

Creación de la animación en 3D con extensión *.DAE: Es necesario descargar un plug-in del software Autodesk 3Ds Max. Para realizar esta operación, el plug-in se conoce con el nombre de OpenCOLLADA y su instalación está disponible bajo varias plataformas.

La Fig. 4.25 muestra la animación del modelo en el tiempo 25/100. La exportación se la hace desde archivo, exportar donde aparece la ventana mostrada en la Fig. 4.26. **[26]**

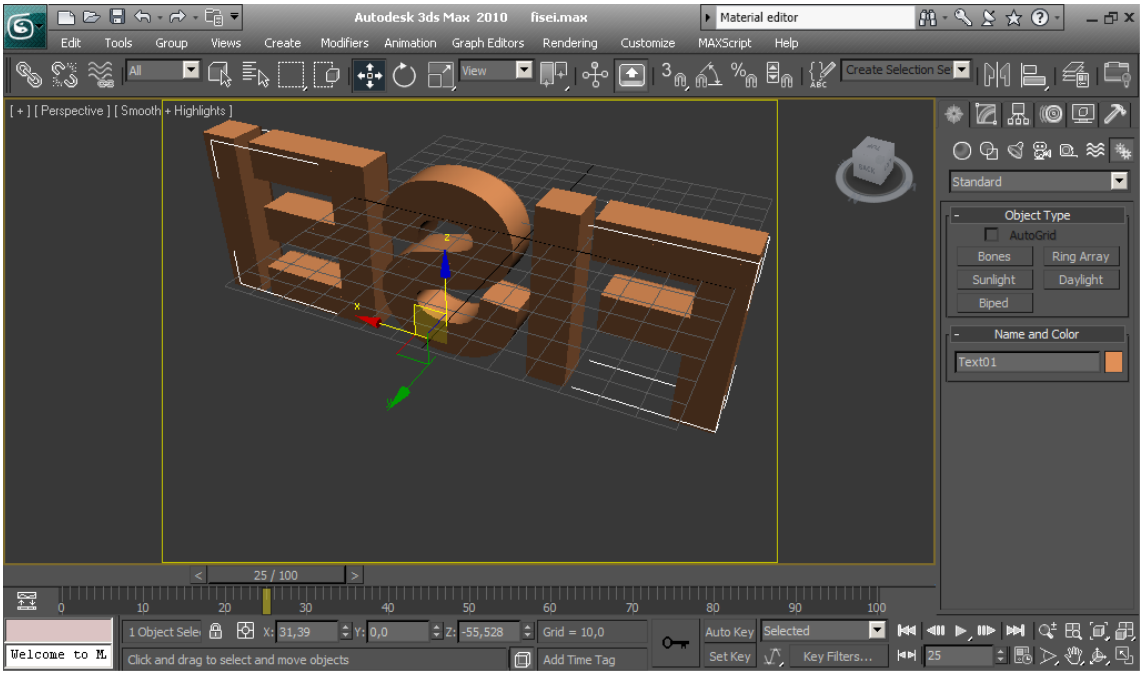


Fig. 4.25: Animación del modelo en el tiempo 25/100

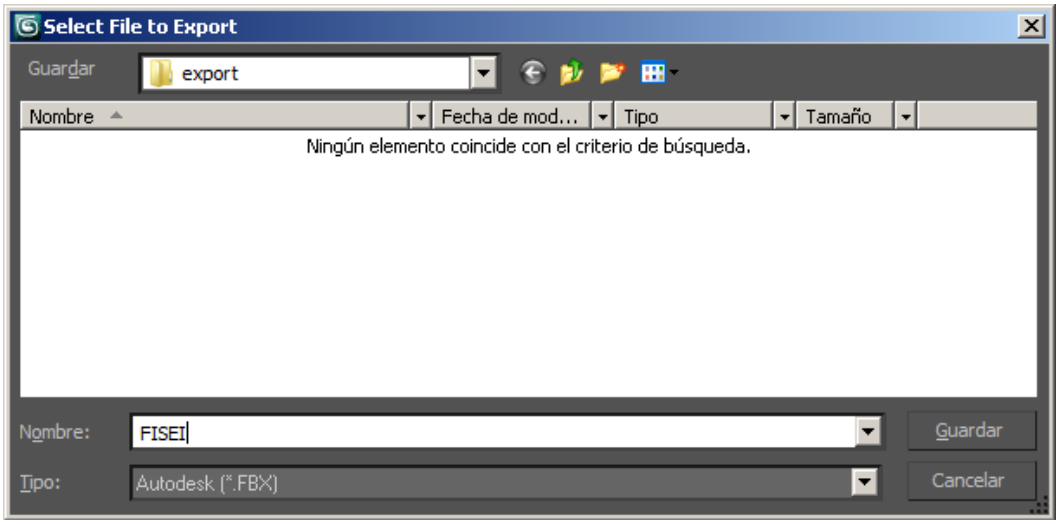


Fig. 4.26: Exportar un archivo en formato *.DAE de Autodesk 3Ds Max

En tipo se selecciona la opción OpenSceneGraph Exporter (*.DAE) y se guarda el archivo, el mismo que será utilizado por la aplicación de RA posteriormente. **[26]**

La Fig. 4.27 muestra la animación del modelo en el tiempo 50/100.

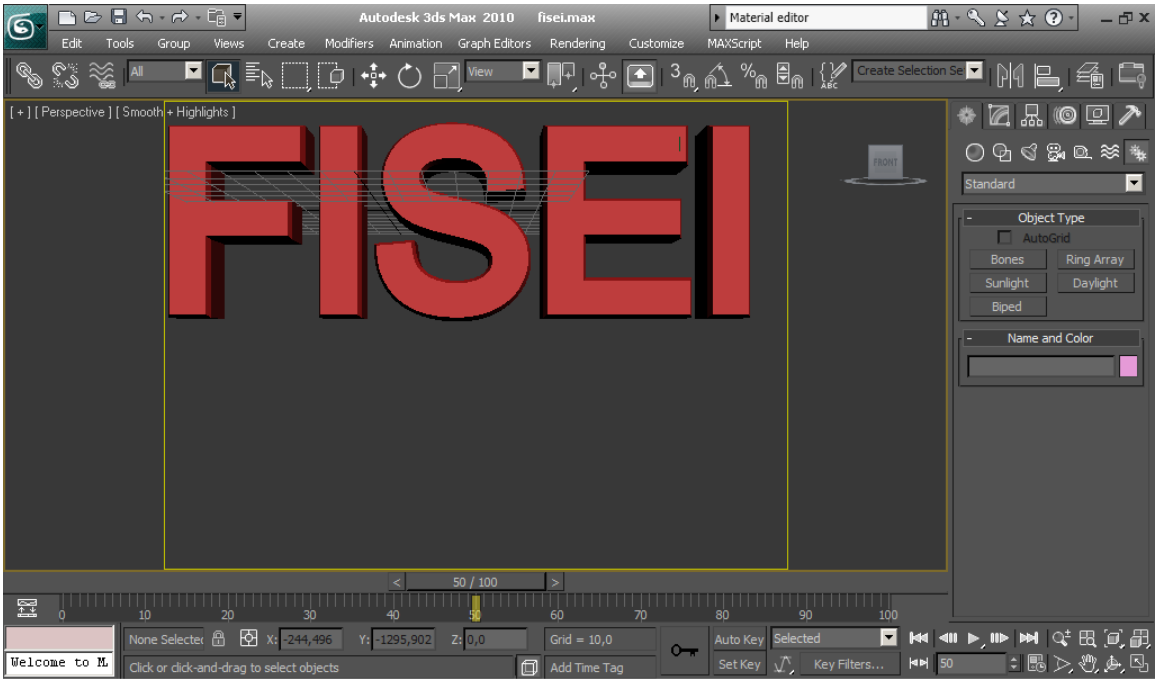


Fig. 4.27: Animación del modelo en el tiempo 50/100

La Fig. 4.28 muestra la animación del modelo en el tiempo 75/100.

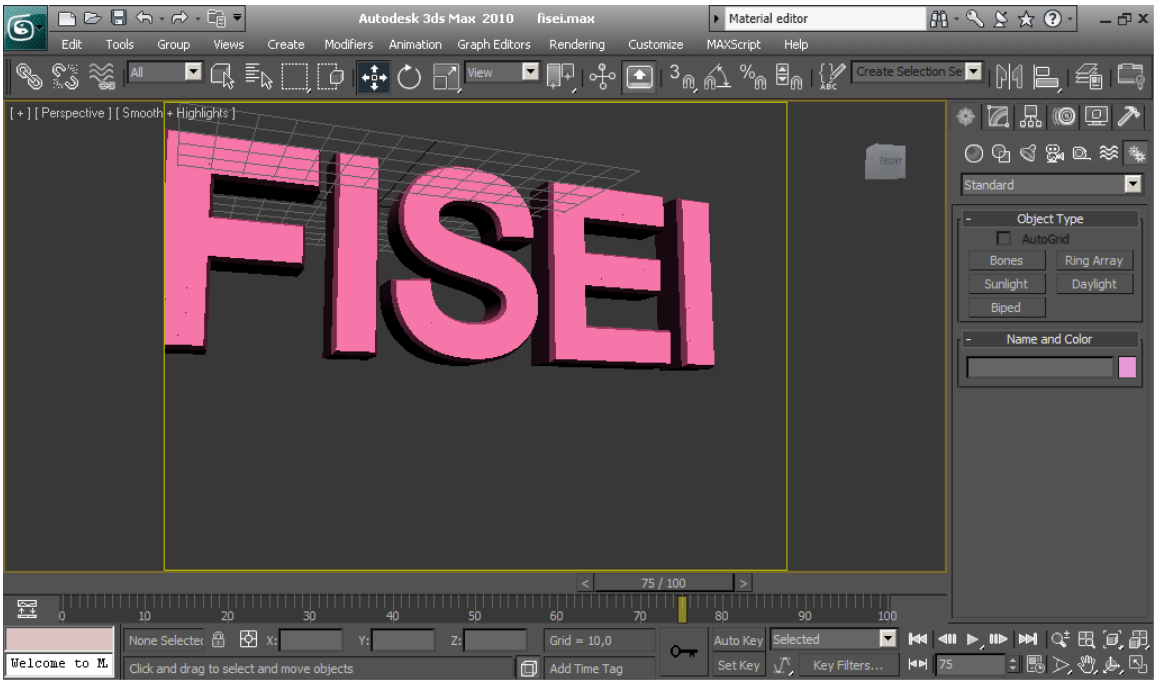


Fig. 4.28: Animación del modelo en el tiempo 75/100

La Fig. 4.29 muestra la animación del modelo en el tiempo 100/100.

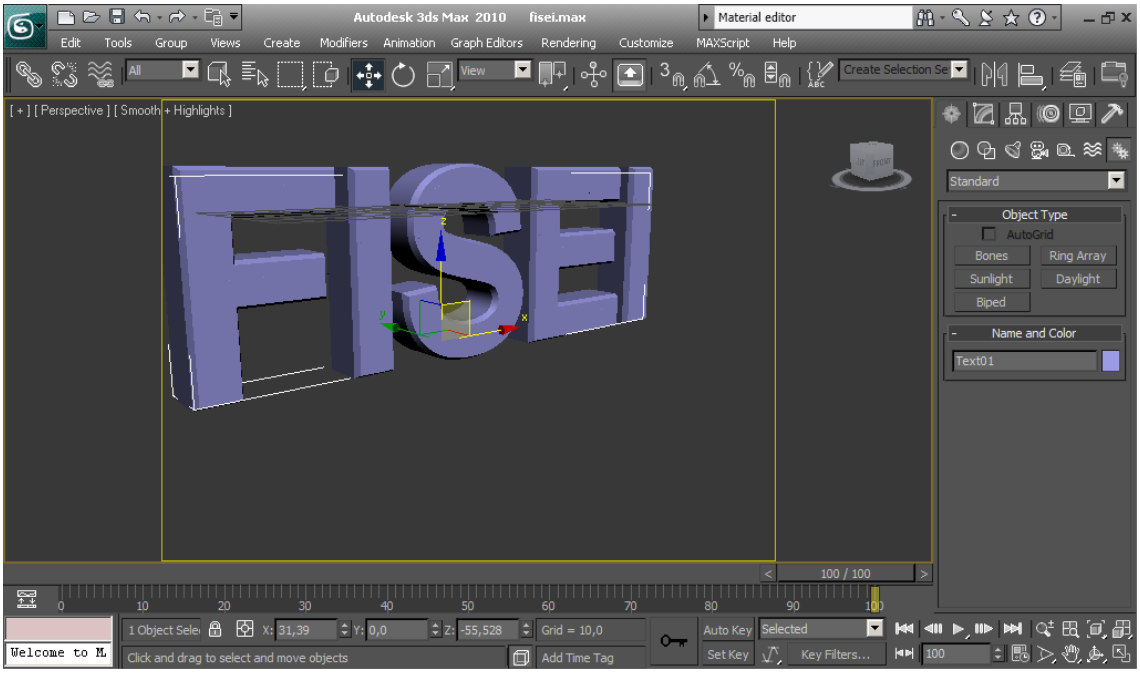


Fig. 4.29: Animación del modelo en el tiempo 100/100

4.6.4 Codificación

La Pizarra Virtual usando Realidad Aumentada utiliza la codificación de los siguientes lenguajes de programación:

Descripción del lenguaje de desarrollo Windows Forms: El lenguaje de programación orientada a objetos, utilizado para el diseño e implementación de la Interfaz Gráfica de Usuario (GUI) se detalla a continuación:

Microsoft Visual Studio: Microsoft Visual Studio 97, fue la primera versión que salió al mercado, ésta incluía Visual Basic 5.0 y Visual C++ 5.0 para realizar software para Windows específicamente, mientras que Visual J++ 1.1 era para Java y Windows. Los otros lenguajes eran Visual Fox Pro 5.0 para la Bases de Datos y Visual InterDev 1.0 para crear sitios dinámicos con ASP. Esto, como se puede asociar por la versión del producto, se realizó en 1997. Microsoft Visual Studio 6.0 salió al siguiente año, en 1998. Con eso se movieron los números de versión de todos los lenguajes de programación y fue la última versión que Visual J++ y Visual InterDev aparecieron en una paquetería de Visual Studio. **[27]**

Microsoft Visual Studio .NET 2002 fue un salto completamente drástico, ya que se cambia la estructura totalmente. Ahora con .NET los programas no se compilan para generar un ejecutable máquina (un archivo .EXE conocido comúnmente para el ambiente Microsoft Windows), sino que se genera un archivo intermedio para poder ser ejecutado en diferentes Plataformas (al decir plataformas, las distintas arquitecturas de software y hardware; como GNU/Linux, Solaris de Sun Microsystems o Mac OS X de Apple Inc.). La Fig. 4.30 muestra el logotipo de Microsoft Visual Studio. **[27]**

Fig. 4.30: Logotipo de Microsoft Visual Studio
Fuente. www.microsfot.com/08/M_logo.svg_.png

A esta versión se le agregan nuevos lenguajes que son: Visual J# (sucesor del desaparecido Visual J++), Visual C#, .NET. Microsoft Visual Studio 2010, es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows, soporta varios lenguajes de programación como C++, Visual C#, Visual J#, ASP.NET. Visual Studio permite a los desarrolladores crear aplicaciones, así como servicios Web en cualquier entorno que soporte la plataforma .NET; así como dispositivos móviles. **[27]**

Microsoft Visual Studio 2010 Professional: Visual Studio 2010 es una herramienta para programación que soporta lenguaje C, C++, C# (C Sharp) y Visual Basic; lo que permite un procesamiento rápido del código, ideal para el desarrollo del software de la Pizarra Virtual usando Realidad Aumentada; funciona muy bien para cualquier versión de OpenCV. La Fig. 4.31 muestra el logotipo de Microsoft Visual Studio Professional 2010. **[27]**

Fig. 4.31: Logotipo de Microsoft Visual Studio 2010 Professional
Fuente. www.microsfot.com/13/M_logoVS2010.svg_.png - [27].

Este IDE crea como proyecto, una solución, la misma puede ser una aplicación tipo: aplicación de consola (líneas de comandos Win32), aplicación Windows Forms (Interfaz Gráfica de Usuario), entre otras. Sin embargo el inconveniente que se presenta con esta herramienta, es que por sí sola no tiene entorno gráfico para C++, que es el lenguaje en que se trabaja con OpenCV; es decir, en un principio se está limitado a usar OpenCV solo con aplicaciones de consola. **[27]**

Pero es posible hacerlo con la ayuda de librerías (Qt) o trabajar en lenguaje compatible con interfaz gráfica de Windows como C# o Basic en una aplicación Windows Forms, llamando a las funciones y clases de OpenCV con la ayuda de la herramienta EmguCV.

Se puede usar programación estructurada y dividir el proyecto en partes o trabajar con programación orientada a objetos usando las clases para procesamiento de video que usa OpenCV y su envoltorio EmguCV. Se elige trabajar con Microsoft Visual Studio 2010 Professional ya que es compatible para versiones de Windows 7, Windows 8, Windows 8.1 y posteriores que son las versiones que predominan en uso a la fecha de desarrollo del presente proyecto.

Descripción de las librerías de Visión Artificial: Para detección de objetos y tracking se cuenta con las librerías de OpenCV que son de código abierto, se disponen varias versiones que difieren un poco unas de otras, un hecho importante es que las versiones más actuales poseen librerías ya compiladas por lo que no es necesario usar un compilador auxiliar como C-Make; sin embargo existe amplia documentación sobre cómo llevar a cabo la creación de nuestras propias librerías. Se analiza la versión más actual a la fecha de inicio del presente proyecto, se esta OpenCV 2.4.9. **[27]**

Las versiones más actualizadas de OpenCV usan tipos de variables y funciones propias, algunas difieren de las versiones anteriores, por ello se hará un breve resumen tanto de las variables como funciones principales que se requieren para poder trabajar con las librerías de OpenCV dentro del entorno de Microsoft Visual Studio 2010. **[27]**

Por otra parte la aplicación requiere un entorno visual, para lograr este acometido se usó EmguCV que permite el uso de las funciones y clases de OpenCV para trabajar en el ambiente gráfico de Visual C#.NET o Basic.Net como una aplicación Windows Forms.

Será necesario hacer un resumen del modo de empleo y bondades de estas librerías sobre Microsoft Visual Studio ya que la aplicación será ejecutada sobre este IDE; por otra parte se hace un enfoque en el lenguaje de programación a usar, en virtud de cumplir a cabalidad los objetivos de la aplicación en cuanto a tracking, procesamiento de imágenes en movimiento, detección y reconocimiento de marcadores, entrenamiento Haar para detección y reconocimiento de objetos, visualización de imágenes en 3D; requerimientos ineludibles para este proyecto.

OpenCV: El 13 de Junio de 2000, Inetl Corporation anunció que estaba trabajando en la realización de una nueva biblioteca de estructuras y funciones en lenguaje C, junto con un grupo de investigadores (Jitendra Malik, University of California Berkeley; Takeo Kanade, Carnegie, Mello University; Pietro Perona, NSF Engineering Research Center, etc.) en visión por computador. **[27]**

De esta forma se desarrolló The Open Source Computer Vision Library (OpenCV), especialmente diseñado para el tratamiento, captura y visualización de imágenes en áreas como interacción hombre-máquina, robótica, monitorización, biométrica, segmentación y reconocimiento de objetos, seguridad; y que proporcionan bibliotecas de tipos de datos estáticos y dinámicos (matrices, grafos, árboles, etc.). OpenCV está optimizado para ser usado bajo procesadores Intel, Pentium Pro, Pentium III y Pentium 4, etc., pero puede ser usado bajo cualquier otro tipo de procesadores. **[27]**

OpenCV es ampliamente utilizado en entornos de vigilancia y seguimiento de objetos. La biblioteca OpenCV es una API de más de 500 funciones escritas en lenguaje C y que tienen las siguientes características:

- Su uso es de libre tanto para uso comercial como no comercial. **[28]**
- No utiliza bibliotecas numéricas externas, aunque puede nacer uso de alguna de ellas en tiempo de ejecución si están disponibles. **[28]**
- Es compatible con IPL (Intel Procesing Library) y utiliza IPP (Intel Integrated Performance Primitives) para mejorar su rendimiento, si están disponibles en el sistema. **[28]**

Además, OpenCV dispone de interfaces para otros lenguajes y entornos como:

- EiC-Intérprete ANSI C escrito por Ed Breen, actualmente se encuentra en desuso. Hawk y CvEnv son entornos interactivos (escritos en MFC y TLC, respectivamente).
- Ch-Intérprete ANSI C/C++ con algunas características de scripting, desarrollado y mantenido por la compañía SoftIntegration (http://www.softintegration.com). Los wrappers para Ch están disponibles en opencv/interfaces/ch. **[28]**
- MATLAB-Entorno para procesamiento numérico y simbólico desarrollado por Mathworks. El interfaz de MATLAB para algunas funciones de OpenCV se encuentra disponible en opencv/interfaces/matlab/toolbox. En cuanto a la integración con Matlab, OpenCV puede utilizar estructuras nativas de Matlab, y es compatible con Image Processing Toolbox. **[28]**

OpenCV implementa una gran variedad de herramientas para la interpretación de la imagen. OpenCV es principalmente una librería de algoritmos para las técnicas de calibración (calibración de la cámara), detección de rasgos, para rastrear (flujo óptico), análisis de la forma (geometría, contorno que procesa), análisis del movimiento (plantillas del movimiento, estimadores), reconstrucción 3D (transformación de vistas), segmentación de objetos y reconocimiento (histogramas, bordes Canny, etc.). Además de la reconstrucción 3D, manejo de imágenes 3D y el entrenamiento Haarcascade categorías principales en el software de la Pizarra Virtual usando Realidad Aumentada. La Fig. 4.32 muestra el logotipo de la librería OpenCV. **[28]**

Fig. 4.32: Logotipo de la librería OpenCV
Fuente. http://opencv.org/wp-content/themes/opencv/images/logo.png - [28].

La versión más actual de OpenCV a la fecha de implementación del proyecto es 2.4.9 y por ser una librería en código abierto, está disponible para descarga gratuita en su página oficial; la misma que describe que OpenCV está liberada bajo la licencia BSD (Berkeley Software Distribution) y por lo tanto es libre tanto para uso comercial como para académico. **[28]**

Esta librería tiene interface para C++, C, Python y Java además soporta Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para eficiencia computacional y con un profundo enfoque en las aplicaciones en tiempo real. Escrito en lenguaje C, C++ optimizado, la librería puede tomar ventaja del procesamiento multi-core. **[28]**

Estructura: Todas las herramientas de alto nivel de OpenCV hacen uso de un paquete de clases C++ y funciones C que usan a su vez funciones muy eficientes escritas en C. Concretamente, el conjunto de funciones suministradas por la librería OpenCV pueden agruparse en los siguientes bloques:

- Estructuras y operaciones básicas. **[28]**
- Procesamiento y análisis de imágenes: filtros, histogramas, momentos, etc. **[28]**
- Análisis estructural: geometría, procesamiento del contorno, etc. **[28]**
- Análisis del movimiento y seguimiento de objetos, etc. **[28]**
- Reconocimiento de objetos: objetos propios (eigen objects), modelos HMM, etc.
- Calibración de la cámara: morphing, geometría epipolar, estimación de la pose, etc.
- Reconstrucción tridimensional (funcionalidad experimental): detección de objetos, seguimiento de objetos tridimensionales, etc. **[28]**

OpenCV se puede dividir en varias partes bien diferenciadas para su uso, que están a su vez divididas en cabeceras (*.h) y son los siguientes: Cv, CvAux, HighGUI y CvCam. La Fig. 4.33 muestra la estructura de OpenCV. **[28]**

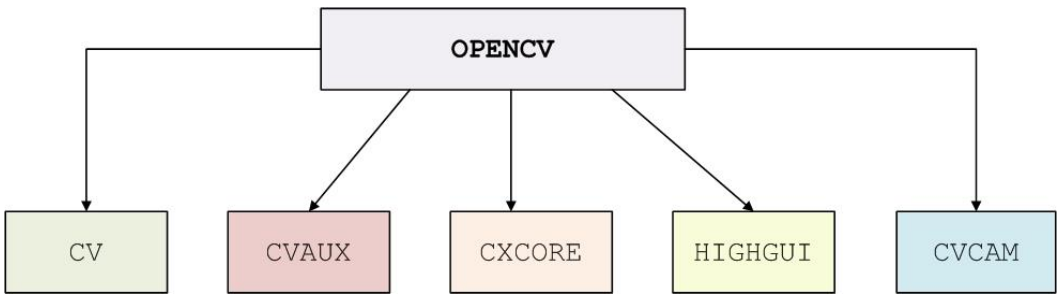


Fig. 4.33: Estructura OpenCV
Fuente. The OpenCV Reference Manual – [28].

Cada parte está especializada en una tarea en concreto:

- En las bibliotecas **Cv** se puede encontrar los operadores estándar para el cálculo de gradientes, detección de bordes, esquinas y contornos, conversión de modelos de

color, operaciones morfológicas como dilatación, erosión y adelgazamiento, transformadas geométricas (*Hough*), operaciones con momentos (momento central, momentos invariantes, etc.), operaciones con histogramas y de matching. **[28]**

Sobre análisis estructural, se puede encontrar funciones para procesamiento de contornos (momentos, representación jerárquica de contornos, etc.) y computación geométrica. También se encuentra funciones para tratar aspectos de análisis de movimiento y seguimiento de objetos, procesamiento de flujo óptico, reconocimiento de patrones y estimación de la posición de objetos, así como funciones para reconstrucción en 3D (calibración de la webcam, reconocimiento de gestos, etc.). **[28]**

- En **CvAux** están las funciones en fase de prueba o experimentales, como pueden ser algoritmos para visión estéreo, seguimiento 3D, etc. **[28]**
- **HighGUI** permite la escritura y lectura de imágenes en numerosos formatos (BMP, JPEG, TIFF, PxM, Sun Raster, etc.) y la captura de stream de video de capturadoras Martos y cámaras-capturadoras con controlador VFW-WDM (la mayoría del mercado). **[28]**

También permite la creación de ventanas para visualizar imágenes en ellas. Estas ventanas recuerdan su contenido (no es necesario implementar callbacks de repintado) y además, nos proporciona mecanismos muy fáciles para interaccionar con ellas: trackbars que capturan la entrada del teclado y el ratón. **[28]**

- **CvCam** proporciona un único interfaz de captura y reproducción bajo Linux y Win32; callbacks para la gestión de stream de video o ficheros AVI y un mecanismo fácil para implementar visión estéreo con dos cámaras USB o estéreo-cámara. **[28]**

A partir de la versión 0.9.6. Estas funciones específicas de cámara se eliminaron para el sistema GNU/Linux, dejando solamente operativas para el trabajo con cámaras el módulo HighGUI. **[28]**

- **CxCore** dispone de estructuras de datos dinámicas y operadores para estas estructuras. Entre estas estructuras de datos se puede encontrar vectores (cvArr), matrices (cvMat), imágenes, grafos, etc. **[28]**

- La mayoría recibe por parámetro un vector máscara para especificar sobre qué elementos de la estructura se aplica el operador. Se puede encontrar operadores aritméticos (Add, Mult, Sub, AbsDiff, etc.), lógicos (AND, OR, NOT, XOR), de comparación, etc. Así como funciones para aplicar cálculos estadísticos (SUM, MAX, MIN, AVG, etc.) y operadores de conversión de tipo o escala de colores (ConvertScale). **[28]**

Instalación de OpenCV: Es posible descargar de forma gratuita esta herramienta en su página web oficial, luego de elegir la pestaña de downloads se elige la versión y el sistema operativo en donde será usada, para el caso particular de este proyecto se trabajará en Windows al ser de mayor utilidad en la Unidad Educativa "Tirso de Molina", una vez descargada esta librería se procede a la instalación.

Para usar este paquete se tienen dos opciones; instalarlo usando las librerías pre-compiladas o realizando nuestras propias librerías, siendo la primera mucho más sencilla y rápida sin embargo, al recurrir a esta vía rápida, las librerías compiladas solo funcionarán con una versión actual de Microsoft Visual Studio y además no se tendrá ventaja sobre los avances que se han implementado en las últimas versiones como el uso de multi-core que permite ejecutar procesos en paralelo con el fin de optimizar una aplicación en tiempo real. **[28]**

Instalación usando librerías Pre-compiladas: Una vez descargada la librería se la debe descomprimir en un directorio conocido, pues será necesario conocer la dirección donde se instala la carpeta de OpenCV para futuras acciones, se recomienda instalar en una carpeta titulada OpenCV en un disco local del ordenador la que contiene el sistema operativo, por ejemplo C:\OpenCV; allí se guardaran todas las subcarpetas que pertenecen a las librerías, incluidas las pre-compiladas. **[28]**

Como ya se tiene las librerías resta establecer la variable de entorno que permite trabajar de forma más sencilla para ello se acude a: panel de control - sistema y mantenimiento – sistema; luego se accede a "configuración avanzada del sistema" y en el cuadro de diálogo que aparece se selecciona variable de entorno y en variables de sistema se crea una nueva variable donde su nombre sería "OPENCV_BUILD" y el valor "C:\opencv249\build", por otra parte en la variable del sistema "Path" se añade la ruta

de la librería ya compilada ";%OPENCV_BUILD%\OPENCV249\build\x86\vc10\lib;"; con ello se tiene instalado OpenCV para ser usado en IDE requerido. La Fig. 4.34 muestra la edición de las variables del sistema para OpenCV. **[28]**

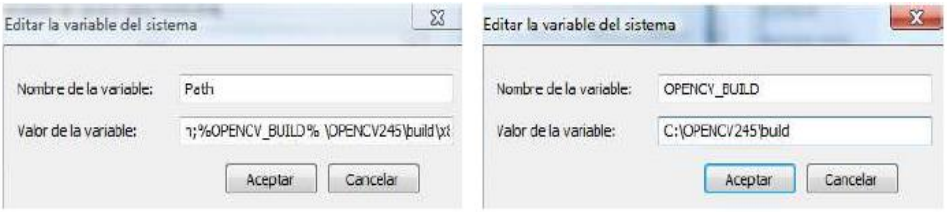


Fig. 4.34: Edición de las variables del sistema para OpenCV

Uso de OpenCV con Microsoft Visual Studio: Para poder realizar una aplicación usando OpenCV se debe generar una hoja de propiedades para el proyecto, en la misma se establece las direcciones de los archivos que serán incluidos a la solución cuando se trabaje con OpenCV, sea con las librerías pre-compiladas o con las librerías personalizadas, para ello se debe tener muy en cuenta en donde se han almacenado los archivos de OpenCV. **[28]**

No se debe olvidar establecer la variable de entorno antes de nada. En VS2010 OpenCV trabaja para aplicaciones de Consola, por ellos se debe crear un nuevo proyecto de este tipo para Visual C++. En el administrador de propiedades, se elige Debug y se añade una nueva hoja de propiedades que servirá para todos los proyectos que requieran el uso de librerías de OpenCV en una aplicación de este tipo. La Fig. 4.35 muestra la ventana de propiedades de OpenCV. **[28]**

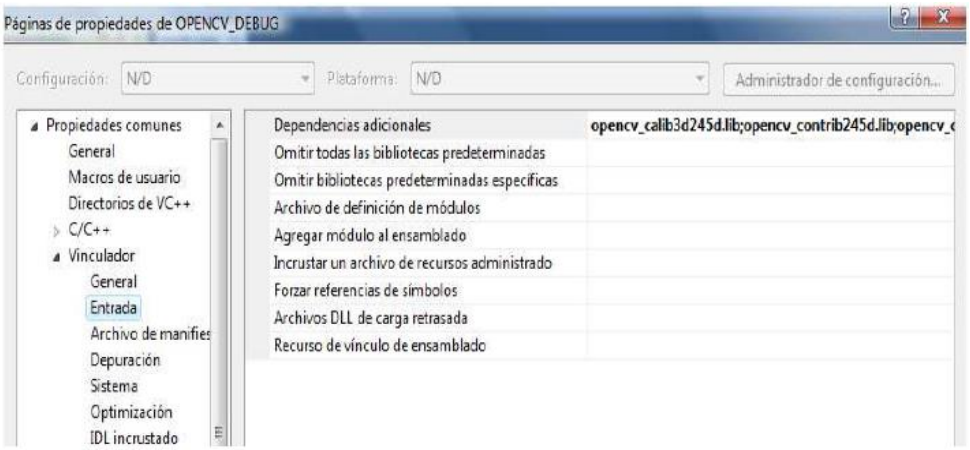


Fig. 4.35: Ventana de propiedades de OpenCV

Una vez creada la nueva hoja de propiedades del proyecto se la accede a la ventana de configuración al dar clic derecho sobre la misma, y se elige la pestaña propiedades. Aparece una ventana similar a la de la Fig. 2.35 donde se debe configurar de la siguiente manera:

- Se accede a la pestaña: Propiedades comunes, C/C++, General, Directorios de inclusión adicionales; se procede a editar y se añade la dirección "$(OPENCV_BUILD)\include", donde OPENCV_BUILD es la variable de entorno establecida anteriormente en las variables de entorno del sistema. **[28]**
- Luego en la pestaña: Vinculador, General, directorio de bibliotecas adicionales se edita y se añade la dirección "$(OPENCV_BUILD)\x86\vc10\lib". **[28]**

Por último en la pestaña: Vinculador, Entrada, Dependencias adicionales; se edita y asigna las dependencias:

- opencv_calib3d249d.lib
- opencv_contrib249d.lib
- opencv_core249d.lib
- opencv_features2d249d.lib
- opencv_flann249d.lib
- opencv_gpu249d.lib
- opencv_haartaining_engined.lib
- opencv_highgui249d.lib
- opencv_imgproc249d.lib
- opencv_legacy249d.lib
- opencv_ml249d.lib
- opencv_nonfree249d.lib
- opencv_objdetect249d.lib
- opencv_photo249d.lib
- opencv_stitching249d.lib
- opencv_superres249d.lib
- opencv_ts249d.lib
- opencv_video249d.lib
- opencv_videostab249d.lib

NOTA: Estos son los archivos contenidos en el directorio OPENCV_BUILD\build\x86\vc10\lib; aunque no todos son necesarios para la aplicación en desarrollo. **[28]**

La configuración de OpenCV para VS2010 está lista y se puede probar ejecutando el código de una aplicación sencilla como la siguiente, la cual accede al video de una cámara web y lo muestra en una ventana. El código y comentarios se muestran a

continuación en donde las primeras líneas son siempre requeridas para acceder a las librerías de OpenCV. **[28]**

```
#include “opencv2/opencv.hpp”
using namespace cv; //Directiva para el uso de clases y funciones de OpenCV
int main(){
Mat imagen; //Se crea un objeto donde s almacena cada imagen de la trama
VideoCapture video; //Se crea un objeto para captura de video
Video.open(0); //Se accede al video de la cámara
namedWindow(“WEBCAM”, 1); //Se crea una ventana para mostrar la imagen
while(1)
{
Video>>imagen; //Se asigna una trama la webcam a una imagen
Imshow((“WEBCAM”, imagen); //Se muestra la imagen en la pantalla
waitKey(16); //retardo para tener aproximadamente 60 fps
}
return 0;
}
```

Para probar el funcionamiento de OpenCV en VS2010 ya se han usado algunas funciones y clases propias de OpenCV, todas ellas localizadas en el namespace cv; para una mejor comprensión de los argumentos que se involucran, se hace un breve resumen de las mismas. **[28]**

Principales estructuras de OpenCV:

OpenCV usa tipos de datos propios de las librerías, así que se exponen los más importantes para la aplicación y se toma sus definiciones. Los expuestos son estructuras para trabajo en C y se puede aplicar en programación con VS2010. **[28]**

cvMat: Pueden almacenar imágenes para procesar, es el análogo a IplImage que se usa en versiones anteriores de OpenCV, para evitar inconvenientes de compatibilidad se usa una estructura Mat, la cual es mucho más extensa abarcando varios recursos de OpenCV. **[28]**

cvPoint: Tipo de dato para almacenar coordenadas en dos dimensiones. Para declarar un dato de este tipo en C++ su estructura es: Point punto (0,0);

cvRect: Almacena las coordenadas de un rectángulo, importante porque en método detectMultiScale devuelve sus datos en este tipo de estructuras. **[28]**

Principales funciones de OpenCV:

A continuación se analiza la utilidad de las funciones más importantes de OpenCV, según la siguiente descripción:

cvtColor: Convierte una imagen de un espacio de color a otro, como por ejemplo de RGB a HSV o escala de grises. Su estructura para C++ es la siguiente:

void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)

- src: objeto donde esta almacenada la imagen a convertir. **[28]**

- dst: objeto donde esta almacenada la imagen convertida. **[28]**

- code: tipo de conversión.

equalizeHist: Ecualiza el histograma de una imagen en escala de grises, permitiendo normalizar el brillo y aumentar el contraste de la imagen. Su estructura para C++ es la siguiente:

void equalizeHist(InputArray src, OutputArray dst)

- src: objeto donde esta almacenada la imagen a ecualizar. **[28]**

- dst: objeto donde esta almacenada la imagen ecualizada. **[28]**

- code: tipo de conversión.

namedWindow: Crea una ventana que puede ser usada como un muestrario de imágenes y barras. Su estructura para C++ es la siguiente:

void namedWindow(const string& winname, int flags=WINDOW_AUTOSIZE)

- winname: nombre de la ventana que se crea. **[28]**

imshow: Despliega una imagen en una ventana específica. Su estructura para C++ es la siguiente:

void imshow(const string& winname, InputArray mat)

- winname: nombre de la ventana que se despliega. **[28]**

- InputArray: imagen a mostrar en la ventana. **[28]**

waitKey: Espera que se presione una tecla o un retardo en milisegundos, solo funciona cuando una ventana HighGUI está activa. Su estructura para C++ es la siguiente:

int waitKey (int delay=0)

- delay: retardo en milisegundos, sin argumento; significa por siempre. **[28]**

Principales clases de OpenCV:

En la programación orientada a objetos, se ejecutan métodos asociados a un objeto que pertenece a una clase. A continuación se pone de manifiesto algunas clases y algunos métodos más importantes asociados a OpenCV. **[28]**

CascadeClassifier: Es una clase usada para la detección de objetos a la cual se asocian varios métodos, para los más importantes a continuación se describe su aplicación y parámetros. **[28]**

- CascadeClassifier: Su estructura para C++ es:

CascadeClassifier::CascadeClassifier (const string& filename)

Al ejecutar este método a un objeto de la clase CascadeClassifier, se carga en el mismo un classifier cuyo nombre se especifica en su argumento "filename". **[28]**

- load: Su estructura para C++ es:

bool CascadeClassifier::load(const string& filename)

Parecido al método anterior, pero se puede cargar un objeto "HAAR" entrenado con la aplicación "haartraining" o un nueva "cascade classifier" entrenado en la aplicación "traincascade". OpenCV viene con varios classifier ya entrenados que se encuentran en la carpeta "data" sean estos; haarcascades, hogcascades, o lbpcascascades que eligen según el objeto a detectar. **[28]**

- detectMultiScale: Su estructura para C++ es:

void CascadeClassifier::detectMultiScale(const Mat& image, vector<Rect>& objects, doublé scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size mazSize=Size())

Detecta objetos de distintos tamaños en la imagen deseada y devuelve una lista de rectángulos asociados a los objetos, esta función se puede paralelizar con TBB para detectar varios objetos en simultáneo. Sus argumentos se detallan a continuación. **[28]**

cascade.- el cascade usado para la detección. **[28]**

image.- matriz de tipo CV_8U (escala de grises) contiene la imagen en donde se va a detectar el objeto. **[28]**

objects.- vector de datos tipo Rectangle donde cada uno está asociado al objeto detectado. **[28]**

scaleFactor.- especifica que tanto es reducida la imagen en la búsqueda del objeto. **[28]**

minNeighbor.- se especifica cuantos elementos adyancentes debe tener un rectángulo para retener el objeto. **[28]**

flags.- no se usa con CascadeClassifier solo con los Haar. **[28]**

minSize.- mínimo tamaño del objeto. **[28]**

maxSize.- máximo tamaño del objeto. **[28]**

VideoCapture: Es una clase para la captura de video, sea desde archivos o desde cámaras. Esta clase provee una API (Application Programming Interface). Los métodos más importantes son:

- VideoCapture: Su estructura para C++ es:

VideoCapture::VideoCapture(int device)

Se usa como constructor para acceder a un archivo o dispositivo. **[28]**

- release: Su estructura para C++ es:

void VideoCapture::release()

Cierra el archivo o dispositivo de captura. **[28]**

Clasificador de Haar:

Un clasificador de Haar sirve para detectar casi cualquier objeto presente en una imagen. Las librerías de OpenCV comentadas en el epígrafe anterior, implementan una versión del detector de rostros desarrollado por Paul Viola y Michael Jones (conocido como Viola & Jones detector) y posteriormente ampliado por Rainer Lienhart y Jochen Maydt para que use características diagonales. **[29]**

En su trabajo, Viola & Jones demostraron como, a partir de características locales de la imagen basadas en el cambio de intensidad, se podía desarrollar un detector de rostros muy robusto. Partieron de la idea de determinar características usando sumas y restas de los niveles de intensidad de la imagen, para conseguir este objetivo usaron filtros de Haar (Haar-like wavelets). Funcionan aplicando filtros de diversos tamaños que recorre la imagen, si dicho valor está por encima de un cierto umbral, esa zona de la imagen se clasificará como cara. La Fig. 4.36 muestra los Filtros de Haar. **[29]**

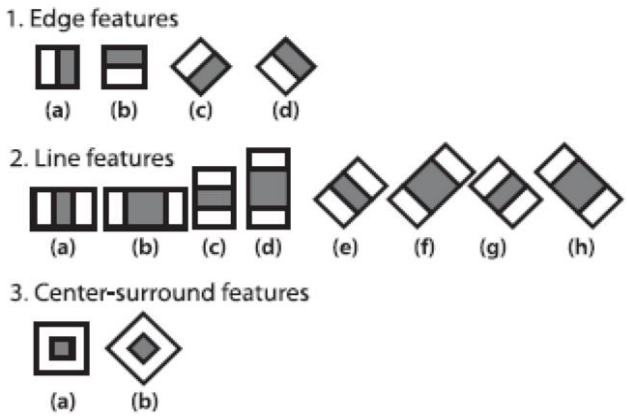


Fig. 4.36: Filtros de Haar
Fuente. Tesis de Ingeniería Industrial – [29].

Estos primeros clasificadores, por sí solos, consiguen resultados muy pobres pero, combinando varios de ellos se puede generar un clasificador mucho más robusto que incrementa exponencialmente el éxito de detección, esta técnica se conoce con el

nombre de boosting. De esta manera se consigue clasificadores robustos muy precisos (del orden del 99.9% de acierto), el problema es que, además, presentan una elevada tasa de falsas detecciones (del orden del 50%). **[29]**

Por este motivo, Viola & Jones propusieron un esquema basado en una cascada de clasificadores robustos. En las primeras etapas de esta cascada de clasificadores se detectan zonas de la imagen que son muy diferentes de una cara, mientras que en las últimas etapas se rechazan ejemplos mucho más complejos. **[29]**

Para cascadas de 20 etapas se consigue una tasa de acierto de 0.999^{20} (a través de toda la cascada), es decir; del 98% aproximadamente, mientras que la tasa de falsos positivos decae hasta el valor de 0.5^{20}, es decir; aproximadamente 0.0001%. **[29]**

OpenCV denomina Clasificador de Haar al detector de Viola & Jones, dado que se usa los Filtros de Haar. Una cascada de Clasificadores de Haar se denomina Cascada de Haar. **[29]**

El objetivo de Viola & Jones era la detección de rostros, pero las Cascadas de Haar no sólo detectan caras, también funcionan bastante bien con objetos "rígidos", es decir; que tienen vistas muy distintas. A la creación de una Cascada de Haar, se le denomina también Entrenamiento de una Cascada de Haar. **[29]**

Boosting: Es un método que consiste en combinar un conjunto de clasificadores sencillos o débiles que, por sí solos no conseguirían un porcentaje de detección aceptable, pero que combinados eficientemente producen un sistema de decisión muy robusto con tasas de detección elevadas y tasas de falsas alarmas de detección reducidas. **[29]**

Los clasificadores débiles se denominan así por su sencillez y su escasa precisión. Se corresponden con reglas de clasificación simples que entregan como resultado un valor de confianza respecto a la predicción que están haciendo. Estos clasificadores evalúan características como los niveles de intensidad de una cierta zona de la imagen mediante sumas y restas para decidir finalmente si es subimagen la clasifican como objeto o como fondo. (No objeto). **[29]**

Los clasificadores débiles generan resultados pobres, pero se pueden definir clasificadores más robustos enlazando varios de ellos, consiguiendo un sistema con una elevada robustez que podrá ser aprendido por una máquina mediante un proceso denominado entrenamiento. De este modo se habrá automatizado un sistema de decisión que permitirá encontrar los objetos buscados en el total de la información que contiene la imagen. **[29]**

Cabe destacar que, aun cuando estos clasificadores robustos pueden llegar a una tasa de detección del 99%, la tasa de falsos positivos puede situarse por encima del 30%. **[29]**

Adaboost: El adaboost fue presentado en 1995 por Yoav Freund y Robert Schapire en la segunda Conferencia de Teoría de Aprendizaje Computacional y debe su nombre a la contratación de Adaptative Boosting, pues es un tipo de boosting que se ajusta adaptativamente a los resultados obtenidos por los clasificadores débiles de etapas previas, mejorando así su rendimiento global. **[29]**

En otras palabras, es una mejora del boosting basada en su mismo principio, la creación de un clasificador fuerte mediante la combinación lineal de varios clasificadores débiles. La diferencia estriba en que el adaboost genera y llama a una clasificador débil nuevo en cada etapa, actualizando el valor de la distribución de pesos de estos clasificadores. **[29]**

El adaboost funciona partiendo de un conjunto de imágenes de muestra a las que le aplica el método boosting de manera recursiva durante T etapas, quedándose en cada etapa con los mejores resultados y modificando el valor de los pesos de los clasificadores débiles que componen el boosting de tal forma que se le va incrementando el peso a aquellas muestras mal clasificadas y decrementando a aquellas bien clasificadas. Finalmente, al llegar a la etapa T, se genera un clasificador que combina los mejores clasificadores débiles de las etapas anteriores. **[29]**

Gentle Adaboost: El algoritmo adaboost antes comentado proporcionó una nueva línea de investigación, llegándose a desarrollar distintas variantes del original adaboost como el real adaboost de Schapire & Singer en 1999 (una generalización del adaboost), o el gentle adaboost de Lienhart, Kuranov & Pisarevsky en 2002. **[29]**

El gentle adaboost, como Lienhart reflejó en su artículo, es el algoritmo que mejores resultados empíricos ante problemas reales ha dado, y es por eso por lo que ha tenido gran éxito. Es el que se utiliza en el presente proyecto, emplea el algoritmo que se detalla en la siguiente página para seleccionar el mejor CART (Classification And Regression Tree) simple que cumple con la tasa e detección escogida inicialmente. **[29]**

Un CART es un árbol de clasificación y regresión, s es, un modelo de predicción lógico que sirve para representar y categorizar una serie de condiciones que ocurren de forma sucesiva para la resolución de un problema. Toma como entrada un objeto o una situación descrita por medio de un conjunto de atributos y, a partir de esto, devuelve una respuesta relacionada con la entrada. La Fig. 4.37 muestra un ejemplo de árbol de clasificación y regresión (CART). **[29]**

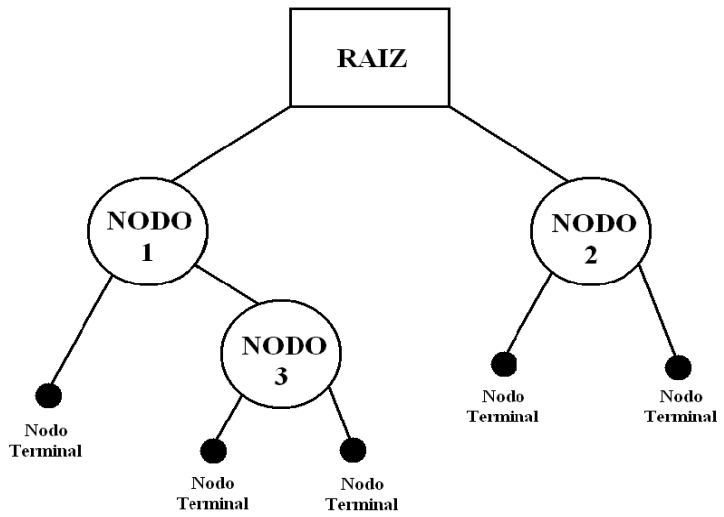


Fig. 4.37: Ejemplo árbol de clasificación y regresión (CART)
Fuente. Tesis de Ingeniería Industrial – [29].

El algoritmo de aprendizaje del gentle adaboost está basado en N muestras $(x_i, y_i), \dots, (x_N, y_N)$, donde x_i son las imágenes e $y_i \in \{-1,1\}$ las salidas de los distintos CART. Al comienzo de la fase de aprendizaje, los pesos ω_i son inicializados con el valor $\omega_i = 1/N$, luego, para seleccionar el mejor CART simple se repiten los tres siguientes pasos hasta que la tasa de detección seleccionada es alcanzada (desde t=1 hasta t=T):

1.- Cada clasificador simple es ajustado a los datos mediante mínimos cuadrados y se calcula el error respecto de los pesos ω_i. **[29]**

2.- Se elige el mejor CART h_t y se incrementa el contador t. **[29]**

3.- Se actualizan los pesos con el siguiente valor: $\omega_i = \omega_i \cdot e^{-y_i h_t(x_i)}$ y se vuelven a normalizar para que el sumatorio de los pesos sea igual a la unidad. **[29]**

La salida final del clasificador es $h(x) = sign(\sum_{t=1}^{T} h_t(x))$, valor en el que se basa la construcción de la cascada de clasificadores. **[29]**

EmguCV: EmguCV es un wrapper (envoltorio) .NET multiplataforma para las librerías de procesamiento de imágenes de Intel OpenCV. Permitiendo que las funciones OpenCV puedan ser llamadas desde lenguajes .NET. EmguCV es compatible con lenguajes tales como: C#, VB, VC++, IronPython, etc. **[30]**

Además esta herramienta puede ser usada para diferentes sistemas operativos y dispositivos, sean estos según la necesidad; Windows, Linux, Mac OS X, iPhone, iPad y Android. En otras palabras EmguCV es OpenCV para entorno .NET en donde se puede trabajar con entorno gráfico ideal para trabajar con aplicaciones Windows Forms de Microsoft Visual Studio. Las librerías están escritas en lenguaje C# y tienen la misma funcionalidad de las versiones de OpenCV. La Fig. 4.38 muestra el logotipo de la librería EmguCV. **[30]**

Fig. 4.38: Logotipo de EmguCV
Fuente. http://www.emgu.com/wiki/index.php/Main_Page - [30].

A diferencia de otros wrappers como OpenCVDotNet o SharperCV, que utilizan código inseguro (clases declaradas como tipo "unsafe", lo que implica que el CLR no regula su acceso a punteros), EmguCV está escrito completamente en C#. La ventaja es que puede ser compilado en Monodevelop, y por tanto, es capaz de ejecutarse en cualquier plataforma que soporte Monodevelop, incluyendo Linux, Solaris, FreeBSD y Mac OS X, entre otros. Es necesario bajar todos los DLL's que vienen en EmguCV en la carpeta donde se ejecuta el código, para que el código pueda correr sin ningún problema. **[30]**

Instalación de EmguCV con Microsoft Visual Studio:

La instalación de esta herramienta es muy sencilla basta con extraer los archivos de las librerías en un directorio conocido. Estas librerías se las obtiene de forma gratuita de su portal electrónico oficial; en distintas versiones para desarrollo de aplicaciones en código abierto. **[30]**

Al igual que OpenCV se requiere establecer la variable de entorno para facilitar la configuración con VS2010, se debe asignar la dirección de las librerías como indica la Fig. 2.39. **[30]**

Fig. 4.39: Variable de entorno del sistema para EmguCV

Que la variable de entorno del sistema operativo sea establecida; será necesario para que la aplicación, a desarrollar pueda ser ejecutada; por lo que esta acción será realizada en cualquier ordenador en el que se desee probar el software programado en EmguCV. **[30]**

Para poder desarrollar la aplicación con VS2010 se debe agregar las referencias de EmguCV en el proyecto. En la pestaña "Proyecto" se escoge la opción "agregar referencia". En la ventana que se abre se escoge Browser y se agrega los archivos con extensión .dll contenidos dentro de la carpeta "bin" de los archivos de EmguCV. **[30]**

Para utilizar los recursos de OpenCV se debe agregar al proyecto elementos existentes, de igual forma en la pestaña "Proyecto" se selecciona la opción "Agregar elemento existente" y se selecciona los archivos cuyo nombre comienza con OpenCV que están localizados en el directorio de EmguCV bin-x86. Se verifica en el explorador de soluciones de VS2010 que los archivos se hayan añadido correctamente. **[30]**

EmguCV cuenta con componentes de .NET framework para trabajar en una aplicación Windows Forms de Visual Studio, que se debe añadir al cuadro de herramientas de VS2010. Para ello se da clic derecho sobre el cuadro de Herramientas y se escoge la

opción “elegir elementos”. En la ventana que aparece se acciona la pestaña de “examinar” y dentro de la carpeta “bin” contenida en los archivos de EmguCV escogemos el ítem Emgu.CV.UI.dll; con ello se habilitan las herramientas gráficas de EmguCV especialmente Image Box, que se usa para mostrar imágenes o video que será necesario para desarrollar la interfaz del proyecto de la Pizarra Virtual usando Realidad Aumentada. La Fig. 4.40 muestra las herramientas gráficas de EmguCV. **[30]**

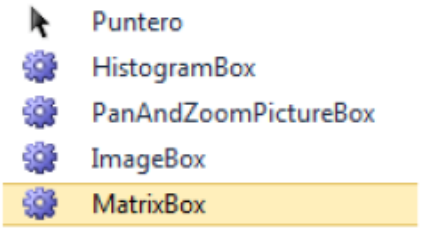


Fig. 4.40: Herramientas gráficas de EmguCV

Procesamiento de imágenes utilizando EmguCV y OpenCV:

EmguCV es una plataforma cruzada asociada a las librerías Intel OpenCV de procesamiento de imágenes. Utiliza librerías OpenCV para el procesamiento de imágenes aplicando el algoritmo PCA (Principal Component Analysis – Análisis de Componente Principal). **[30]**

Los archivos binarios .dll en la herramienta EmguCV se utilizan para:

- Emgu.Util.dll: Es una colección de utilidades .NET. **[30]**
- Emgu.CV.dll: Algoritmos básicos de procesamiento de imágenes desde OpenCV.
- Emgu.CV.UI.dll: Herramientas útiles para los controles Emgu. **[30]**
- Emgu.CV.GPU.dll: Procesamiento GPU (Nvidia Cuda). **[30]**
- Emgu.CV.ML.dll: Algoritmos de aprendizaje automático. **[30]**

Para trabajar con EmguCV se debe definir primero el lenguaje a usar, sin embargo su arquitectura en general es muy parecida al OpenCV por ello se debe simplemente hacer una migración de código encontrando la analogía con C++ hacia el lenguaje en el que se trabaje con EmguCV. Esta herramienta es muy amigable de usar una vez conocida la forma de trabajo con OpenCV. **[30]**

El Análisis de componentes principales (PCA) busca un modelo que mejor describe una cara mediante la extracción de la información más relevante contenida en una imagen. La Fig. 4.41 detalla la estructura de la arquitectura EmguCV. **[30]**

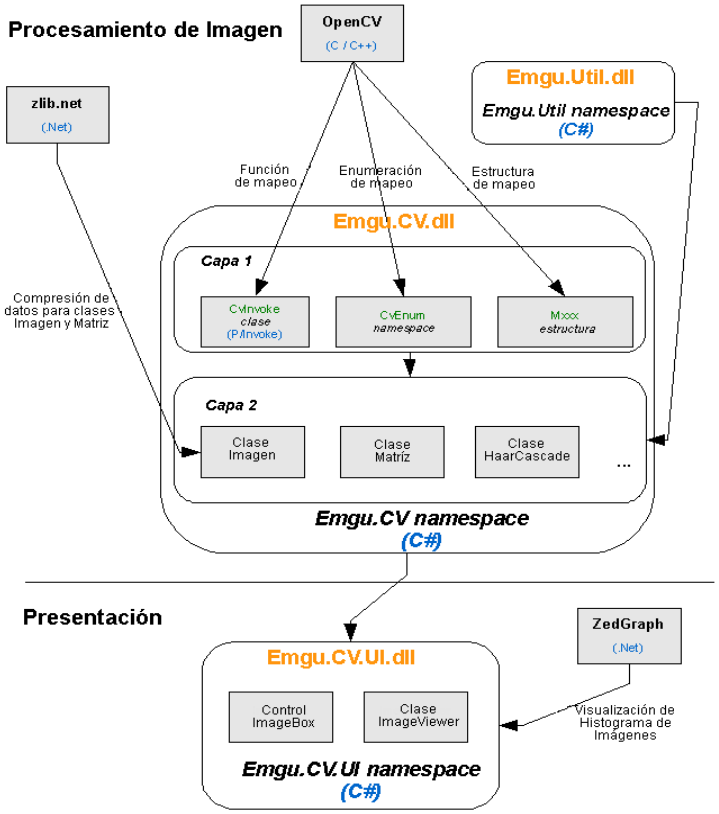


Fig. 4.41: Estructura de la arquitectura EmguCV
Fuente. http://www.emgu.com/wiki/index.php/Main_Page - [30].

Asignación de estructuras en EmguCV:

El namespace Emgu.CV.Structure contiene todas las asignaciones directas a las estructuras básicas OpenCV. La Tabla 4.4 muestra unos ejemplos de asignación de estructuras. **[30]**

Tabla 4.4: Asignación de estructuras en EmguCV

Estructura EmguCV	Estructura OpenCV
Emgu.CV.Structure.MIplImage	IplImage
Emgu.CV.Structure.MCvMat	CvMat
Emgu.CV.Structure.Mxxx	xxxx

Fuente. Herramientas gráficas de EmguCV – [30]
Elaborado por. Jimena Caguana

El prefijo M en la Tabla 4.4 significa estructura administrada. Lo mejor es que EmguCV mezcla gráficos .NET y las estructuras OpenCV juntas. **[30]**

Codificación en EmguCV:

Las funciones y clases son análogas a las usadas en OpenCV, simplemente usando los argumentos con las clases de EmguCV acorde al lenguaje utilizado. La principal clase a ser usada es CvInvoque que provee una vía directa para llamar a funciones dentro de OpenCV hacia lenguajes .NET, donde cada método de la clase CvInvoque corresponde a una función de OpenCV con el mismo nombre. **[30]**

Como referencia para usar EmguCV se tiene en la Tabla 4.5 la forma de usar estructuras de OpenCV. **[30]**

Tabla 4.5: Estructuras equivalentes en .NET a OpenCV

Estructura.NET	Estructura OpenCV
System.Drawing.Point	CvPoint
System.Drawing.PointF	CvPoint2D32f
System.Drawing.Size	CvSize
System.Drawing.Rectangle	CvRect

Fuente. Herramientas gráficas de EmguCV – [30]
Elaborado por. Jimena Caguana

Asignación de enumeraciones en EmguCV:

El namespace Emgu.CV.CvEnum contiene todas las asignaciones directas a enumeraciones OpenCV. Por ejemplo:

```
CvEnum.IPL_DEPTH.IPL_DEPTH_16U
```

Esta enumeración comparte el mismo valor en OpenCV de la siguiente manera:

```
IPL_DEPTH_16U
```

Ambas numeraciones son 16 unidades. Se puede ver que cada tipo de enumeración EmguCV y OpenCV tiene el mismo nombre. Eso es lo que significa asignación directa, a causa de esto programadores OpenCV obtendrán acceso rápido a EmguCV. **[30]**

La clase CvInvoke, proporciona una manera de invocar directamente a una función de OpenCV desde .NET. Cada método de la clase corresponde a una función en OpenCV del mismo nombre. En el siguiente código se muestra una llamada a una función de la clase CvInvoke en OpenCV. **[30]**

```
using Emgu.CV; //EmguCV implementa el wrapper con las funciones para OpenCV
//Función Mapping-Emgu.CV.CvInvoke
IntPtr image = CvInvoke.cvCreateImage(new MCvSize(400, 300),
CvEnum.IPL_DEPTH.IPL_DEPTH_8U,1);
```

Que es equivalente a la siguiente llamada de una función en C:

```
IplImage* image = cvCreateImage(cvSize(400,300),IPL_DEPTH_8U,1);
```

A continuación se mostrará un ejemplo de un simple código escrito en C# que realiza la creación y muestra de una ventana con el texto "Hello Word". **[30]**

```
using Emgu.CV;
using Emgu.CV.CvEnum;
...
String win1 = "Test Window"; //Nombre de la ventana
CvInvoke.cvNamedWindow(win1); //Creación de una ventana con un nombre
específico
Image<Bgr, Byte> img = new Image<Brg, byte>(400,200, new Bgr(255,0,0));
//Creación de una imagen de 400x200 color azul
MCvFont f= new MCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1.0, 1.0); //Creación de
la fuente
img.Draw("Hello, world", ref f, new Point2D<int>(10,80), new Bgr(0,255,0));
//Escribe "Hello, world" en la imagen usando una fuente específica
CvInvoke.cvShowImage(win1, img); //Muestra la imagen
CvInvoke.cvWaitKey(0); //Espera a que el usuario pulse una tecla
CvInvoke.cvDestroyWindow(win1); //Destruye la ventana
```

AForge.NET: Es un framework de C# diseñado para desarrolladores e investigadores en los campos de la Visión por Computadora e Inteligencia Artificial-procesamiento de imágenes, visión artificial, redes neuronales, algoritmos genéticos, aprendizaje automático, etc. La Fig. 4.42 muestra el logotipo de las librerías AForge.NET. **[31]**

Fig. 4.42: Logotipo de AForge.NET
Fuente. http://www.aforgenet.com/img/logo.gif - [31].

AForge.NET contiene una interfaz para el tratamiento de imágenes dentro del ensamblado AForge.Imaging.Filters. **[31]**

Instalación de AForge.NET con Microsoft Visual Studio:

La instalación de estas librerías es muy sencilla basta con extraer los archivos de las librerías en un directorio conocido. Esta herramienta se las obtiene de forma gratuita de su portal electrónico oficial; en distintas versiones para desarrollo de aplicaciones en código abierto. Una vez descargado el archivo ejecutable del programa se lo instala y se puede ver los archivos binarios en la ruta C:\Archivos de programa\AForge.NET\Framework\Release; para el sistema operativo Windows de 32 bits. La Fig. 4.43 muestra los archivos de la librería AForge.NET. **[31]**

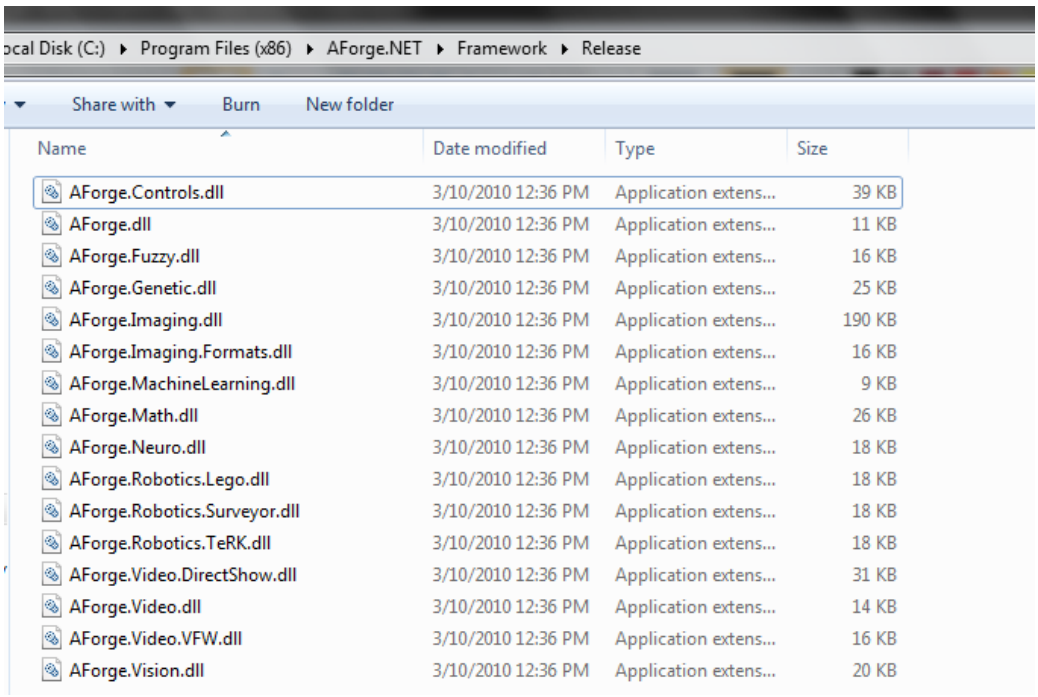


Fig. 4.43: Archivos de la librería AForge.NET

Al crear un nuevo proyecto Windows Forms se debe añadir las librerías AForge.NET como referencia; para esto se hace clic derecho en referencias del proyecto en el explorador de soluciones. La Fig. 4.44 muestra el agregado de referencia en Visual Studio C#. **[31]**

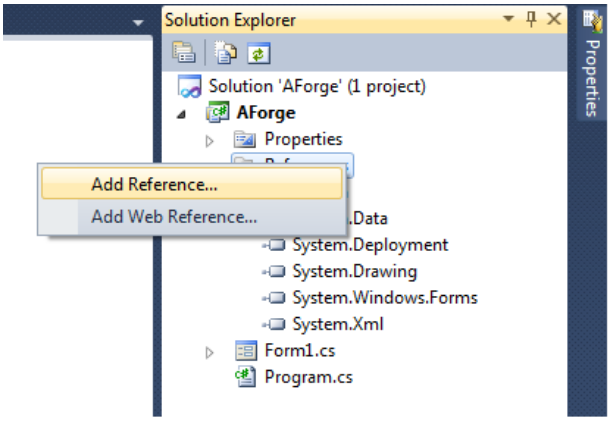


Fig. 4.44: Agregado de referencia en Visual Studio C#

Se hace una exploración a la carpeta donde se instaló las librerías AForge.NET para Windows de 32 bits semejante a la siguiente ruta C:\Archivos de programa\AForge.NET\Framework\Release; y para Windows de 64 bits similar a la siguiente ruta C:\Archivos de programa\(x86)\AForge.NET\Framework\Release. La Fig. 4.45 las referencias .dll para trabajar con AForge.NET. **[31]**

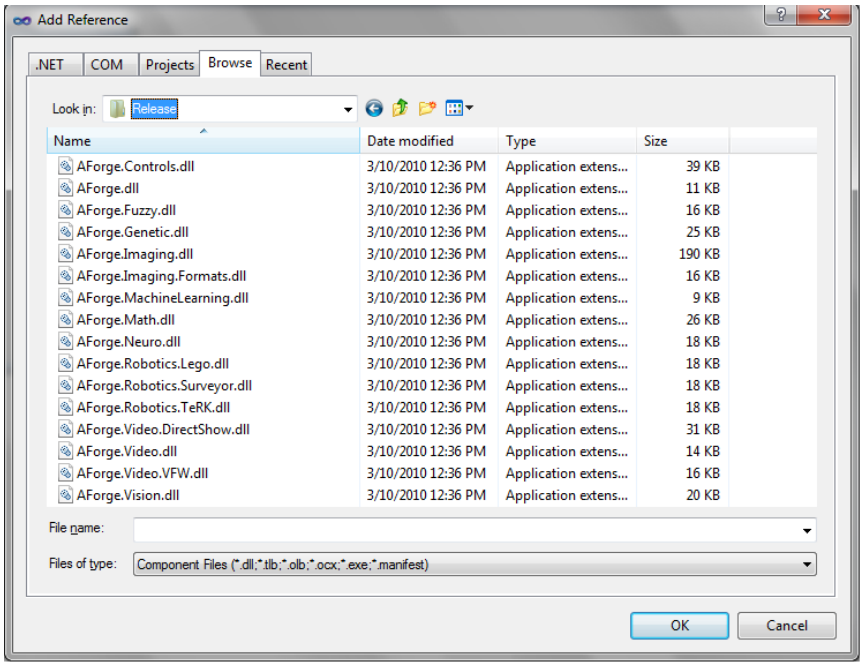


Fig. 4.45: Referencia .dll para trabajar con AForge.NET

A partir de la ruta se agregan como referencia los binarios necesarios (.dll) para el tipo de aplicación en la que se esté trabajando, por ejemplo para trabajar con una webcam Aforge.Video.dll y Aforge.DirectShow.dll. La Fig. 4.46 muestra los archivos binarios para trabajar con una webcam. **[31]**

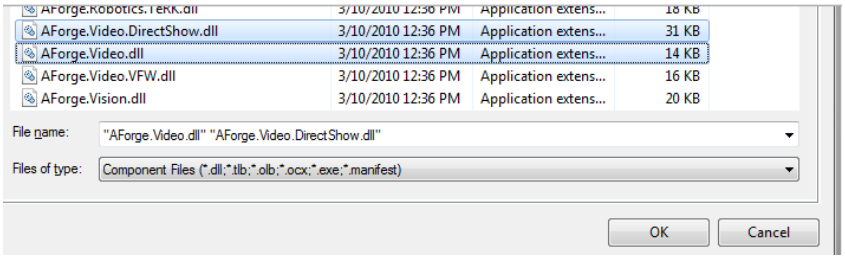


Fig. 4.46: Binarios para trabajar con una webcam

Al hacer clic en "Aceptar" se debe ser capaz de ver los binarios (.dll) que figuran en el explorador de soluciones del proyecto, bajo la carpeta de referencia como muestra la Fig. 4.47. **[31]**

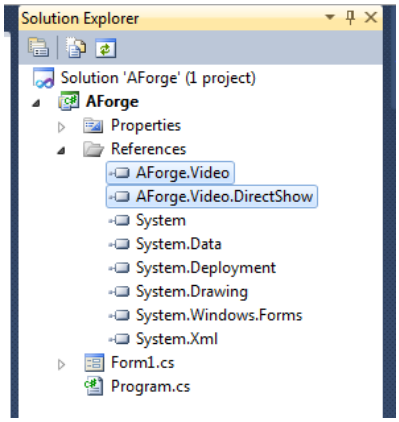


Fig. 4.47: Visualización de las referencias agregadas

Una vez que se muestren en referencia las librerías AForge.NET para la aplicación Windows Forms que se esté trabajando, se puede programar con ellas y construir cualquier interfaz que haga uso de ellas sin inconvenientes de depuración o ejecución, en esta ocasión no hace falta abrir la ruta en las variables de entorno del sistema operativo como ocurrió con OpenCV y EmguCV respectivamente.

4.6.5 Diseño de la pizarra

La pizarra es una herramienta común ordinaria conocida como pizarrón blanco, o pizarra para marcador; la cual tiene una estructura de tablero rectangular de color blanco usado para dibujar las esquinas que serán reconocidas por el sistema de visión artificial de la Pizarra Virtual usando Realidad Aumentada; y a su vez, usada para escribir la letra respectiva para la proyección del modelo 3D; además de ser el cuadro de enfoque para la detección y reconocimiento de los respectivos marcadores de realidad aumentada. La

constitución de la pizarra utilizada con fines prácticos en el desarrollo y consecución del Trabajo de Graduación, está detallada como se muestra en la Fig. 4.48.

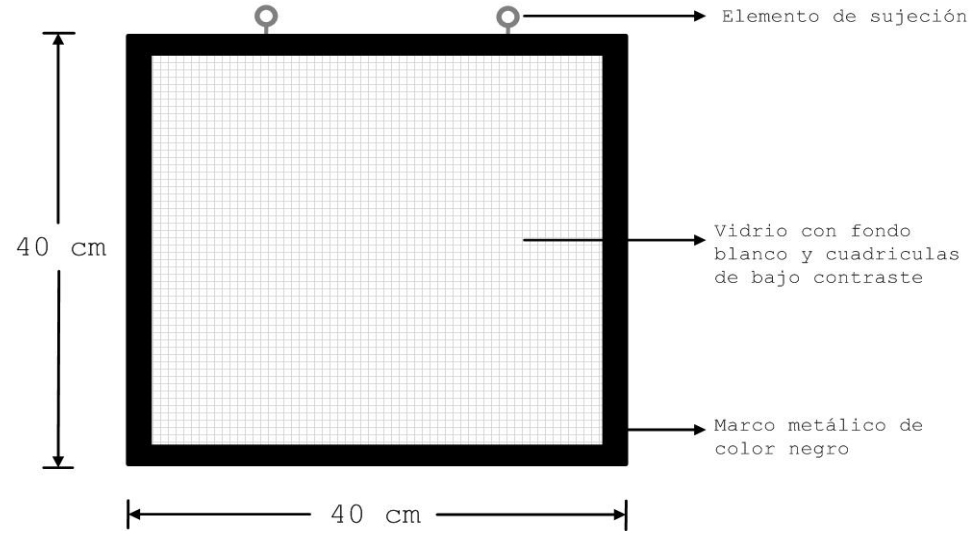


Fig. 4.48: Pizarra de marcador

NOTA: La pizarra como instrumento común dentro del aula de clase, puede ser empleada en el software de la Pizarra Virtual usando Realidad Aumentada siempre y cuando cumpla con características semejantes a la pizarra modelo de la Fig. 4.48, en el sentido compositivo más no longitudinal o de tamaño, y sirva para escribir con un rotulador de fácil borrado.

4.6.6 Pruebas

Permite tener una visión general de los niveles, requisitos, codificación, técnicas y actividades evaluativas de la Pizarra Virtual usando Realidad Aumentada con el propósito de cumplir el objetivo general y los objetivos específicos de la propuesta. Las pruebas del software se realizó de manera itinerante en el Laboratorio Nº 1 de Computación de la Unidad Educativa "Tirso de Molina".

Las pruebas de la Pizarra Virtual usando Realidad Aumentada como metodología técnica de comprobación dinámica, implicaron la ejecución del software iterativamente para evaluar la calidad del programa y mejorarlo identificando erratas y problemas. El Estudio de Usabilidad (ANEXO B) complementa las pruebas realizadas a la aplicación de la pizarra. El ámbito de las pruebas varió en tres niveles, según lo recomienda la ingeniería de software para herramientas de Visión Artificial:

Módulo único: Pruebas unitarias realizadas en cada interfaz gráfica de usuario. El acceso al código fuente fue relevante en la ejecución de esta prueba para verificación del funcionamiento aislado de cada componente y su eficaz depuración. **[32]**

Grupo de módulos: Pruebas de integración realizada al entrelazado de cada objeto, clase, componente y formulario que representan el software de la Pizarra Virtual usando Realidad Aumentada. Se efectuaron pruebas de tipo incremental ascendente (bottom-up) e incremental descendente (top-down) para optimización de resultados. **[32]**

Sistema Completo: Pruebas de sistema realizada en el funcionamiento del software y hardware de la pizarra, detección de esquinas, escritura con el rotulador, reconocimiento y detección de escritura y marcadores de realidad aumentada. Este nivel permitió comprobar requisitos no funcionales como: Velocidad, exactitud, fiabilidad y entorno operativo. **[32]**

4.6.7 Verificación

Para la implementación óptima de la Pizarra Virtual usando Realidad Aumentada, se ejecutó los siguientes tipos de verificaciones recomendadas en la ingeniería de software.

Basadas en uso: Entorno en el que trabajará el software de la pizarra.

- *Perfil operacional:* Entorno operativo en que deberá funcionar el programa; al ser un software de visión artificial y realidad aumenta se tomó mayor énfasis la iluminación y el tracking de la cámara web. **[32]**
- *SRET (Software Reliability Engineered Testing):* Este tipo de evaluación, verificación y validación sobreponen la fiabilidad y ergonomía de la pizarra virtual con respecto a las funcionalidades de RA y visión computacional. **[32]**

Basadas en los casos de uso: Ingeniería de requisitos del software de la pizarra.

- *Casos de verificación:* Este tipo de verificación se centró en acciones visibles al usuario, obtenidos a partir de los casos de uso, diagramas de flujo y análisis de requisitos. **[32]**

4.6.8 Mantenimiento

El estándar IEEE 1219 define el Mantenimiento del Software como "la modificación de un producto software con el fin de corregir defectos, mejorar el rendimiento u otros atributos". **[32]**

En el estándar ISO 12207, de Procesos del Ciclo de Vida del software se establece que "el proceso de Mantenimiento se activa cuando el producto de software sufre modificaciones en el código y la documentación asociada, debido a un problema o a la necesidad de mejora o adaptación". Antes de la respectiva implementación de la Pizarra Virtual usando Realidad Aumentada se realizó cuatro tipos de mantenimiento del software de la pizarra, acorde a lo estipulado en ingeniería de software:

Mantenimiento Correctivo: Se localizó y eliminó los posibles defectos del programa; defecto que se constituye en una característica potencial de causar un fallo. **[32]**

- *Procesamiento:* Salidas incorrectas del programa, NO detección y reconocimiento de los marcadores de realidad aumentada. **[32]**
- *Rendimiento:* Tiempo de respuesta demasiado alto en procesamiento de imágenes y visión artificial. **[32]**
- *Programación:* Inconsistencia en el diseño de clases y declaración de variables. **[32]**
- *Documentación:* Inconsistencia entre la funcionalidad del programa y el manual de usuario. **[32]**

Mantenimiento Adaptativo: Se identificó y modificó el programa debido a cambios en el entorno (localidad, hardware o software) en el cual se ejecutará. El mantenimiento adaptativo es cada vez más usual debido principalmente al cambio, cada vez más rápido, en los diversos aspectos de la informática: nuevas generaciones de hardware cada dos años, nuevos sistemas operativos, nuevas versiones de software y mejoras en los periféricos o en otros elementos del sistema. Frente a esto, la vida útil de un sistema software puede superar fácilmente los diez años (Pressman, 1993). **[32]**

- *Entorno de los procesos:* Migrando a una nueva plataforma de desarrollo con componentes distribuidos y amigables al programador; así como al producto software final para el usuario. El software de la Pizarra Virtual usando Realidad

Aumentada se inició en Visual C++ y se la migró a Visual C# para mejores condiciones de perfeccionamiento y calidad. **[32]**

Mantenimiento Perfectivo: Cambios en la especificación, normalmente debidos a cambios en los requisitos del producto software. El progreso efectivo del software de la Pizarra Virtual usando Realidad Aumentada implicó la incorporación de nuevos módulos aplicativos desde sus versiones iniciales, adquiriendo nuevas funcionalidades que trascienden al prototipo a una versión final de excelentes prestaciones. **[32]**

- *Mantenimiento perfectivo de ampliación:* Orientado a la incorporación de nuevas funcionalidades. **[32]**
- *Mantenimiento perfectivo de eficiencia:* Que busca la mejora de la eficiencia de ejecución. **[32]**

Mantenimiento Preventivo: Consiste en la modificación del software para mejorar sus propiedades sin alterar sus especificaciones funcionales. El programa de la Pizarra Virtual usando Realidad Aumentada tuvo que agrupar la capacidad de uso del rotulador y marcadores de realidad aumentada en una misma aplicación informática. **[32]**

- *Mantenimiento para la reutilización:* El software de la Pizarra Virtual usando Realidad Aumentada está constituido con bibliotecas para ser fácilmente reutilizable, especializando su capacidad de reusabilidad. **[32]**

4.7. Pizarra Virtual usando Realidad Aumentada

Para el desarrollo y posterior implementación de la pizarra virtual usando Realidad Aumentada, se ha tomado la compresión léxica del nombre, a PIVRA; que significa (**PI**zarra **V**irtual usando **R**ealidad **A**umentada).

4.7.1 Introducción

En los últimos años; la Visión Artificial, ha dado origen al procesamiento de imágenes de video y a la Realidad Aumentada, con el fin de integrar objetos sintéticos en imágenes reales. La Visión Artificial proporciona a las máquinas capacidades visuales similares a los de los seres humanos, esta característica es de uso masivo en robótica; pero a partir de su desarrollo y su capacidad cualitativa de expansión, ha trascendido a

la Realidad Aumentada, categoría que inicio como una actividad técnica de efectos especiales, en el cine; añadiendo algo que no existe en imágenes captadas de la realidad.

4.7.2 Arquitectura general del sistema PIVRA

La Pizarra Virtual usando Realidad Aumentada en forma general, está constituido con el siguiente diagrama de bloques, según lo muestra la Fig. 4.49.

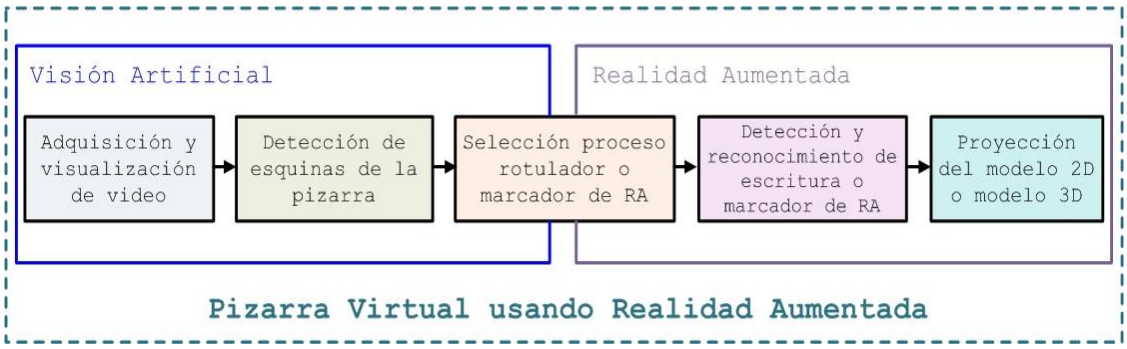


Fig. 4.49: Arquitectura general del sistema PIVRA

Adquisición y visualización de video: En esta etapa del desarrollo del software PIVRA, se realizó el accesos a la cámara web y la visualización del video en la GUI principal de la solución creada en Visual C#.

Detección de esquinas de la pizarra: A partir de la adquisición y visualización del video, se implementó un algoritmo para detectar y reconocer la letra X escrita con un rotulador en las esquinas de la pizarra; estas letras encerradas en un recuadro otorgan el área de trabajo para la escritura de la letra de reconocimiento para realidad aumentada con un rotulador, la muestra del marcador de realidad aumentada y la proyección del modelo 2D o modelo 3D.

Selección procesos rotulador o marcador de Realidad Aumentada: A partir del área de trabajo, en este ítem del software PIVRA se permitirá escoger la opción; escribir la letra para su posterior detección y proyección del modelo 2D o modelo 3D; o la opción en la que se muestra un marcador de realidad aumentada para la detección y reconocimiento del software con el propósito de proyectar encima del mismo, el modelo 2D o modelo 3D respectivo.

Detección y reconocimiento de escritura o marcador de Realidad Aumentada: En esta etapa del software PIVRA se realizó un algoritmo que detecta y reconoce la letra correspondiente a cada modelo 2D o modelo 3D. Además, el algoritmo para detectar y reconocer el respectivo marcador de realidad aumentada para la proyección sobre el mismo del modelo 2D o modelo 3D correspondiente.

Proyección del modelo 2D o modelo 3D: Una vez detectada y reconocida la letra escrita o el marcador de realidad aumentada; en esta etapa del software PIVRA se construyó un algoritmo que proyecte sobre la letra o el marcador de RA, el correspondiente modelo 2D o modelo 3D. Estos modelos constituyen imágenes digitales de alta definición en dos o tres dimensiones respectivamente.

4.7.3 Arquitectura detallada del sistema PIVRA

Para la implementación del sistema PIVRA, se ha cumplido con la metodología mostrada en el diagrama de bloques de la Fig. 4.50.

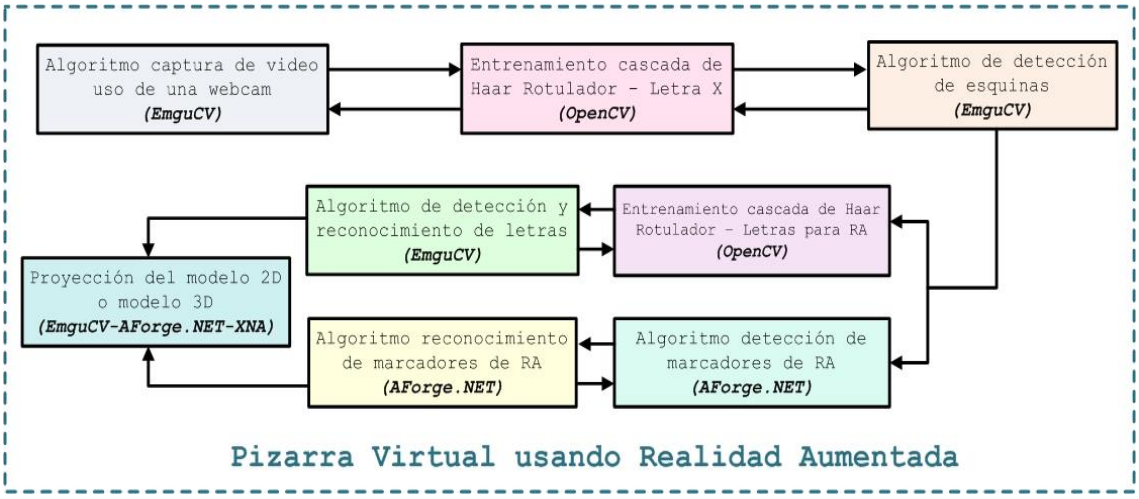


Fig. 4.50: Arquitectura detallada del sistema PIVRA

4.8. Implementación del sistema PIVRA

Describe el contenido informático eficiente a la implementación del sistema PIVRA.

4.8.1 Captura de video uso de una webcam

El algoritmo para la captura y visualización de video a través de una cámara web, es semejante al implementado en el sistema PIVRA. El código de programación adquiere y

visualiza el video a través de la webcam y lo muestra en un Imagebox y también procesa la imagen, y la muestra en otro Imagebox. Se procede con crear una aplicación Windows Forms en blanco. La Fig. 4.51 muestra creación de una aplicación Windows Forms.

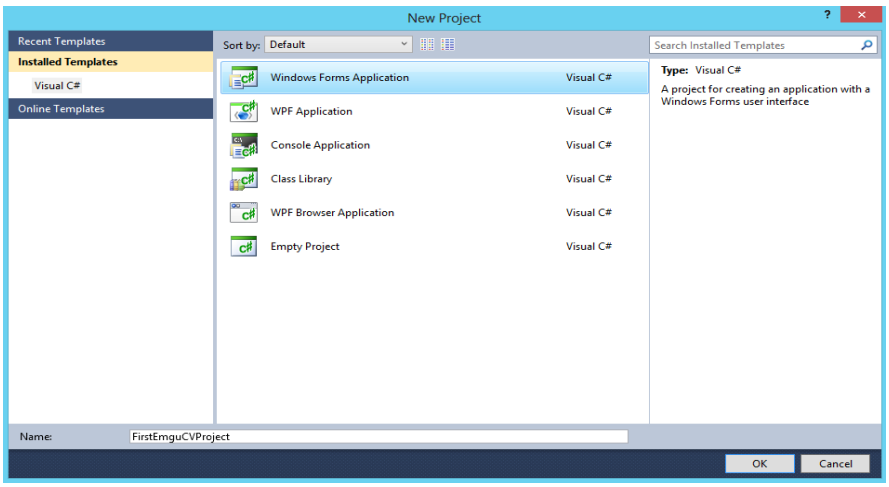


Fig. 4.51: Creación de una aplicación Windows Forms

El siguiente paso es agregar las referencias EmguCV, para ello; busque la carpeta "bin" EmguCV (por defecto se encuentra en C:\Emgu\emgucv-windows-x86 2.4.9 \bin), en la carpeta "bin" debe haber algunos .dll; añadir todos aquellos que comienzan con "Emgu.CV" (elegir Emgu.CV.DebuggerVisualizers.VS2010.dll según el Visual Studio que está utilizando; independiente de las tipologías (Express, Ultimate, Professional, etc.) para el caso del sistema PIVRA, es Emgu.CV.DebuggerVisualizers.VS2010.dll). La Fig. 4.52 muestra las referencias del proyecto hacia EmguCV.

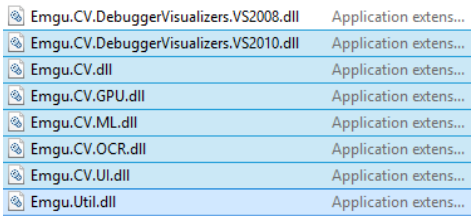


Fig. 4.52: Referencias del Proyecto hacia EmguCV

Ahora se tiene que añadir algunos elementos existentes, se ubica en el explorador de la solución; proyecto, clic derecho; añadir elementos existentes y se busca nuevamente el directorio "bin" de EmguCV y esta vez se elige todos los archivos .dll que empiezan por "opencv_"; se requiere estos archivos .dll para cada tiempo de la salida generado a

través de EmguCV, por eso se les agrega al directorio del proyecto; también se realiza un cambio a las propiedades a fin de que se copien siempre a la carpeta de salida. Por lo tanto, se selecciona todos los .dll agregados y en el explorador de propiedades se cambia la propiedad "Copiar a directorio de salida" por "Copiar siempre". La Fig. 4.53 detalla la ventana agregar archivos binarios existentes .dll de EmguCV.

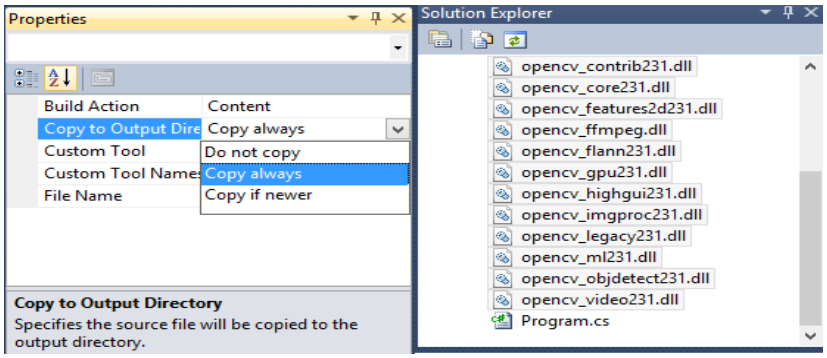


Fig. 4.53: Agregar archivos binarios existentes .dll de EmguCV

Ya se ha agregado los controles personalizados EmguCV a la barra de herramientas, ahora se debe diseñar el formulario, para ello se utilizan dos ImageBox (controles EmguCV), un botón y un cuadro de texto (controles C#), el diseño GUI como ejemplo queda semejante al de la Fig. 4.54.

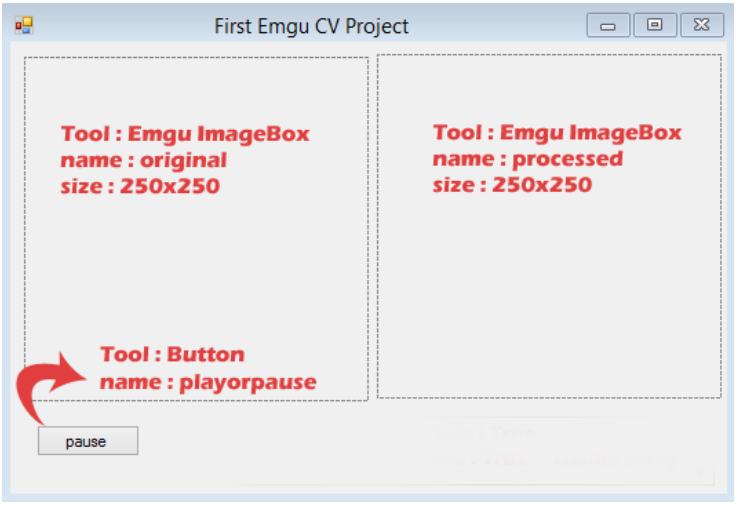


Fig. 4.54: GUI ejemplo para captura de video en EmguCV

Codificación:

Ahora se converge de la vista de diseño del proyecto, a la vista de código Form1.cs, y se añade los siguientes namespaces;

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.UI;
```

Después se declaran algunas variables generales:

```
Capture capturecam = null; //Instancia para la captura usando cámara web
bool CapturingProcess = false;  //Boolean que indica el estado del proceso de
captura
Image<Bgr, Byte> imgOrg;       //Imagen tipo RGB (o Bgr como en OpenCV)
Image<Gray, Byte> imgProc;    // la imagen procesada será una imagen en modo
escala de grises
```

Ahora es el momento de añadir en el evento Form Load, el inicio de la captura; a través de una webcam.

```
capturecam = new Capture();
```

Esto va a asociar la cámara web por defecto con el objeto capturecam.

```
private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
                capturecam = new Capture();
            }
            catch (NullReferenceException exception)
            {
                MessageBox.Show(exception.Message);
                return;
            }
            Application.Idle += new EventHandler(ProcessFunction);
            CapturingProcess = true;
        }
        void ProcessFunction(object sender, EventArgs e)
        {
            imgOrg = capturecam.QueryFrame();
            if (imgOrg == null) return;
            imgProc = imgOrg.InRange(new Bgr(50, 50, 50), new Bgr(255, 255,
255));
            imgProc = imgProc.SmoothGaussian(9);
            original.Image = imgOrg;
            processed.Image = imgProc;
        }
```

Se ha añadido el código para capturar objetos en el bloque *try* y *catch* para evitar el error si la cámara ya está en uso. Además, un controlador de eventos para *Application.Idle*, por lo que realiza la tarea cuando está inactivo, que es conseguir el siguiente fotograma. *ProcessFunction* es llamada en cada estado de inactividad de la

aplicación. La función *QueryFrame* obtiene el siguiente fotograma de la cámara web, si es nulo significa que hay algún problema y por lo tanto se detiene la aplicación en ese estado.

La función *InRange* toma dos parámetros, el rango mínimo y el rango máximo de Bgr. La función *SmoothGaussian* aplica el suavizado de Gauss con la longitud = 9 x, y; se puede también pasar distintos parámetros para x e y.

Ahora en la parte de codificación del botón:

```
private void playorpause_Click(object sender, EventArgs e)
        {
            if (CapturingProcess == true)
            {
                Application.Idle -= ProcessFunction;
                CapturingProcess = false;
                playorpause.Text = "Play";
            }
            else
            {
                Application.Idle += ProcessFunction;
                CapturingProcess = true;
                playorpause.Text = "Pause";
            }
        }
```

Esta es la parte más sencilla de la programación, comprueba el valor booleano de *CapturingProcess* y por consiguiente cambia el texto del botón, además detiene la transmisión de la webcam.

4.8.2 Entrenamiento cascada de Haar

El entrenamiento de una cascada Haar sirve para conseguir que un computador sea capaz de reconocer objetos haciendo uso de un único archivo y un programa que lo ejecuta. Para que una cascada de Haar funcione es necesario entrenarla mediante muestras; se necesitan dos tipos de muestras: positivas y negativas. **[33]**

Las muestras positivas son imágenes recortadas que contienen el objeto a detectar. En el caso del sistema PIVRA, las muestras positivas deben ser un conjunto de imágenes recortadas con vistas frontales de la letra X. Las muestras negativas son imágenes que NO contienen el objeto a detectar; es decir, fondos. Lo más apropiado para el sistema

PIVRA son imágenes de pizarrones de marcador vacías, aunque, en general; cualquier imagen que no contenga el objeto de interés es válida. **[33]**

Para un buen funcionamiento de la cascada es necesario recolectar entre 1.000 y 10.000 muestras de cada tipo. En el caso del sistema PIVRA se hizo un entrenamiento Haar con 1.050 muestras positivas y 2.050 muestras negativas.

Primer paso: Creación de los ficheros index

Una vez se tengan las muestras, lo siguiente es generar dos archivos *index* (.idx), a través del bloc de notas o cualquier otra aplicación de escritura informática. Para las muestras positivas, el archivo de texto debe ser de la forma:

<path>/img_name_1 count_1 $x_{11}\, y_{11}\, w_{11}\, h_{11}\, x_{12}\, y_{12}$ …
<path>/img_name_2 count_2 $x_{21}\, y_{21}\, w_{21}\, h_{21}\, x_{22}\, y_{22}$ …

Donde <*path*> es la ruta del directorio donde se encuentra la imagen e *img_name_1* es el nombre de la imagen (con su extensión). A continuación se debe indicar el número de objetos de interés presentes en la imagen (*count_1*) y su correspondiente ubicación dentro de la imagen, mediante las coordenadas x-y del vértice superior izquierdo, el ancho y el alto de un rectángulo que lo contiene; es decir, $x_{11}\, y_{11}\, w_{11}\, h_{11}$ respectivamente. **[33]**

Por ejemplo, si la colección de imágenes está ubicada en el directorio *C:\imagenes\equis*; el fichero index tendrá un aspecto parecido al que sigue:

```
C:\imagenes\equis\equis1.bmp 1 65 99 25 37
C:\imagenes\equis\equis2.bmp 1 140 112 28 45
```

O, si se dispone de imágenes con más de una letra por foto:

```
C:\imagenes\equis\equis3.bmp 2 153 220 45 48 14 28 50 53
```

Una vez escrito el archivo de texto, sólo queda guardarlo como index; para ello se escribe el nombre que se desee seguido de la extensión .idx. Por ejemplo, llamando *equis.idx* al archivo se tendrá algo como la Fig. 4.55. **[33]**

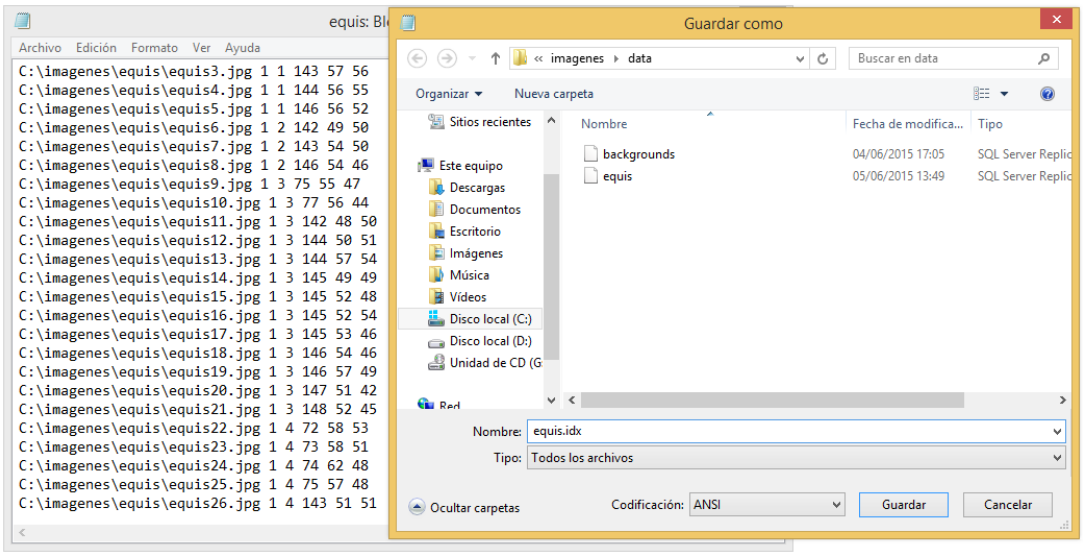


Fig. 4.55: Ejemplo fichero index

Al hacer clic en el botón guardar, se conseguiría el primer fichero index buscado. Cabe destacar que existe una manera alternativa y más rápida de generar las muestras positivas y el archivo index; es a través del programa *Imageclipper* que se detallará más adelante. **[33]**

Para generar el segundo archivo se hace uso de las muestras negativas. El proceso es similar al explicado anteriormente, salvo por el hecho que en las muestras negativas no está presente el objeto de interés y, por tanto, no se deben añadir los términos *count_1* $x_{11}\ y_{11}\ w_{11}\ h_{11}$, quedando el archivo de texto como sigue:

```
C:\imagenes\fondos\fondo1.bmp
C:\imagenes\fondos\fondo2.bmp
```

Se elige un nombre para el archivo y se guarda con la extensión .idx. **[33]**

Segundo paso: Creación de un fichero vector

Por ahora no se ocupa el archivo index generado mediante las muestras negativas; es decir, se usará por ahora únicamente el fichero equis.idx a continuación. **[33]**

Con el fichero de las muestras positivas se debe generar un fichero *vector* (.vec) que se usará para entrenar la cascada. Este fichero, se crea usando la aplicación incluida en las librerías de OpenCV: *opencv_createsamples*. **[33]**

La aplicación *opencv_createsamples* extrae las muestras positivas de las imágenes, las normaliza y las reescala al tamaño indicado y genera un archivo de salida con extensión .vec para poder entrenar la cascada. **[33]**

Esta aplicación tiene más funcionalidades, como la posibilidad de aplicar transformaciones geométricas a las muestras, añadir ruido, alterar colores, etc., tareas que permiten incrementar el número de muestras "diferentes" de las que se dispone. Estas opciones son muy útiles en caso de no disponer de un relativamente elevado número de muestras positivas, o para detectar un objeto muy concreto, como pudiera ser un logo de una compañía; ya que se evalúan distintos ángulos, rotaciones y distorsiones en la imagen. **[33]**

Volviendo al archivo vector; para generarlo, un ejemplo de la línea de comandos a escribir desde el terminal del sistema operativo sería:

```
C:\OpenCV249\bin> opencv_createsamples –vec equis.vec –info equis.idx –w 24 –
h 24
```

Donde *equis.vec* es el archivo de salida que se obtendría y *–w 24 –h 24* es el tamaño (ancho y alto) del reescalado de cada muestra extraída. **[33]**

Destacar que por defecto, la aplicación *createsamples* genera mil muestras o menos; es decir, si se dispone de más de mil muestras, y no se indica nada en la línea de comandos; *createsamples* no creará un archivo vector con el número de muestras que se tenga, para ello se debe añadir a la línea de comandos:

```
C:\OpenCV249\bin> opencv_createssamples –vec equis.vec –info equis.idx –w 24
–h 24 –num 1050
```

Con lo que se generaría, por ejemplo; 1.050 muestras. También, si se intenta crear un archivo vector con menos de mil muestras o indicándole más muestras de las que se dispone, la aplicación nos avisará con un *parse error*, este mensaje es sólo informativo, esto es, el archivo se generará correctamente, pero no con mil muestras o con el número de muestras que se le haya indicado, respectivamente; sino con tantas muestras como se disponga. **[33]**

Paso final: Entrenamiento y generación de la cascada

Una vez obtenido el archivo equis.vec del paso anterior, ya se puede entrenar y crear una cascada de Haar. Para ello, se utilizará otra aplicación suministrada con las librerías de OpenCV: *opencv_haartraining*. **[33]**

La aplicación *opencv_haartraining* genera una cascada de Haar extrayendo muestras positivas según se indica en el archivo vector; y muestras negativas aleatorias indicadas en el archivo index de los fondos. Para conseguir esto se debe escribir la siguiente línea de comandos desde el terminal del sistema operativo:

```
C:\OpenCV249\bin> opencv_haartraining –vec equis.vec –data haarcascade –w 24 –h 24 –bg backgrounds.idx –nstages 20 –nsplits 1 –minhitrate 0.998 –maxfalsealarm 0.5
```

Siendo *haarcascade* un archivo .xml de salida con la cascada resultante; es decir, lo que verdaderamente se busca después de todo el entrenamiento. **[33]**

Los parámetros *–w 24 –h 24* (ancho y alto) deben coincidir con los que se eligió en el segundo paso, la creación del archivo vector; para que no se produzca algún error, e indican lo mismo que antes, el tamaño de la extracción de cada muestra. **[33]**

El archivo *backgrounds.idx* es el fichero index con los fondos, el generado en el primer paso, *-nstages 20* significa que la cascada constará de 20 etapas, cada una formada por un clasificador robusto. Cuanto mayor sea el número de etapas que se elija, mayor será el tiempo de cómputo para la generación de la cascada. **[33]**

Por último, los parámetros *–nsplits 1 –minhitrate 0.998 –maxfalsealarm 0.5* definen características del clasificador de cada etapa, *-nsplits 1* significa el número de divisiones o ramificaciones hasta alcanzar el valor deseado; un valor 1 crea un nodo con dos posibles caminos de salida, estando uno de ellos conectado al siguiente nodo, mientras que un valor de 4 genera un árbol de decisión (Classification And Regression Tree o CART) mucho más complejo. Por supuesto, se puede incrementar este parámetro, con lo que se obtendrían mejores resultados, aunque un mayor valor significa un mayor tiempo de cómputo para generación de la cascada. La Fig. 4.56 muestra una cascada de rechazo con nsplits = 1.

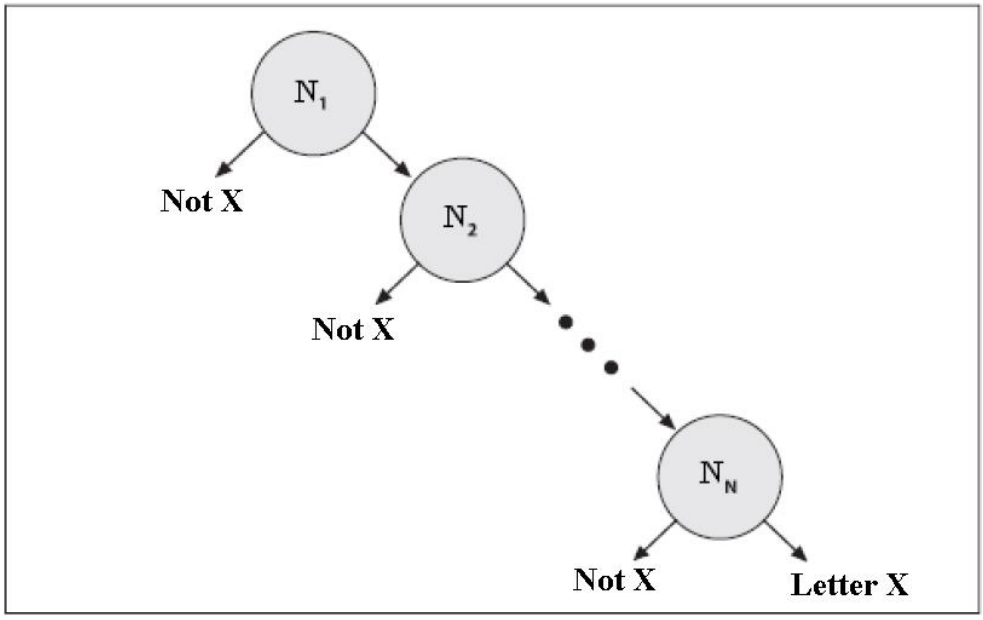


Fig. 4.56: Cascada de rechazo con nsplits = 1
Fuente. Tesis de Ingeniería Industrial – [33].

Los dos últimos parámetros indican, en tanto por uno, el mínimo valor de acierto y de fallo que se desea, respectivamente. En el ejemplo se ha tomado una tasa de acierto del 99.8% y una tasa de fallo del 50% o inferior. Una vez más, un mayor índice de acierto o una menor tasa de fallo suponen un mayor tiempo de cómputo para la generación de la cascada. **[33]**

Antes de ejecutar esta línea de comandos en el terminal, hay que tener muy presente que el entrenamiento de una cascada de Haar no es instantáneo, se ha anotado que el valor de los parámetros influye en el tiempo de cómputo, pues bien, este es un hecho muy importante; porque en función de ellos y del número de muestras que se tomen, este proceso de entrenamiento puede llevar desde horas hasta semanas, incluso en los ordenadores más potentes. **[33]**

Por supuesto, si el tiempo no es un factor decisivo, se puede crear una cascada especialmente precisa; pero, en general, hay que llegar a un acuerdo entre precisión deseada y tiempo, por lo que la elección de estos parámetros puede no ser tan trivial.

Cuando se haya decidido el valor de los parámetros se ejecuta la aplicación, para cada etapa se irá mostrando algo similar a la Fig. 4.57. **[33]**

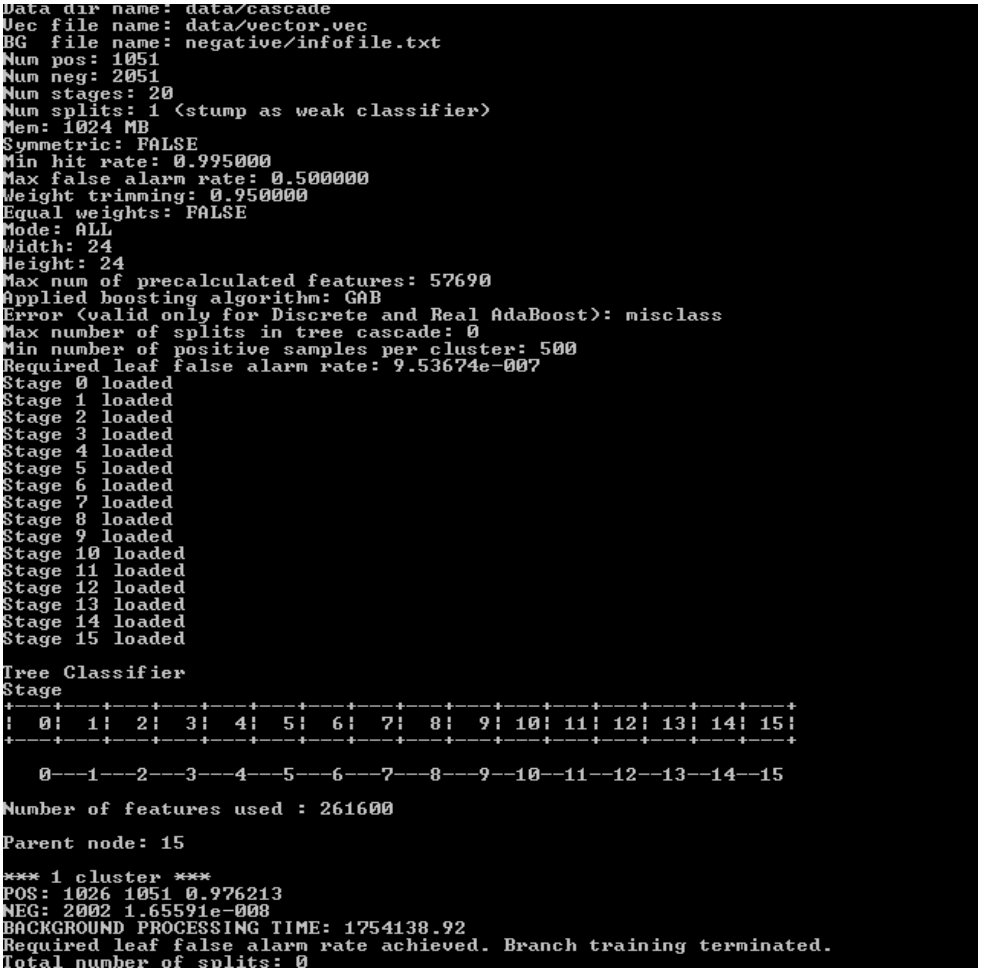


Fig. 4.57: Ejemplo ejecución opencv_haartraining

En la figura anterior se muestra la etapa 15 de la creación de una cascada de 16 etapas con *nsplits = 1*, *minhitrate = 0.998* y *maxfalsealarm = 0.5*. Como se puede observar, la etapa termina cuando *maxfalsealarm* es menor que 0.5, ya que *minhitrate* va disminuyendo muy lentamente. **[33]**

Cuando finalmente termina la ejecución de *opencv_haartraining* se mostrará un mensaje en el terminal como el de la Fig. 4.58.

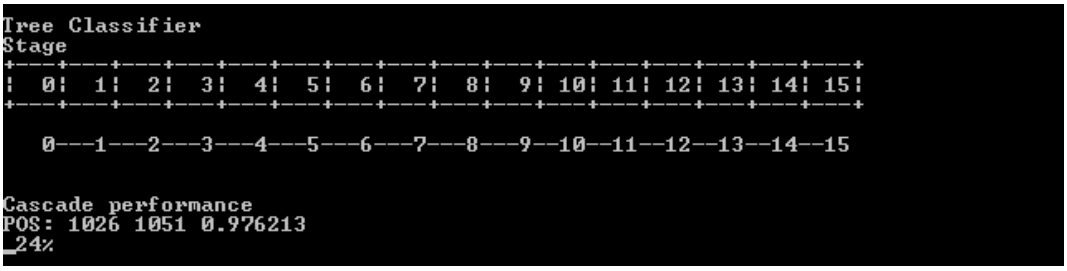


Fig. 4.58: Ejemplo fin de ejecución opencv_haartraining

Después de salir del terminal se habrá creado un directorio con el nombre *haarcascade* y un archivo .xml también con el mismo nombre. Si se inspecciona este directorio se verá que está formado por varías carpetas numeradas desde cero hasta un número menos de etapas que se haya elegido para la cascada como se muestra en la Fig. 4.59. **[33]**

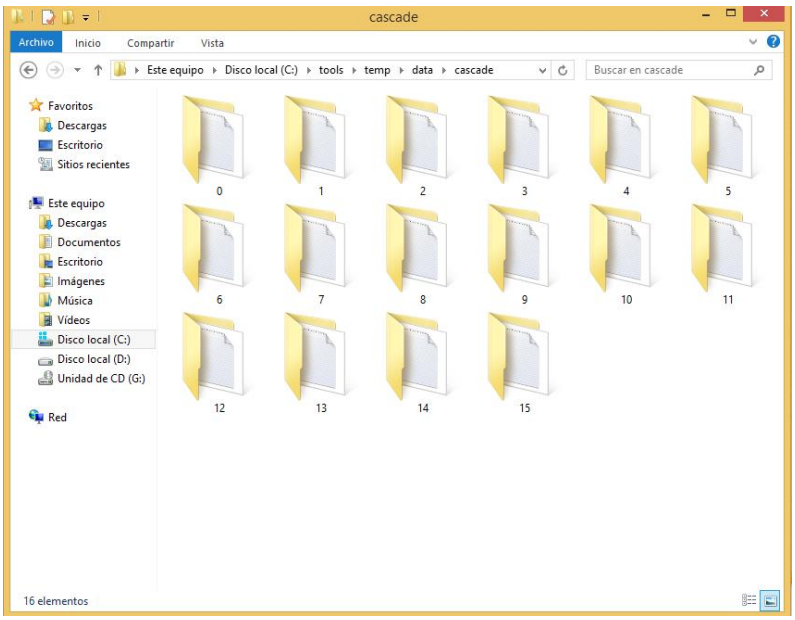


Fig. 4.59: Ejemplo directorio de la cascada

Cada una de estas carpetas contiene un archivo de texto en el que se recoge la información del Clasificador de Haar robusto de esa etapa, este clasificador se obtiene mediante la técnica del *boosting*, más concretamente la técnica del *adaboost*, desarrollada por Freud & Schapire en 1995 y posteriormente generalizada por Schapire & Singer en 1997. **[33]**

En un análisis más profundo, en realidad, no se trata del *adaboost* puramente diseñado por Freund & Schapire, sino que se trata de una variante, el denominado *gentle adaboost* para CARTs. Esta variante, es la más exitosa en el procedimiento de detección de rostros según Lienhart y otros autores; se basa en seleccionar un conjunto de CARTs simples para lograr alcanzar los valores de *minhitrate* y *maxfalsealarm* seleccionados.

Así, pues se ha conseguido crear una propia cascada de Haar para detectar la letra X en las esquinas de la pizarra. Las carpetas con los archivos de texto no serán de utilidad en el proyecto, sólo se utilizarán los archivos, como *haarcascade.xml* del ejemplo, resultantes del entrenamiento. **[33]**

La Fig. 4.60 muestra un ejemplo de una etapa del entrenamiento de la cascada. **[33]**

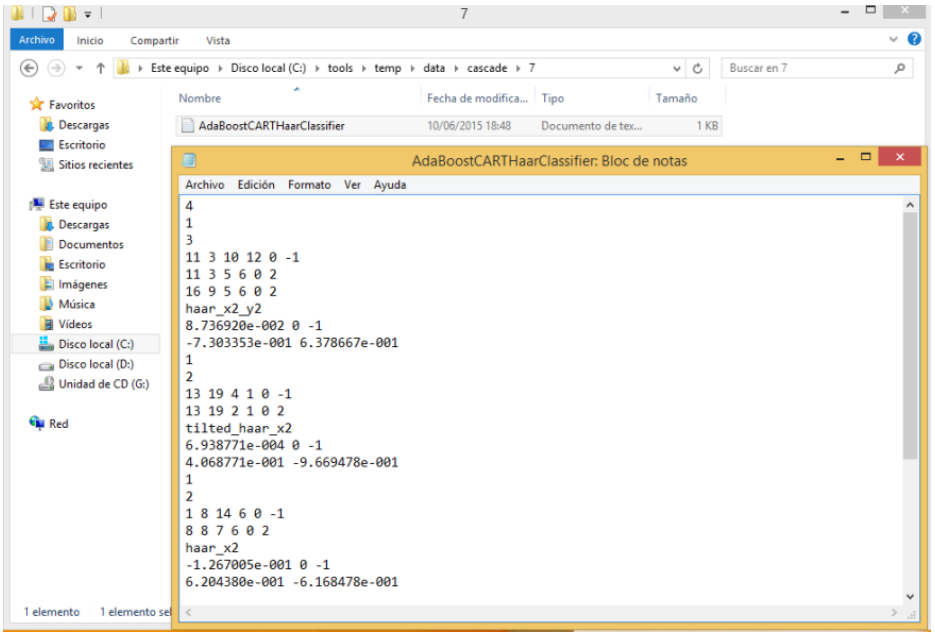


Fig. 4.60: Ejemplo etapa de la cascada

Aplicación Imageclipper

Las aportaciones al mundo científico de Naotoshi Seo, confieren el uso de la herramienta *Imageclipper*, una aplicación software libre que permite recolectar muestras positivas de forma manual pero muy rápida para el entrenamiento de nuevas cascadas de Haar; es decir, posibilita extraer aquellas zonas de la imagen con el objeto de interés que se quiere detectar. **[33]**

Este software permite:

- Abrir las imágenes de un directorio secuencialmente. **[33]**
- Abrir un video *frame* a *frame*. **[33]**
- Seleccionar una zona de la imagen o del video con el ratón y, posteriormente, guardarla y pasar a la siguiente imagen del directorio pulsando la barra espaciadora.

Estas características hacen de *Imageclipper* una herramienta de gran utilidad para el sistema PIVRA.

También hay que destacar que *Imageclipper* guarda estas imágenes recortadas en formato .png; pero lo más importante y útil es la forma particular con la que renombra

los recortes. Por ejemplo, si se tiene una imagen llamada *image.jpg*; el recorte podría tener un nombre como el siguiente:

```
image.jpg_0068_0066_0090_0080.png
```

Lo que permite generar el archivo de texto con las muestras positivas, haciendo uso del siguiente comando:

```
$find imageclipper/*_*_* -exec basename \{\}; | perl –pe\
‘s/([^_]*).*_0*(\d+)_0*(\d+)_0*(\d+)_0*(\d+)\.[^.]*$/$1 $2 $3 $4 $5\n/g’\ |
tee clipping.txt
```

Así pues, si se escribe el comando anterior en la consola de comandos se obtendrá un archivo de texto con el nombre *clipping.txt* con la siguiente información:

```
image1.jpg 68 47 89 101
image2.jpg 87 66 90 80
image3.jpg 95 105 33 32
image4.jpg 109 93 65 90
image5.jpg 117 97 52 95
```

Una última opción, permite generar un archivo .dx o .dat (ambos son válidos para realizar el entrenamiento) sin necesidad de crear un archivo de texto intermedio; para ello, sólo se necesita tener las imágenes recortadas en un directorio, el archivo *haartrainingformat.pl* adjuntado con la aplicación *Imageclipper* y escribir en la consola de comandos la siguiente línea:

```
$ \ls imageclipper/*_*_* | perl haartrainingformat.pl --ls --trim --basename
| tee info.dat
```

Lo cual generaría el archivo *info.dat* a partir del que se obtendría el archivo vector y la posterior cascada de Haar, como se explicó en el paso final del entrenamiento en párrafos anteriores. Por último, decir que la aplicación, al ser libre; puede ser descargada gratuitamente desde su sitio web. **[33]**

4.8.3 Detección y reconocimiento de esquinas

Esta iteración es la más importante del proyecto; el usuario debe limitar el frame mediante cuatro marcadores. El programa una vez los haya detectado, será capaz de tomar sus posiciones. De este modo, la aplicación muestra al usuario la zona deseada para uso de la pizarra.

Un detector de objetos requiere de un proceso previo, denominado “entrenamiento”, que se puede considerar como un aprendizaje de la máquina sobre el objeto que deberá detectar en las imágenes. El entrenamiento es el proceso donde se aplicará el boosting, y consiste en encontrar las reglas que mejor clasifiquen el objeto, y combinarlas para cada etapa del detector.

Durante el entrenamiento, distintos rasgos son extraídos de las imágenes y los rasgos más distintivos pueden ser usados para clasificar el objeto seleccionado. Esta información se introduce como parámetros del modelo estadístico. En el entrenamiento se pueden producir dos errores: no detectar un objeto (falso negativo) o detectarlo erróneamente (falso positivo).

Para que un sistema de detección de objetos se considere de buena calidad, se deben tener en cuenta básicamente dos valores, que son la probabilidad de detección y la probabilidad de falsos positivos. La primera de ellas se obtiene como el número de objetos reales que ha detectado el sistema frente el número de objetos que hay realmente en un conjunto de imágenes. Este valor será el que cualquier sistema de detección de objetos intentará maximizar.

En cuanto a la probabilidad de falso positivo, se obtendrá como el número de detecciones incorrectas que ha realizado el sistema; es decir, el número de veces que el sistema ha encontrado un objeto donde realmente no lo había, frente al número total de detecciones. Y ese valor es el que se quiere minimizar, teniendo en cuenta que ambos valores mantienen una relación directa: si uno de ellos aumenta, el otro también lo hará.

La finalidad de la detección y reconocimiento de esquinas, consiste en que la webcam sea capaz de detectar el área designada por el usuario como importante, para posteriormente mostrarla por la pantalla, rechazando el resto de información que capture del entorno. Para ello, el usuario previamente ha debido marcar las esquinas de la región que desee con una X.

Uno de los mayores riesgos a considerar en esta etapa es el tiempo. Esto es debido, a que la técnica de Haartraining es muy costosa y supone mucho tiempo de espera, sin garantizar resultados óptimos a su finalización.

Otro de los riesgos a tener en cuenta, y que está presente a lo largo de todo el desarrollo del sistema PIVRA, es la iluminación. El efecto de la luz provoca brillos y tonalidades no deseadas, lo que implica que una misma X sea reconocida o no dependiendo de si hay mucha o poca luz alrededor.

A pesar de las dificultades de iluminación, además; del tiempo y coste del entrenamiento Haar, se ha creado el entrenamiento de la cascada de Haar para la detección y reconocimiento de la letra X, que corresponde a las esquinas de la pizarra. Utilizando el archivo *haarcascade.xml* obtenido en el entrenamiento de la Cascada de Haar para detectar la letra X, se puede programar en EmguCV de la siguiente manera:

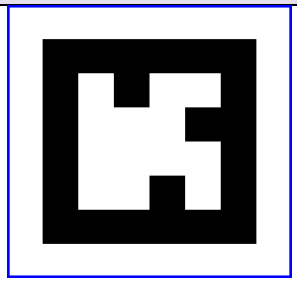
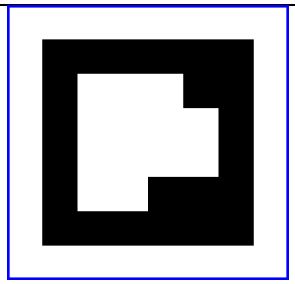
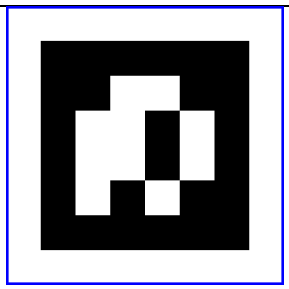
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.CV.GPU;

namespace EquisDetection
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Run();
      }

      static void Run()
      {
          Image<Bgr, Byte> image = new Image<Bgr, byte>("equis.jpg"); //Read
the files as an 8-bit Bgr image
         long detectionTime;
         List<Rectangle> equis = new List<Rectangle>();
         DetectFace.Detect(image,      "haarcascade.xml",      equis,      out
detectionTime);
         foreach (Rectangle esquina in equis)
            image.Draw(esquina, new Bgr(Color.Red), 2);

         //display the image
         ImageViewer.Show(image, String.Format(
```

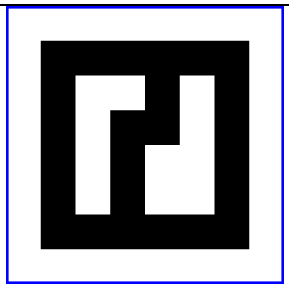
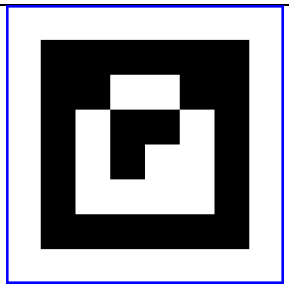
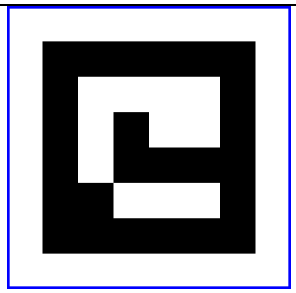
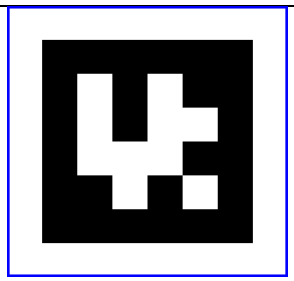
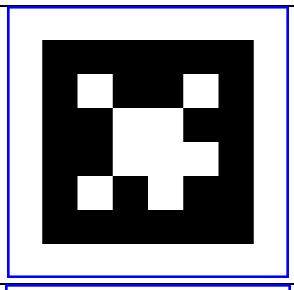
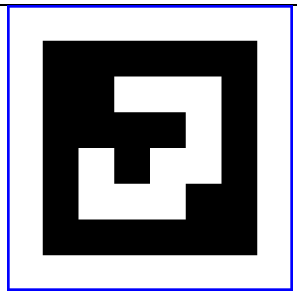
```
                "Completed equis detection using {0} in {1} milliseconds",
                GpuInvoke.HasCuda ? "GPU": "CPU",
                detectionTime));
        }

    }
}
```

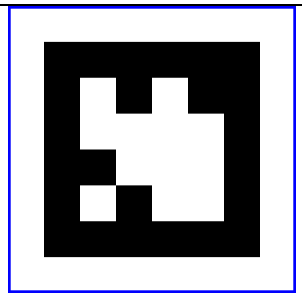
4.8.4 Detección de marcadores de Realidad Aumentada

La detección de marcadores es un tema que tiene aplicaciones en una amplia gama de diferentes áreas. La aplicación más popular, es la realidad aumentada, donde un algoritmo de visión por computador los encuentra en una secuencia de vídeo, y es sustituido con objetos generados artificialmente. Otra área de aplicación de marcadores ópticos es la robótica, donde los marcadores se pueden utilizar para dar órdenes a un robot, a su vez ayuda para que el robot navegue dentro de un entorno. La Tabla 4.6 muestra los marcadores utilizados en la funcionalidad del sistema PIVRA.

Tabla 4.6: Asignación de letra y arte de los marcadores del sistema PIVRA

Arte del marcador	Letra	Modelo Asignado
	C	Cerebro
	Z	Corazón
	E	Estomago

	H	Hígado
	I	Intestinos
	P	Pulmones
	R	Riñones
	O	Ojo
	N	Cráneo

	M	Músculo

Elaborado por. Jimena Caguana

Los algoritmos de procesamiento de imágenes enfocados a la detección de marcadores de realidad aumentada, se basan en el framework AForge.NET. Para la creación de algoritmos se utilizó la aplicación IPPrototyper, que es parte del framework. **[34]**

La inducción en la detección y reconocimiento de marcadores ópticos propone encontrar todas las áreas del cuadrilátero; es decir, se tiene que encontrar cuatro esquinas de cada marcador en la imagen de origen. El primer paso es trivial, se hace grayscaling de la imagen original, ya que se reduce la cantidad de datos a procesar; además de que no es necesaria la información de color para realizar esta tarea. **[34]**

Los marcadores ópticos son objetos de bastante contraste, un marcador con borde negro sobre papel blanco. Como se puede apreciar en la Fig. 4.61. Una de las ideas para detectar marcadores es hacer un umbral de la imagen y luego análisis de blob para encontrar cuadriláteros negros. Por supuesto que no se va a usar umbralización regular con umbral predefinido, ya que no se efectuará nada, porque no se puede fijar un valor umbral para todas las posibles condiciones de luz y medio ambiente. Al utilizar el umbral Otsu puede producir buenos resultados como se muestra en Fig. 4.62. **[34]**

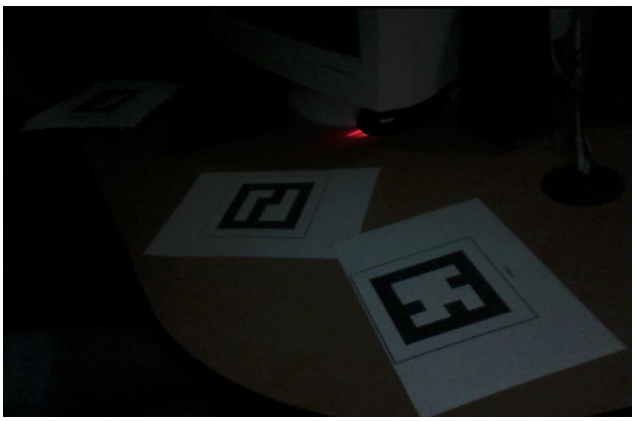


Fig. 4.61: Ejemplo1 marcadores a detección en imagen

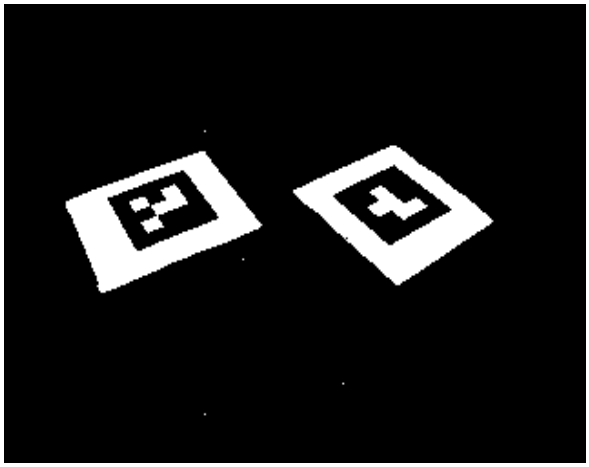


Fig. 4.62: Ejemplo umbral Otsu

La Fig. 4.63 muestra un ejemplo dos de la detección de marcadores en video; la Fig. 4.64 muestra el ejemplo dos aplicado el umbral Otsu. **[34]**

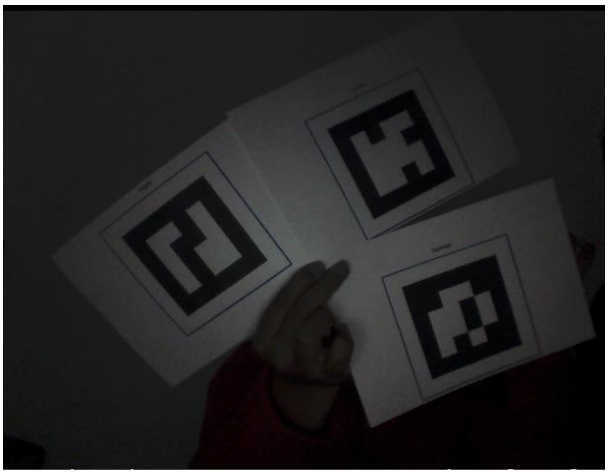


Fig. 4.63: Ejemplo2 marcadores a detección en video

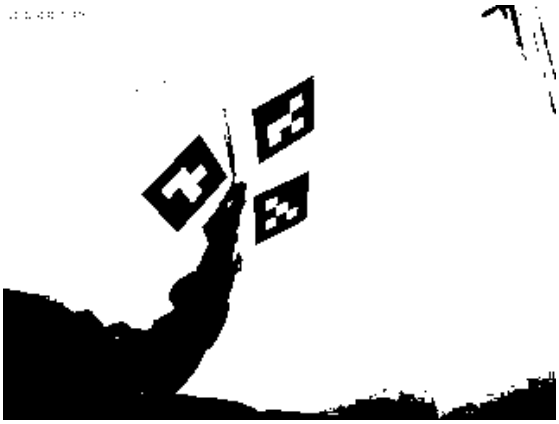


Fig. 4.64: Ejemplo2 con umbral Otsu

Como se observó en las imágenes, el umbral Otsu hizo su trabajo bastante bien, se consiguió cuadriláteros negros rodeados de áreas blancas. Al utilizar un contador blob se pueden encontrar todos los objetos negros en las imágenes binarias anteriores, o realizar algunas comprobaciones para asegurarse de que estos objetos son cuadriláteros, etc. El problema es que el umbral Otsu trabajó para las imágenes de arriba y que en realidad funciona para muchas otras imágenes. Pero no para otras imágenes como se muestra en la Fig. 4.66, en las que no funciona como se supone. La Fig. 4.65 muestra un ejemplo tres de la detección de marcadores en video y la Fig. 4.66 el umbral Otsu fallido en el ejemplo tres. **[34]**

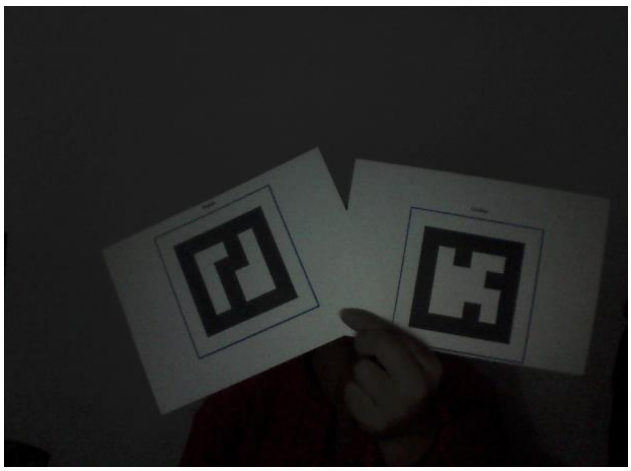


Fig. 4.65: Ejemplo3 marcadores a detección en video

Fig. 4.66: Umbral Otsu fallido

La imagen de arriba muestra que la umbralización global no funciona muy bien para ciertas condiciones de iluminación. Así que se tiene que encontrar otra solución para la detección de marcadores. **[34]**

Como ya se mencionó, los marcadores ópticos son objetos de bastante contraste, imágenes en color negro rodeado de un área blanca. Por supuesto, el contraste puede cambiar dependiendo de las condiciones de luz; es decir, las zonas negras pueden conseguirse más brillantes, pero las áreas blancas pueden conseguirse más oscuras. Pero aún así la diferencia debe ser suficiente, a menos que se tenga absolutamente mala iluminación. Así que en lugar de tratar de encontrar cuadriláteros negros o blancos, se puede tratar de encontrar regiones en las que el brillo de la imagen cambia bruscamente. Esta es la tarea de un detector de bordes, por ejemplo Difference Edge Detector (Diferencia del Detector de Bordes). La Fig. 4.67 muestra la aplicación de detección de bordes en el ejemplo tres. **[34]**

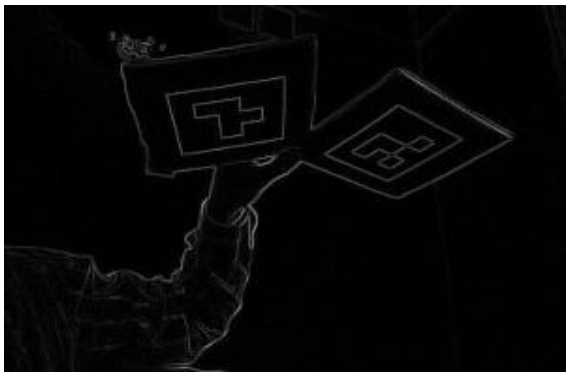


Fig. 4.67: Detector de bordes Ejemplo3

Para deshacerse de las áreas donde los cambios de brillo de la imagen son de manera insignificante, se hace umbralización. La Fig. 4.68 muestra los resultados en el ejemplo uno de la aplicación simultanea del umbral Otsu y la detección de bordes. **[34]**

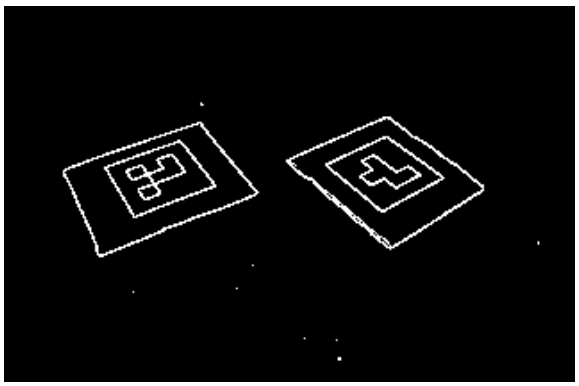


Fig. 4.68: Umbralización Otsu y detección de bordes Ejemplo1

Las Fig. 4.69 y 4.70 muestran los resultados al aplicar el umbral Otsu con la detección de bordes en los ejemplos dos y tres respectivamente. **[34]**

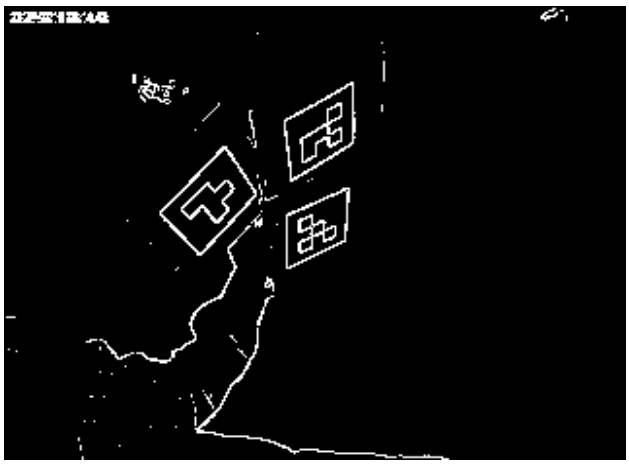


Fig. 4.69: Umbralización Otsu y detección de bordes Ejemplo2

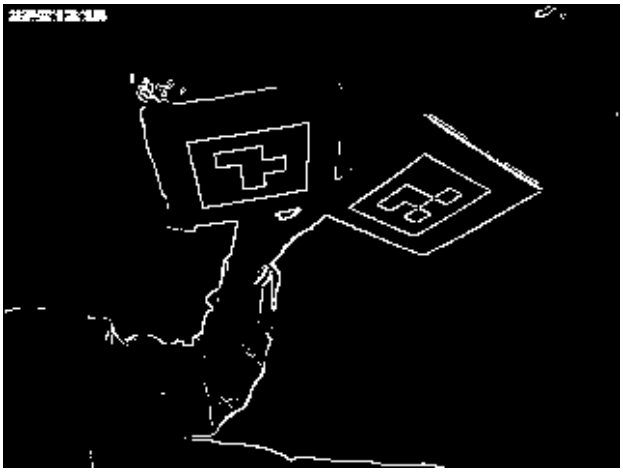


Fig. 4.70: Umbralización Otsu y detección de bordes Ejemplo3

Como se puede ver en las imágenes, todos los marcadores detectados están representados por un cuadrilátero formando un bloque independiente. En el caso si la condición de iluminación no es del todo mala, los cuadriláteros de los marcadores tienen un borde bien definido, por lo que realmente están representados con un solo bloque, que será fácil de extraer con unos algoritmos de conteo blob. **[34]**

A continuación se muestra un ejemplo de las malas condiciones de iluminación, donde tanto el umbral Otsu y la detección de bordes de umbral no producen ningún buen resultado que se podría utilizar para su posterior detección y reconocimiento de

marcadores. Las figuras Fig. 4.71, Fig. 4.72 y Fig. 4.73 muestran el procesamiento de imágenes fallido en el ejemplo cuatro al aplicar el umbral Otsu y la detección de bordes.

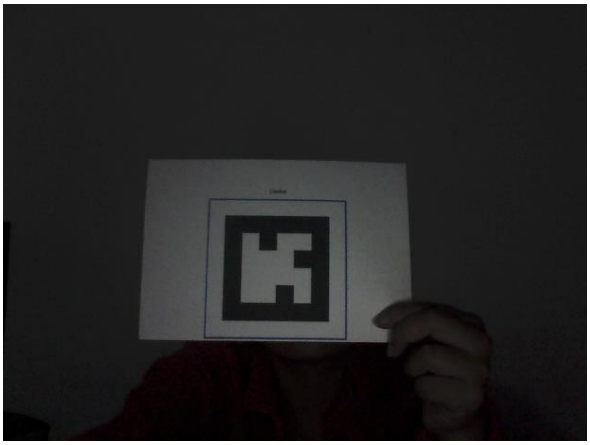


Fig. 4.71: Ejemplo4 detección de marcadores en video

Fig. 4.72: Umbralización Otsu en el Ejemplo4

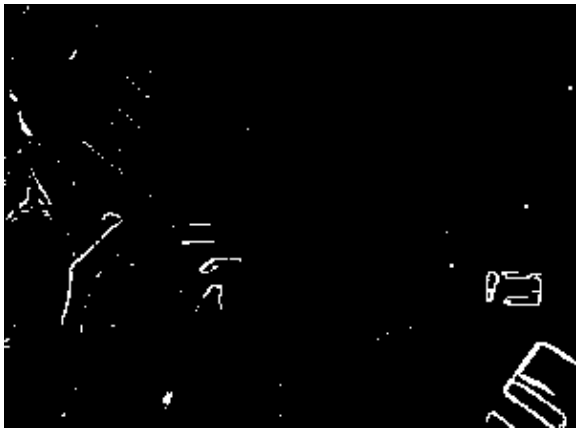


Fig. 4.73: Umbralización Otsu y detección de bordes Ejemplo4

Así que la mejor decisión es la detección de bordes, después de aplicar el umbral Otsu. A partir de esta premisa se inicia con el código de programación (se usará UnmanagedImage para evitar bloqueos supletorios / desbloqueos de imagen administrada de .NET):

```
// 1 - grayscaling
UnmanagedImage grayImage = null;
if ( image.PixelFormat == PixelFormat.Format8bppIndexed )
{
    grayImage = image;
}
else
{
    grayImage = UnmanagedImage.Create( image.Width, image.Height,
        PixelFormat.Format8bppIndexed );
    Grayscale.CommonAlgorithms.BT709.Apply( image, grayImage );
}
// 2 - Edge detection
DifferenceEdgeDetector edgeDetector = new DifferenceEdgeDetector( );
UnmanagedImage edgesImage = edgeDetector.Apply( grayImage );
// 3 - Threshold edges
Threshold thresholdFilter = new Threshold( 40 );
thresholdFilter.ApplyInPlace( edgesImage );
```

Ahora, cuando se tiene una imagen binaria que contiene bordes significativos de todos los objetos, se tiene que procesar todos los bloques formados por estos bordes y comprobar si alguno de los blobs puede representar una ventaja de un marcador. Para ir a través de todas los blobs de manera separada, se puede utilizar BlobCounter:

```
// create and configure blob counter
BlobCounter blobCounter = new BlobCounter( );
blobCounter.MinHeight    = 32;
blobCounter.MinWidth     = 32;
blobCounter.FilterBlobs  = true;
blobCounter.ObjectsOrder = ObjectsOrder.Size;
// 4 - find all stand alone blobs
blobCounter.ProcessImage( edgesImage );
Blob[] blobs = blobCounter.GetObjectsInformation( );
// 5 - check each blob
for ( int i = 0, n = blobs.Length; i < n; i++ )
{
    // ...
}
```

Como se puede ver en las imágenes de bordes binarios, se tiene un montón de aristas. Pero no todas ellas forman un objeto cuadrilátero. El interés es únicamente en los blobs que buscan cuadriláteros, porque un marcador será siempre representado por un cuadrilátero con independencia de la forma en que se gira. Para hacer un chequeo de

cuadrilátero, se puede recoger puntos de borde de blob utilizando *GetBlobsEdgePoints* ( ) y luego utilizar el método *IsQuadrilateral* ( ) para comprobar si estos puntos pueden formar un cuadrilátero. Si no, entonces se salta el blob y se procede a continuación:

```
List<IntPoint> edgePoints = blobCounter.GetBlobsEdgePoints( blobs[i] );
List<IntPoint> corners = null;
// does it look like a quadrilateral ?
if ( shapeChecker.IsQuadrilateral( edgePoints, out corners ) )
{
    // ...
}
```

Bien, ahora se tiene todos los blobs que se parecen a los cuadriláteros. Sin embargo no todo cuadrilátero es un marcador. Como ya se ha mencionado, un marcador tiene un borde negro y se imprime sobre papel blanco. Así que se tiene que hacer una comprobación de que una marca que se tiene, es el interior negro, pero fuera blanco; es decir, debería ser mucho más oscuro en el interior que en el exterior (ya que la iluminación puede variar y la comprobación de perfecto blanco / negro no funcionará).

Para realizar una comprobación si el blob es más oscuro en el interior que en el exterior, es posible a través de los puntos del borde derecho del blob usando el método *GetBlobsLeftAndRightEdges* ( ) y luego calcular la diferencia media de brillo entre píxeles en las afueras del blob y su interior. Si la diferencia promedio es lo suficientemente significativa, lo más probable es que se tiene un objeto oscuro rodeado de un área más clara. **[34]**

```
// get edge points on the left and on the right side
List<IntPoint> leftEdgePoints, rightEdgePoints;
blobCounter.GetBlobsLeftAndRightEdges( blobs[i],
    out leftEdgePoints, out rightEdgePoints );

// calculate average difference between pixel values from outside of the
// shape and from inside
float diff = CalculateAverageEdgesBrightnessDifference(
    leftEdgePoints, rightEdgePoints, grayImage );

// check average difference, which tells how much outside is lighter than
// inside on the average
if ( diff > 20 )
{
    // ...
}
```

Para aclarar la idea de calcular la diferencia media entre los píxeles fuera y dentro de un blob, con el método *CalculateAverageEdgesBrightnessDifference* ( ). Tanto para los bordes izquierdo y derecho, el método construye dos listas de puntos (lista de los puntos que son un poco a la izquierda desde el borde y la lista de puntos que son un poco a la derecha desde el borde) (3 píxeles de distancia desde el borde). Para cada una de las listas de puntos que recoge los valores de píxel correspondientes a estos puntos usando el método *Collect8bppPixelValues* ( ). Entonces se calcula la diferencia media, para el borde del blob izquierdo; resta el valor del píxel en el lado derecho del borde (en el interior del blob) del valor del píxel en el lado izquierdo del borde (fuera del blob); por la orilla del blob derecho hace una diferencia contraria. Cuando se realiza el método de cálculo produce un valor, que es una diferencia promedio entre los píxeles fuera y dentro del blob. **[34]**

```
const int stepSize = 3;

// Calculate average brightness difference between pixels outside and
// inside of the object bounded by specified left and right edge
private float CalculateAverageEdgesBrightnessDifference(
    List<IntPoint> leftEdgePoints,
    List<IntPoint> rightEdgePoints,
    UnmanagedImage image )
{
    // create list of points, which are a bit on the left/right from edges
    List<IntPoint> leftEdgePoints1  = new List<IntPoint>( );
    List<IntPoint> leftEdgePoints2  = new List<IntPoint>( );
    List<IntPoint> rightEdgePoints1 = new List<IntPoint>( );
    List<IntPoint> rightEdgePoints2 = new List<IntPoint>( );

    int tx1, tx2, ty;
    int widthM1 = image.Width - 1;

    for ( int k = 0; k < leftEdgePoints.Count; k++ )
    {
        tx1 = leftEdgePoints[k].X - stepSize;
        tx2 = leftEdgePoints[k].X + stepSize;
        ty = leftEdgePoints[k].Y;

        leftEdgePoints1.Add( new IntPoint(
            ( tx1 < 0 ) ? 0 : tx1, ty ) );
        leftEdgePoints2.Add( new IntPoint(
            ( tx2 > widthM1 ) ? widthM1 : tx2, ty ) );

        tx1 = rightEdgePoints[k].X - stepSize;
        tx2 = rightEdgePoints[k].X + stepSize;
        ty = rightEdgePoints[k].Y;

        rightEdgePoints1.Add( new IntPoint(
            ( tx1 < 0 ) ? 0 : tx1, ty ) );
```

```
            rightEdgePoints2.Add( new IntPoint(
                ( tx2 > widthM1 ) ? widthM1 : tx2, ty ) );
        }

        // collect pixel values from specified points
        byte[] leftValues1  = image.Collect8bppPixelValues( leftEdgePoints1 );
        byte[] leftValues2  = image.Collect8bppPixelValues( leftEdgePoints2 );
        byte[] rightValues1 = image.Collect8bppPixelValues( rightEdgePoints1 );
        byte[] rightValues2 = image.Collect8bppPixelValues( rightEdgePoints2 );

        // calculate average difference between pixel values from outside of
        // the shape and from inside
        float diff = 0;
        int pixelCount = 0;

        for ( int k = 0; k <leftEdgePoints.Count; k++ )
        {
            if ( rightEdgePoints[k].X - leftEdgePoints[k].X > stepSize * 2 )
            {
                diff += ( leftValues1[k]  - leftValues2[k] );
                diff += ( rightValues2[k] - rightValues1[k] );
                pixelCount += 2;
            }
        }

        return diff / pixelCount;
    }
```

Ahora es el momento de revisar los resultados de los dos chequeos que se realizó, para si es cuadrilátero y para la diferencia media entre los píxeles dentro y fuera del blob. Subrayando los bordes de todas las marcas que pasan estos controles y ver si se puede estar más cerca de la detección del marcador. Las figuras Fig. 4.74, Fig. 4.75 y Fig. 4.76 muestran la detección del blob en el ejemplo uno, ejemplo dos y ejemplo tres respectivamente. **[34]**

Fig. 4.74: Detección del blob Ejemplo1

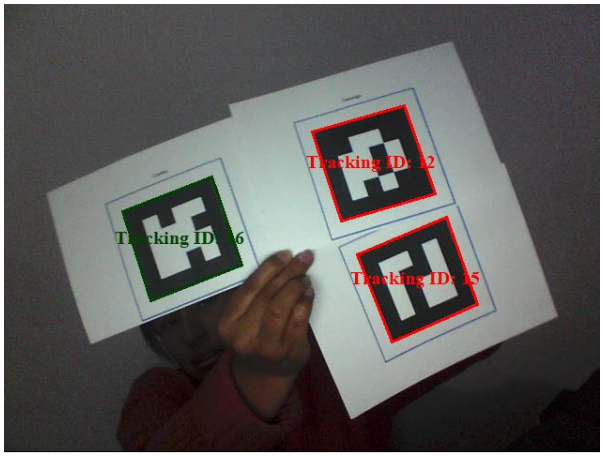


Fig. 4.75: Detección del blob Ejemplo2

Fig. 4.76: Detección del blob Ejemplo3

Se puede apreciar que el resultado de los dos chequeos que se realizó es realmente aceptable, únicamente los blobs que contienen los marcadores ópticos se destacaron y nada más. Potencialmente, puede ocurrir que algunos otros objetos pueden satisfacer dichos controles y el algoritmo puede encontrar algún otro cuadrilátero oscuro rodeado de áreas blancas. Sin embargo experimentos muestran que no sucede muy a menudo. Incluso si sucede a veces, todavía falta el proceso de reconocimiento de marcadores, que puede filtrar marcadores "falsos". **[34]**

4.8.5 Reconocimiento de marcadores de Realidad Aumentada

Ahora, se tiene las coordenadas de marcadores potenciales (sus cuadriláteros), se puede hacer su reconocimiento real. Es posible desarrollar un algoritmo, que hace el reconocimiento del marcador directamente en la imagen de origen. Sin embargo, se va a

simplificar un poco la tarea al tener una imagen cuadrada separada para cada marcador potencial, que contiene sólo datos del marcador. Esto se puede hacer usando el método *QuadrilateralTransformation* ( ). A continuación la Fig. 4.77 presenta los marcadores extraídos de algunas de las imágenes procesadas previamente. **[34]**

```
// 6 - do quadrilateral transformation
QuadrilateralTransformation quadrilateralTransformation =
    new QuadrilateralTransformation( quadrilateral, 100, 100 );
UnmanagedImage glyphImage = quadrilateralTransformation.Apply( image );
```

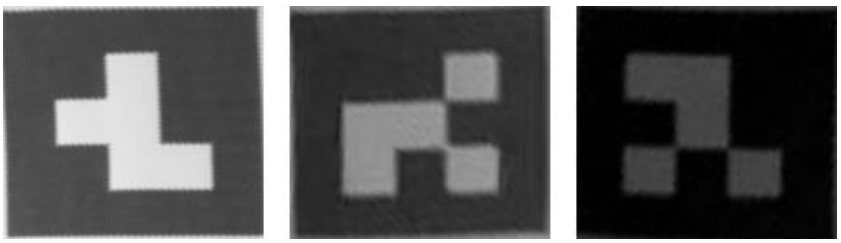


Fig. 4.77: Marcadores extraídos

Como se puede apreciar en las imágenes anteriores, las condiciones de iluminación pueden variar mucho y algunos marcadores no pueden estar tan contrastados como los demás. Así que se puede usar Otsu thresholing en esta etapa para binarizar los marcadores. **[34]**

```
// otsu thresholding
OtsuThreshold otsuThresholdFilter = new OtsuThreshold( );

otsuThresholdFilter.ApplyInPlace( glyphImage );
```

En esta etapa está todo listo para entrar en el reconocimiento final del marcador. Hay diferentes maneras para hacer esto, al igual que el reconocimiento de formas, comparación de plantillas, etc. Aunque puede haber beneficios de usar el reconocimiento de formas, es demasiado complejo para una simple tarea de reconocer un marcador; pero a su vez, satisface la tarea a partir de que el marcador representa un cuadrilátero. Como se mencionó antes, todos los marcadores están representados por una cuadrícula, donde cada célula está llena de color negro o blanco. Así, un algoritmo de reconocimiento puede hacerse con esta suposición; dividiendo la imagen del marcador en células y comprobar cuál es el promedio de color (más común) de la célula. **[34]**

Antes de entrar en el código de reconocimiento del marcador, se va a hacer algunas aclaraciones a la forma en que se divide el marcador en las células. Por ejemplo, en la Fig. 4.75 se puede ver cómo el marcador está dividido por líneas de color gris oscuro en la cuadrícula de células de 5x5 que tienen misma anchura y altura. Entonces, lo que se hace, es contar el número de píxeles blancos en cada una de esas células y verificar si el número es mayor que la mitad de la superficie de la célula. Si es mayor, entonces se supone que la célula se llena por el color blanco, permite decir que corresponde a "1". Y si el número es menos de la mitad de la superficie de la célula, entonces se tiene una célula negra llena, que corresponde a "0". También se puede introducir un nivel de confianza para cada celda; si toda la célula se llena de píxeles de color blanco o negro, entonces se está 100% seguro sobre el color-tipo de célula. Sin embargo, si una célula tiene 60% de los píxeles en blanco y 40% de píxeles negros, entonces la confianza de reconocimiento cae a 60%. Cuando una célula está media llena de blanco y media llena de color negro, a continuación, la confianza es igual a 50%, lo que significa que no se está seguro del todo sobre el color-tipo de células. La Fig. 4.78 muestra la cuadrícula sobrepuesta en el marcador detectado. **[34]**

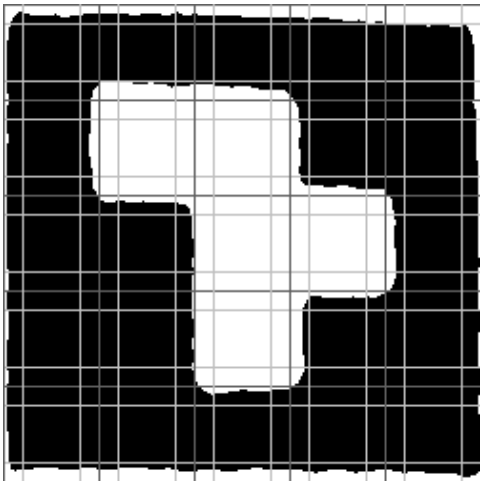


Fig. 4.78: Cuadrícula del marcador

Sin embargo, con el enfoque descrito anteriormente será apenas posible encontrar una célula, que pueda dar el 100% de nivel de confianza. Como se pudo ver en la imagen de arriba, todo el proceso de localización del marcador, extracción, umbral, etc. puede causar algunas imperfecciones; algunos bordes de células pueden contener también partes de las áreas blancas que rodean un marcador, pero algunas células internas que se

supone que son negro puede contener píxeles blancos causados por las células vecinas blancas, etc. Así que en lugar de calcular el número de píxeles blancos en el borde de toda la célula, se puede introducir un pequeño espacio alrededor de las fronteras de la célula y excluirla del procesamiento. La imagen de arriba muestra la idea con lagunas, en lugar de escanear toda la célula que se destaca por las líneas de color gris oscuro, se escanea la zona interior más pequeña que se destaca con líneas de color gris claro. **[34]**

Ahora, cuando la idea del reconocimiento parece estar clara, se puede llegar a su implementación. En primer lugar el código va a proporcionar la imagen y calcular la suma de los valores de píxeles para cada celda. A continuación, estas sumas se utilizan para calcular la plenitud de cada célula; lo lleno, es una célula llena de píxeles blancos. Finalmente la plenitud de la célula se utiliza para determinar su tipo ("1" - blanco lleno o "0" - negro lleno) y el nivel de confianza. **[34]**

NOTA: Antes de usar esta función (método), el usuario debe configurar el tamaño de marcador a reconocer. **[34]**

```
public byte[,] Recognize( UnmanagedImage image, Rectangle rect,
    out float confidence )
{
    int glyphStartX = rect.Left;
    int glyphStartY = rect.Top;

    int glyphWidth  = rect.Width;
    int glyphHeight = rect.Height;

    // glyph's cell size
    int cellWidth  = glyphWidth  / glyphSize;
    int cellHeight = glyphHeight / glyphSize;

    // allow some gap for each cell, which is not scanned
    int cellOffsetX = (int) ( cellWidth  * 0.2 );
    int cellOffsetY = (int) ( cellHeight * 0.2 );

    // cell's scan size
    int cellScanX = (int) ( cellWidth  * 0.6 );
    int cellScanY = (int) ( cellHeight * 0.6 );
    int cellScanArea = cellScanX * cellScanY;

    // summary intensity for each glyph's cell
    int[,] cellIntensity = new int[glyphSize, glyphSize];

    unsafe
    {
        int stride = image.Stride;
```

```
            byte* srcBase = (byte*) image.ImageData.ToPointer( ) +
                ( glyphStartY + cellOffsetY ) * stride +
                glyphStartX + cellOffsetX;
            byte* srcLine;
            byte* src;

            // for all glyph's rows
            for ( int gi = 0; gi < glyphSize; gi++ )
            {
                srcLine = srcBase + cellHeight * gi * stride;

                // for all lines in the row
                for ( int y = 0; y < cellScanY; y++ )
                {
                    // for all glyph columns
                    for ( int gj = 0; gj < glyphSize; gj++ )
                    {
                        src = srcLine + cellWidth * gj;

                        // for all pixels in the column
                        for ( int x = 0; x < cellScanX; x++, src++ )
                        {
                            cellIntensity[gi, gj] += *src;
                        }
                    }

                    srcLine += stride;
                }
            }
        }

        // calculate value of each glyph's cell and set
        // glyphs' confidence to minim value of cell's confidence
        byte[,] glyphValues = new byte[glyphSize, glyphSize];
        confidence = 1f;

        for ( int gi = 0; gi < glyphSize; gi++ )
        {
            for ( int gj = 0; gj < glyphSize; gj++ )
            {
                float fullness = (float)
                    ( cellIntensity[gi, gj] / 255 ) / cellScanArea;
                float conf = (float) System.Math.Abs( fullness - 0.5 ) + 0.5f;

                glyphValues[gi, gj] = (byte) ( ( fullness > 0.5f ) ? 1 : 0 );

                if ( conf < confidence )
                    confidence = conf;
            }
        }

        return glyphValues;
    }
```

Con la función de lo previsto anteriormente, el siguiente paso después de binarización del marcador quedaría:

```
// recognize raw glyph
float confidence;

byte[,] glyphValues = binaryGlyphRecognizer.Recognize( glyphImage,

    new Rectangle( 0, 0, glyphImage.Width, glyphImage.Height ), out
confidence );
```

En esta etapa se tiene una matriz de bytes 2D que contiene elementos "0" y "1" correspondientes a las células blanco y negro de la imagen de un marcador. Por ejemplo, la función debería proporcionar como resultado para la imagen del marcador de arriba, la siguiente matriz:

```
0 0 0 0 0
0 1 1 0 0
0 0 1 1 0
0 0 1 0 0
0 0 0 0 0
```

Ahora, hay que hacer algunas comprobaciones para asegurar que se procesó una imagen marcador sin limitaciones como se propuso al principio. En primer lugar, se va a ver el nivel de confianza, si es inferior a cierto límite (por ejemplo: 0.6 que corresponde al 60%), a continuación, se salta al objeto procesado. También se salta, en el caso si el marcador no tiene una frontera hecha de células negras (si los datos del marcador contienen al menos un único valor "1" en la primera-última fila o columna) o si no tiene al menos un blanco en la célula, en cualquier fila o columna interna. **[34]**

```
if ( confidence >= minConfidenceLevel )
{
    if ( ( CheckIfGlyphHasBorder( glyphValues ) ) &&
         ( CheckIfEveryRowColumnHasValue( glyphValues ) ) )
    {
        // ...
        // further processing
    }
}
```

Eso es todo acerca de los datos en la extracción-reconocimiento de marcadores. Si una imagen contiene un marcador potencial y ha pasado todos estos pasos y chequeos, entonces puede ser reconocido con precisión. **[34]**

Marcador encontrado en una base de datos de varios marcadores: Aunque se hizo la extracción de los datos de un marcador en una imagen, este no es el último paso en la tarea de reconocimiento de marcadores. Aplicaciones que se ocupan de la realidad

aumentada o la robótica, suelen tener una base de datos de los marcadores, donde cada marcador puede tener su propio significado. Por ejemplo, en la realidad aumentada cada marcador está asociado con un objeto virtual que se muestra en lugar de ese marcador, pero en aplicaciones de robótica cada marcador puede representar un comando o dirección para un robot. Así que el último paso es el reconocimiento de los datos extraídos de un marcador con una base de datos de varios marcadores, y recuperar la información relacionada con dicho marcador; es ID, nombre, etc. **[34]**

Para completar el paso de reconocimiento entre un grupo de marcadores con éxito, se debe tener en cuenta que los marcadores se pueden girar, por lo que la comparación de los datos extraídos uno a uno del marcador con marcadores almacenados en la base de datos, no funcionará. Para encontrar un marcador correspondiente en la base de datos de marcadores, se tiene que hacer cuatro comparaciones en los datos extraídos del marcador con cada marcador en la base de datos; además, comparar cuatro posibles rotaciones de los datos extraídos del marcador con la base de datos. **[34]**

Otra cosa importante es que todos los marcadores en la base de datos deben ser variantes de rotación con el fin de ser únicos e independientes a su rotación. Si un marcador puede parecer el mismo después de la rotación, entonces es un marcador invariante en rotación. Para la rotación de los marcadores invariantes no se puede determinar su ángulo de rotación, que es muy importante para aplicaciones como la realidad aumentada. También puede que no sea posible encontrar el marcador correcto en una base de datos, si contiene marcadores invariantes de rotación, que pueden parecer el mismo si se hace girar uno de ellos. **[34]**

A continuación la Fig. 4.79 muestra marcadores invariantes en rotación y variantes de rotación. Marcadores (1) y (2) son variantes de rotación (si se giran, se mirarán siempre diferentes). Marcadores (3), (4) y (5) son invariantes de rotación (si rotan, se verá la mismo, por lo que no es posible detectar su ángulo de rotación). También se puede ver que el marcador (4) es en realidad igual al marcador (5), pero sólo en rotación; por lo que la base de datos de marcadores no debe contener a los dos. **[34]**

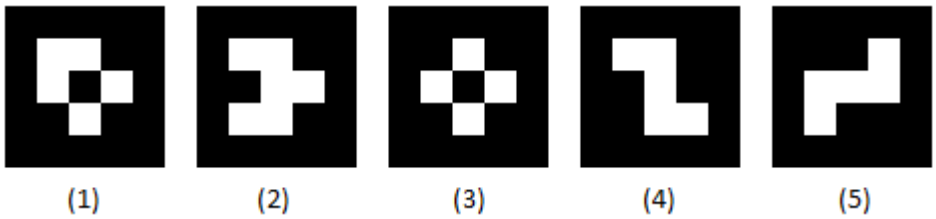


Fig. 4.79: Marcadores variante e invariantes de rotación

```
public int CheckForMatching( byte[,] rawGlyphData )
{
    int size = rawGlyphData.GetLength( 0 );
    int sizeM1 = size - 1;

    bool match1 = true;
    bool match2 = true;
    bool match3 = true;
    bool match4 = true;

    for ( int i = 0; i < size; i++ )
    {
        for ( int j = 0; j < size; j++ )
        {
            byte value = rawGlyphData[i, j];

            // no rotation
            match1 &= ( value == data[i, j] );
            // 180 deg
            match2 &= ( value == data[sizeM1 - i, sizeM1 - j] );
            // 90 deg
            match3 &= ( value == data[sizeM1 - j, i] );
            // 270 deg
            match4 &= ( value == data[j, sizeM1 - i] );
        }
    }

    if ( match1 )
        return 0;
    else if ( match2 )
        return 180;
    else if ( match3 )
        return 90;
    else if ( match4 )
        return 270;

    return -1;
}
```

Como se puede ver en el código anterior, el método devuelve -1 si los datos proporcionados por el marcador no se ajustan a los datos guardados en la variable de datos (miembro de una clase marcador). Sin embargo, si se encuentra en la base de datos de marcadores devuelve el ángulo de rotación (0, 90, 180 o 270 grados en sentido

contrario a las agujas del reloj) que se utilizan para obtener los datos de marcadores especificados desde el marcador original. **[34]**

Ahora, todo lo que se tiene que hacer es ir a través de todos los marcadores en una base de datos y comprobar si los datos del marcador que se ha extraído de la imagen coinciden con cualquiera de los marcadores de la base de datos. Si no se encuentra coincidencia, entonces se puede obtener todos los datos asociados con el marcador emparejado y utilizarlo para la visualización, dando orden para robots, etc. **[34]**

4.8.6 Detección y reconocimiento de letras

Al igual que en el apartado *4.8.3*, se utiliza el código de programación para detección de la letra X; mismo que es útil en todas las letras empleadas en el sistema PIVRA en la Tabla 4.6 del apartado *4.8.4*. A continuación se detalla las sentencias de código de programación para detección de la letra C, al ser redundante con la detección de las demás letras:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.CV.GPU;

namespace letterCDetection
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Run();
      }

      static void Run()
      {
          Image<Bgr, Byte> image = new Image<Bgr, byte>("letterC.jpg");
//Read the files as an 8-bit Bgr image
```

```
            long detectionTime;
            List<Rectangle> letterC = new List<Rectangle>();
            DetectFace.Detect(image, "haarcascade.xml", letterC, out
detectionTime);
            foreach (Rectangle letra in letterC)
               image.Draw(letra, new Bgr(Color.Red), 2);

            //display the image
            ImageViewer.Show(image, String.Format(
               "Completed letterC detection using {0} in {1} milliseconds",
               GpuInvoke.HasCuda ? "GPU": "CPU",
               detectionTime));
         }

      }
}
```

4.8.7 Proyección del modelo 2D o modelo 3D

Una vez, reconocido el marcador óptico, es el momento de probar algunos modelos 2D de realidad aumentada. Lo primero que se tiene que hacer es corregir el cuadrilátero del marcador (que se obtuvo desde *IsQuadrilateral* ( ) para convocar a la fase de localización del marcador). Como ya se mencionó el marcador que se extrajo del cuadrilátero encontrado, puede no ser exactamente el mismo que en la base de datos de marcadores, pero puede ser rotado. Así que se tiene que girar el cuadrilátero de tal manera que un marcador extraído se vea exactamente igual que en la base de datos. Para ello tenemos que utilizar el ángulo de rotación proporcionado por el método *CheckForMatching* ( ) realizado en fase de adaptación del marcador:

```
if ( rotation != -1 )
{
    foundGlyph.RecognizedQuadrilateral = foundGlyph.Quadrilateral;

    // rotate quadrilateral's corners
    while ( rotation > 0 )
    {
        foundGlyph.RecognizedQuadrilateral.Add(
foundGlyph.RecognizedQuadrilateral[0] );
        foundGlyph.RecognizedQuadrilateral.RemoveAt( 0 );

        rotation -= 90;
    }
}
```

Todo lo que se necesita hacer ahora para completar la realidad aumentada 2D es poner una imagen que se desee en el cuadrilátero corregido. Para ello se utiliza

BackwardQuadrilateralTransformation; igual que QuadrilateralTransformation, pero en vez de extraer la imagen del cuadrilátero especificado se pone otra imagen en ella. **[34]**

```
// put glyph's image onto the glyph using quadrilateral transformation
BackwardQuadrilateralTransformation quadrilateralTransformation =
    new BackwardQuadrilateralTransformation( );

quadrilateralTransformation.SourceImage = glyphImage;
quadrilateralTransformation.DestinationQuadrilateral                    =
glyphData.RecognizedQuadrilateral;

quadrilateralTransformation.ApplyInPlace( sourceImage );
```

La Fig. 4.80 muestra los resultados de la Realidad Aumentada en 2D. **[34]**

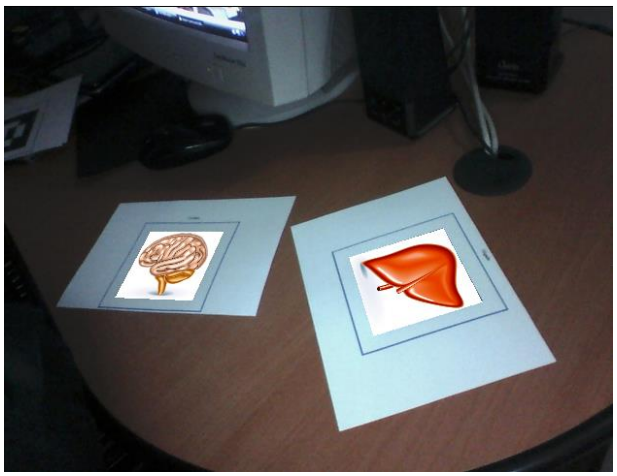


Fig. 4.80: Realidad Aumentada 2D

Estimación de pose: Como es obvio resulta que la realidad aumentada en 3D no es tan simple como la realidad aumentada 2D. Para colocar un objeto 3D en la parte superior de un marcador, no es suficiente conocer las coordenadas de las cuatro esquinas del marcador. En su lugar, se requiere conocer las coordenadas 3D del centro del marcador en el mundo real y sus ángulos de rotación alrededor de los ejes X -Y- Z. Así que antes de ir más lejos en la realidad aumentada 3D, se tiene que encontrar la manera de cómo determinar la pose 3D del marcador en el mundo real. **[34]**

Hay una serie de artículos de investigación publicados sobre pose de estimación 3D, describiendo diferentes algoritmos. El más popular de ellos es el algoritmo POSIT, que es bastante fácil de seguir y aplicar. El algoritmo se describe en "Objeto basado en modelos POSIT en 25 líneas de código" del paper de Daniel F. DeMenthon y Larry S. Davis. **[34]**

El propósito del algoritmo de POSIT es estimar la pose 3D de un objeto, que incluye la rotación sobre ejes X -Y - Z y el desplazamiento a lo largo de los ejes X -Y - Z. Para ello, el algoritmo requiere coordenadas de la imagen de los puntos de algún objeto (mínimo 4 puntos, exactamente el número de esquinas que se tiene). Entonces se necesita saber las coordenadas modelo de estos puntos. Con ellos se sabe las coordenadas de los puntos correspondientes en el modelo. Y, finalmente, el algoritmo requiere longitud focal efectiva de la cámara utilizada para representar el objeto. **[34]**

Se puede recopilar fácilmente toda la información requerida para el algoritmo POSIT para hacer la tarea de pose del modelo 3D. Sin embargo, el algoritmo tiene una limitación que hace que sea un poco inútil para proyección de realidad aumentada; el algoritmo está diseñado para el caso de valores no coplanares. En otras palabras, los puntos que se utilizan para la estimación de pose de los modelos, no pueden estar todos en el mismo plano. Lamentablemente este es el caso exacto que se tiene. Desde marcadores planares, que hace que sea imposible estimar su pose con POSIT. **[34]**

Por suerte los investigadores no se detuvieron en POSIT y acertaron con el algoritmo de extensión; que es coplanar POSIT. Esencialmente la misma POSIT pero para el caso coplanar. La descripción del algoritmo se puede encontrar en el documento "Iterativo Estimación Pose utilizando característica de puntos coplanar " documento escrito por Oberkampf, Daniel F. DeMenthon y Larry S. Davis. En cuanto a la aplicación que se va utilizar, es la clase CoplanarPOSIT del framework AForge.NET. **[34]**

En primer lugar, se inicia con las coordenadas de la imagen y los puntos que se va a utilizar para la estimación de pose. La imagen siguiente muestra cuatro puntos de colores en amarillo, azul, rojo y verde. Las coordenadas de los puntos son (todas las coordenadas son en relación con el centro de la imagen; dirección positiva del eje Y, es de centro a arriba; tamaño original de la imagen es de 640x480):

```
1.- (-77, 48) - yellow;
2.- (44, 66) - blue;
3.- (75, -36) - red;
4.- (-61, -58) – green.
```

La Fig. 4.81 muestra la estimación de pose del marcador con el algoritmo coplanar POSIT. **[34]**

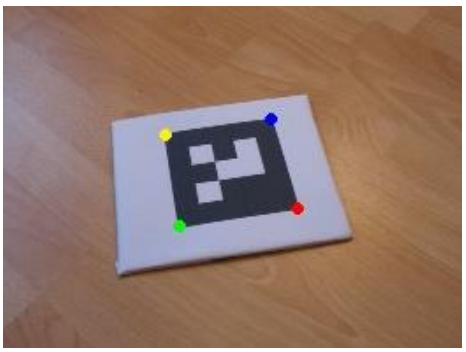


Fig. 4.81: Estimación de pose del marcador con el algoritmo coplanar POSIT
Fuente. http://www.codeproject.com/graphics/glyph.jpg - [34].

Se tiene que obtener las coordenadas del modelo de estos puntos. Suponiendo que se tiene que coordinar el centro del sistema PIVRA justo en el centro del marcador, el marcador se encuentra en el plano XZ , la ley de la mano derecha hace que el sistema de coordenadas con el eje Z va lejos del espectador, cuando X e Y van bien y se corresponden. Así que si el tamaño verdadero del marcador es de 113 mm, por ejemplo, entonces su definición del modelo debería ser algo como esto:

```
1.- (-56,5, 0, 56,5) - amarillo;
2.- (56,5, 0, 56,5) - azul;
3.- (56,5, 0, -56.5) - rojo;
4.- (-56,5, 0, -56.5) - verde.
```

La última tarea que se necesita, es la longitud focal efectiva. El ancho de la imagen puede ser tomado como una buena aproximación de la misma. Dado que el tamaño de la imagen de la fuente del ejemplo es de 640x480, se torna la distancia focal efectiva igual a 640. A partir de esto, se está listo para estimar la pose del marcador usando el siguiente código. **[34]**

```
// define model of glyph with side length equal to 113 mm
Vector3[] modelPoints = new Vector3[]
{
    new Vector3( -56.5f, 0,  56.5f ),
    new Vector3(  56.5f, 0,  56.5f ),
    new Vector3(  56.5f, 0, -56.5f ),
    new Vector3( -56.5f, 0, -56.5f ),
};

// define image points
AForge.Point[] imagePoints = new AForge.Point[]
```

```
{
    new AForge.Point( -77,  48 ),
    new AForge.Point(  44,  66 ),
    new AForge.Point(  75, -36 ),
    new AForge.Point( -61, -58 ),
};

// create instance of pose estimation algorithm
CoplanarPosit coposit = new CoplanarPosit( modelPoints, 640 );

// estimate pose of the object
Matrix3x3 rotationMatrix;
Vector3 translationVector;

coposit.EstimatePose( imagePoints, out rotationMatrix, out translationVector
);
```

Dado que el tema de este trabajo no cubre matrices de transformación 3D, proyección en perspectiva, etc., no se va a entrar en detalles sobre cómo interpretar la matriz de transformación calculada. En su lugar se tiene el código, que utiliza la rotación de la matriz y el vector de transformación obtenido, se pondrá en los ejes X -Y - Z en la parte superior del marcador para ver exactamente como se plantea la estimación 3D:

```
// model used to draw coordinate system's axes

private Vector3[] axesModel = new Vector3[]
{
    new Vector3( 0, 0, 0 ),
    new Vector3( 1, 0, 0 ),
    new Vector3( 0, 1, 0 ),
    new Vector3( 0, 0, 1 ),
};

// transform the model and perform perspective projection
AForge.Point[] projectedAxes = PerformProjection( axesModel,
    // create tranformation matrix
    Matrix4x4.CreateTranslation( translationVector ) *        // 3: translate
    Matrix4x4.CreateFromRotation( rotationMatrix ) *          // 2: rotate
    Matrix4x4.CreateDiagonal( new Vector4( 56, 56, 56, 1 ) ), // 1: scale
    imageSize.Width );

...

private AForge.Point[] PerformProjection( Vector3[] model,
                        Matrix4x4 transformationMatrix, int viewSize )
{
    AForge.Point[] projectedPoints = new AForge.Point[model.Length];

    for ( int i = 0; i < model.Length; i++ )
    {
        Vector3 scenePoint = ( transformationMatrix *
            model[i].ToVector4( ) ).ToVector3( );
```

```
            projectedPoints[i] = new AForge.Point(
                (int) ( scenePoint.X / scenePoint.Z * viewSize ),
                (int) ( scenePoint.Y / scenePoint.Z * viewSize ) );
        }

        return projectedPoints;

    }
```

Una vez proyectado los puntos del modelo 3D, sólo se tiene que dibujarlo:

```
// cx and cy are coordinates of image's centre

using ( Pen pen = new Pen( Color.Blue, 5 ) )
{
    g.DrawLine( pen,
        cx + projectedAxes[0].X, cy - projectedAxes[0].Y,
        cx + projectedAxes[1].X, cy - projectedAxes[1].Y );
}

using ( Pen pen = new Pen( Color.Red, 5 ) )
{
    g.DrawLine( pen,
        cx + projectedAxes[0].X, cy - projectedAxes[0].Y,
        cx + projectedAxes[2].X, cy - projectedAxes[2].Y );
}

using ( Pen pen = new Pen( Color.Lime, 5 ) )
{
    g.DrawLine( pen,
        cx + projectedAxes[0].X, cy - projectedAxes[0].Y,
        cx + projectedAxes[3].X, cy - projectedAxes[3].Y );

}
```

La Fig. 4.82 muestra la estimación y dibujo del plano 3D con el algoritmo coplanar POSIT. **[34]**

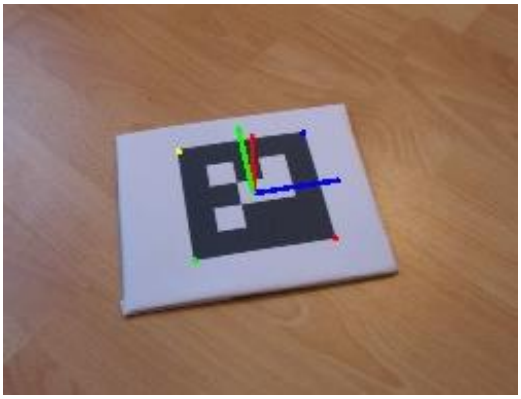


Fig. 4.82: Estimación y dibujo del plano 3D con coplanar POSIT
Fuente. http://www.codeproject.com/graphics/pose_coposit.jpg - [34].

La única diferencia para el usuario entre los algoritmos POSIT y coplanar POSIT, es el hecho que el algoritmo coplanar POSIT ofrece dos estimaciones de la pose del objeto; hay dos soluciones de sistema de ecuaciones para la versión coplanar del algoritmo. La única manera de comprobar que se plantee la mejor estimación, es aplicar ambas transformaciones que estima el modelo, para llevar a cabo la proyección de perspectiva y comparar con puntos proporcionados de la imagen. La estimación de pose que conduce a puntos similares de la imagen se supone que es la mejor. NOTA: Todo esto se hace mediante la aplicación del algoritmo coplanar POSIT de forma automática, por lo que proporciona la mejor estimación. **[34]**

Realidad Aumentada 3D:

Ahora, que se tiene todos los bits necesarios, es el momento de poner a todos juntos con el fin de obtener realidad aumentada 3D, donde un objeto virtual 3D se pone encima de un marcador real impreso. **[34]**

Una de las primeras ideas para comenzar es el uso de librerías-framework, para la representación 3D. Para este proyecto de realidad aumentada, se decidió probar el framework XNA de Microsoft. **[34]**

El framework XNA está dirigido al desarrollo de juegos, su integración con aplicaciones Windows Forms no era algo sencillo desde su primer lanzamiento. La idea era que XNA gestionara la ventana completa del juego, los gráficos y la entrada- salida. Sin embargo las cosas han mejorado desde entonces y hay eficiencia de integración de XNA en aplicaciones Windows Forms. A raíz de algunos ejemplos y tutoriales de XNA, se utiliza un código simple para la prestación de un pequeño modelo 3D:

```
protected override void Draw( )
{
    GraphicsDevice.Clear( Color.Black );

    // draw simple models for now with single mesh
    if ( ( model != null ) && ( model.Meshes.Count == 1 ) )
    {
        ModelMesh mesh = model.Meshes[0];

        // spin the object according to how much time has passed
        float time = (float) timer.Elapsed.TotalSeconds;

        // object's rotation and transformation matrices
```

```
        Matrix rotation = Matrix.CreateFromYawPitchRoll(
            time * 0.5f, time * 0.6f, time * 0.7f );
        Matrix translation = Matrix.CreateTranslation( 0, 0, 0 );

        // create transform matrices
        Matrix viewMatrix = Matrix.CreateLookAt(
            new Vector3( 0, 0, 3 ), Vector3.Zero, Vector3.Up );
        Matrix projectionMatrix = Matrix.CreatePerspective(
            1, 1 / GraphicsDevice.Viewport.AspectRatio, 1f, 10000 );
        Matrix world = Matrix.CreateScale( 1 / mesh.BoundingSphere.Radius ) *
            rotation * translation;

        foreach ( Effect effect in mesh.Effects )
        {
            if ( effect is BasicEffect )
            {
                ( (BasicEffect) effect ).EnableDefaultLighting( );
            }

            effect.Parameters["World"].SetValue( world );
            effect.Parameters["View"].SetValue( viewMatrix );
            effect.Parameters["Projection"].SetValue( projectionMatrix );
        }

        mesh.Draw( );
    }
}
```

El código anterior se encuentra consigue la experiencia de realidad aumentada, por las siguientes razones: 1) llama al objeto a una escena real en lugar de llenarlo con color negro; 2) utiliza la matriz de transformación (escala, rotación y transformación) para poner el objeto virtual sobre el marcador. **[34]**

Para la escena de realidad aumentada que se necesita para hacer imágenes del mundo real, procedente de la cámara, un archivo o cualquier otra fuente que contenga algunos marcadores ópticos para reconocer. Sin entrar en la adquisición de vídeo-lectura de datos, se puede simplemente asumir que cada nuevo fotograma de vídeo se ofrece como mapa de bits de .NET. Aparentemente el framework XNA no se preocupa demasiado por GDI + mapas de bits y no proporciona medios para la prestación de estos. Así que se necesita un método o herramienta, que permite la conversión de mapas de bits en la textura de XNA 2D:

```
// Convert GDI+ bitmap to XNA texture
public static Texture2D XNATextureFromBitmap( Bitmap bitmap, GraphicsDevice
device )
{
    int width  = bitmap.Width;
    int height = bitmap.Height;
```

```
    Texture2D texture = new Texture2D( device, width, height,
        1, TextureUsage.None, SurfaceFormat.Color );

    BitmapData data = bitmap.LockBits( new Rectangle( 0, 0, width, height ),
        ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb );

    int bufferSize = data.Height * data.Stride;

    // copy bitmap data into texture
    byte[] bytes = new byte[bufferSize];
    Marshal.Copy( data.Scan0, bytes, 0, bytes.Length );
    texture.SetData( bytes );

    bitmap.UnlockBits( data );

    return texture;
}
```

Una vez que el mapa de bits esté contenido en el fotograma de vídeo actual y se convierta en la textura de XNA, puede ser aplicado a la representación de modelos 3D. La única tarea importante a destacar, es que después de hacer alguna presentación 2D, se requiere restaurar algunos estados del dispositivo gráfico de XNA; que son compartidos entre los gráficos 2D y 3D:

```
// draw texture containing video frame
mainSpriteBatch.Begin( SpriteBlendMode.None );
mainSpriteBatch.Draw( texture, new Vector2( 0, 0 ), Color.White );
mainSpriteBatch.End( );

// restore state of some graphics device's properties after 2D graphics,
// so 3D rendering will work fine
GraphicsDevice.RenderState.DepthBufferEnable = true;
GraphicsDevice.RenderState.AlphaBlendEnable  = false;
GraphicsDevice.RenderState.AlphaTestEnable   = false;

GraphicsDevice.SamplerStates[0].AddressU = TextureAddressMode.Wrap;
GraphicsDevice.SamplerStates[0].AddressV = TextureAddressMode.Wrap;
```

La última y la parte más importante es asegurarse que el tamaño, la posición y la rotación del modelo corresponden a la postura y la posición de un marcador que existe en el mundo real. Todo esto no es complejo en este punto, puesto que ya fue descrito en códigos anteriores. **[34]**

El algoritmo coplanar POSIT proporciona estimación de la matriz de rotación y el vector traslación. Algo como esto:

```
// estimate pose of the object
Matrix3x3 rotationMatrix;
Vector3 translationVector;
```

```
coposit.EstimatePose( imagePoints, out rotationMatrix, out translationVector );
```

Cuando se tiene rotación y traslación del marcador conocido, se puede actualizar la parte XNA con el fin de poner en su lugar correcto el modelo 3D, también utilizar la rotación y tamaño apropiado para ello. Aquí está la parte del código que calcula la matriz del modelo para la prestación XNA; se tiene que cambiar esta parte sólo para completar la escena de realidad aumentada, puesto que ya se tiene todo lo demás:

```
...
Matrix world = Matrix.CreateScale( 1 / mesh.BoundingSphere.Radius ) *
    rotation * translation;
...
```

Alguien puede pensar que potencialmente convertir matrices-vectores del framework AForge.NET a matrices de XNA debería ser suficiente para conseguir que todo funcione. Sin embargo no lo es. Aunque XNA utiliza la representación columna matricial, pero el framework AForge.NET utiliza representación fila. Además se debe tener cuidado en el hecho de que XNA utiliza un sistema de coordenadas diferente de la utilizada por el código de la estimación de pose. XNA utiliza un sistema de la mano derecha, donde el eje Z se dirige desde el origen al espectador cuando los ejes X e Y se dirigen a la derecha y arriba respectivamente. En el sistema de tales coordenadas, aumentar la coordenada Z de un objeto hace que sea más cerca la visualización (cámara), lo que hace que parezca más grande en la pantalla proyectada. Sin embargo, en el mundo real se tiene el caso contrario; mayor coordenada Z de un objeto significa que esta más lejos del espectador. Esto se conoce como sistema de coordenadas de la mano izquierda, cuando el eje Z apunta lejos del espectador y los ejes X e Y tienen la misma dirección (derecha-arriba). Así que se tiene que convertir la estimación de pose de las coordenadas del marcador de la mano izquierda con el sistema de la mano derecha. **[34]**

La primera parte de la conversión de coordenadas del mundo real a XNA, es negar al objeto la coordenada Z, por lo que el objeto se ve más lejos en el mundo real. La segunda parte es convertir ángulos de rotación del objeto, que niega la rotación alrededor de los ejes X e Y. **[34]**

Una cosa más importante, se tiene que escalar el modelo 3D de XNA. Como se ha visto anteriormente, se describe el modelo del marcador en milímetros. Así el algoritmo de

estimación del marcador también debe ser en milímetros. Esto dará como resultado del modelo de coordenada Z ajustado a aproximadamente -200, cuando un marcador está a unos 20 centímetros de distancia de la cámara, lo que hará que el modelo 3D se vea pequeño en la escena XNA si el tamaño original del modelo es pequeño. Así que todo lo que se tiene que hacer es escalar el modelo 3D, por lo que tiene el tamaño de "comparable" al tamaño del marcador. Poniendo todo esto en conjunto, se reemplazará la línea de código antes mencionada (que calcula la matriz XNA del objeto) con el siguiente código:

```
float modelSize = 113;

// extract rotation angles from the estimated rotation
float yaw, pitch, roll;
positRotation.ExtractYawPitchRoll( out yaw, out pitch, out roll );

// create XNA's rotation matrix
Matrix rotation = Matrix.CreateFromYawPitchRoll( -yaw, -pitch, roll );

// create XNA's translation matrix
Matrix translation = Matrix.CreateTranslation(
    positTranslation.X, positTranslation.Y, -positTranslation.Z );

// create scaling matrix, so model fits its glyph
Matrix scaling = Matrix.CreateScale( modelSize );

// finally compute XNA object's world matrix
Matrix world = Matrix.CreateScale( 1 / mesh.BoundingSphere.Radius ) *

    scaling * rotation * translation;
```

Finalmente la realidad aumentada está construida; con todo el código anterior, se debe tener una pantalla de XNA como se muestra en la Fig. 4.83. **[34]**

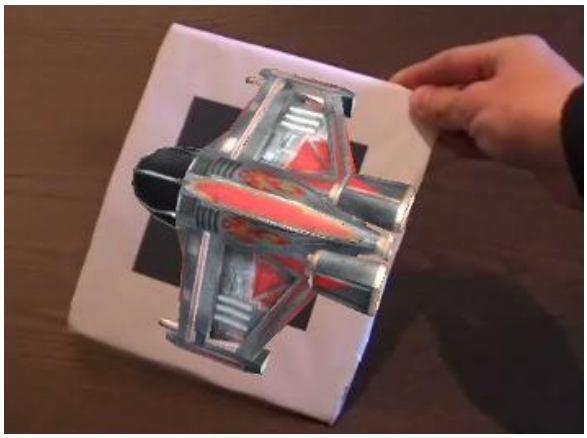


Fig. 4.83: Realidad Aumentada 3D
Fuente. http://www.codeproject.com/graphics/ar_scene.jpg - [34].

4.8.8 Pruebas del sistema PIVRA

Este apartado, consolida la parte evaluativa del sistema PIVRA, la valoración del software está afirmada en su arquitectura detallada. A continuación la Tabla 4.7 muestra la prueba de módulo único.

Tabla 4.7: Prueba módulo único

PRUEBAS PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA				
Nombre Prueba		Módulo único		
Realizada por		Programadora		
Descripción		Evaluar y depurar cada algoritmo de programación del software		
Nro.	**Categoría**	**Respuesta esperada del algoritmo**		**Res.**
1	Visión Artificial	Captura y visualización de video		OK
2	Visión Artificial	Detección y reconocimiento de esquinas		OK
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	OK
			Marcador RA	OK
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK
			Marcador de RA	OK
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	OK
			Modelo 3D	OK

Elaborado por: Jimena Caguana

A continuación, la Tabla 4.8 representa la prueba grupo de módulos.

Tabla 4.8: Prueba grupo de módulos

PRUEBAS PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA					
Nombre Prueba		Grupo de módulos			
Realizada por		Programadora			
Descripción		Evaluar y depurar cada interfaz de usuario del software			
Nro.	**Categoría**	**Respuesta esperada de la interfaz de usuario**		**Incremental ascendente**	**Incremental descendente**
1	Visión Artificial	Captura y visualización de video		OK	OK
2	Visión Artificial	Detección y reconocimiento de esquinas		OK	---
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	OK	---
			Marcador RA	OK	---
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK	---
			Marcador de RA	OK	---
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	OK	---
			Modelo 3D		OK

Elaborado por: Jimena Caguana

A continuación, la Tabla 4.9 describe la prueba de sistema completo.

Tabla 4.9: Prueba sistema completo

PRUEBAS PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA						
Nombre Prueba		Sistema completo				
Realizada por		Programadora				
Descripción		Evaluar y depurar el sistema PIVRA completo				
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Velocidad**	**Exactitud**	**Fiabilidad**
1	Visión Artificial	Captura y visualización de video		100%	100%	100%
2	Visión Artificial	Detección y reconocimiento de esquinas		90%	100%	95%
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	100%	100%	95%
			Marcador RA	100%	100%	95%
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	95%	100%	90%
			Marcador de RA	95%	100%	90%
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	100%	100%	100%
			Modelo 3D	90%	100%	95%

Elaborado por: Jimena Caguana

La evaluación en general es positiva, ya que el sistema PIVRA cumple con los objetivos establecidos en la propuesta y ha superado la prueba de módulo único y la prueba de grupo de módulos, con éxito. A su vez, la prueba de sistema completo obtuvo excelentes resultados en velocidad y exactitud; en la fiabilidad existe una reducción porcentual de entre 5% a 10%, aproximadamente, debido al entorno operativo e iluminación del aula o laboratorio donde se trabajó con el software.

4.8.9 Verificación del sistema PIVRA

El desarrollo del software PIVRA tuvo relativa dificultad, partiendo de la base en cuestión de programación; específicamente, el ámbito del entrenamiento de la cascada de Haar para el reconocimiento de objetos, además de la detección y reconocimiento de marcadores de Realidad Aumentada. La presente estructura de verificación se basa en la arquitectura detallada del sistema PIVRA, misma que es la operatividad principal de la aplicación. Además, está fundamentada de los casos de uso de la ingeniería de requisitos del software. La Tabla 4.10 muestra la verificación basada en el uso.

Tabla 4.10: Verificación basada en el uso

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA					
Nombre		Basada en el uso			
Realizada por		Programadora			
Descripción		Verificar el uso y el entorno de trabajo del sistema PIVRA			
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Perfil operacional**	**SRET**
1	Visión Artificial	Captura y visualización de video		100%	100%
2	Visión Artificial	Detección y reconocimiento de esquinas		100%	95%
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	100%	100%
			Marcador RA	100%	100%
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	98%	95%
			Marcador de RA	98%	95%
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	100%	98%
			Modelo 3D	98%	98%

Elaborado por: Jimena Caguana

La verificación del sistema PIVRA basada en el uso, estípula buenos resultados en virtud al perfil operacional o entorno de trabajo de la pizarra; un valor porcentual promedio de 99,25% reflejan aquel valor cualitativo. Además, el SRET verificado acredita un valor porcentual de 97,62%, beneficiando la ergonomía y fiabilidad del software de la Pizarra Virtual usando Realidad Aumentada.

Además de la verificación basada en el uso, se realizó la verificación basada en los casos de uso. Para la verificación basada en los casos de uso se inició en la bifurcación de los casos de uso planteados en la ingeniería de requisitos, dando como resultado las siguientes categorías de uso:

- Captura y visualización de video.
- Detección y reconocimiento de esquinas.
- Selección modo rotulador de RA – modo marcador de RA.
- Detección y reconocimiento letra de RA o marcador de RA.
- Proyección y visualización de Realidad Aumentada.

A continuación, la Tabla 4.11 detalla la verificación basada en los casos de uso, tabla que representa la operatividad totalitaria del sistema PIVRA.

La categoría de Visión Artificial y Realidad Aumentada fueron construidas a partir de multiples verificaciones, hasta la consecusión del prototipo del sistema PIVRA más eficaz a las características de un sistema de RA con capacidad interactiva.

Tabla 4.11: Verificación basada en los casos de uso

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA				
Nombre		Basada en los casos de uso		
Realizada por		Programadora		
Descripción		Verificar la ingeniería de requisitos y los casos de uso		
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Casos de uso**
1	Visión Artificial	Captura y visualización de video		Tabla 4.12
2	Visión Artificial	Detección y reconocimiento de esquinas		Tabla 4.13
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	Tabla 4.14
			Marcador RA	
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	Tabla 4.15
			Marcador de RA	
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	Tabla 4.16
			Modelo 3D	

Elaborado por: Jimena Caguana

La Tabla 4.12, a continuación muestra la verificación de la captura y visualización de video.

Tabla 4.12: Verificación captura y visualización de video

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA			
Nro.		1	
Nombre		Captura y visualización de video	
Realizada por		Usuario	
Descripción		Testear la adquisición y captura de video a través de una webcam	
Precondiciones		1. La cámara web debe tener sus respectivos drivers. 2. El usuario debe haber iniciado la aplicación.	
Postcondiciones		1. Visualizar el video en la interfaz del software PIVRA	
Paso	**Acción**	**Respuesta esperada del sistema**	**Res.**
1	Clic en el servicio pizarra	Cargando la captura y visualización del video	OK

Elaborado por: Jimena Caguana

La Tabla 4.13, a continuación muestra la verificación de la detección y reconocimiento de esquinas.

Tabla 4.13: Verificación detección y reconocimiento de esquinas

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA			
Nro.		2	
Nombre		Detección y reconocimiento de esquinas	
Realizada por		Usuario	
Descripción		Testear la detección y reconocimiento de la letra X en las esquinas de la pizarra	
Precondiciones		1. Capturar y visualizar el video 2. Utilizar una pizarra de marcador 3. Escribir la letra X en cada esquina de la pizarra	
Postcondiciones		1. Visualizar las letras X encerradas en un recuadro. 2. Establecer el área de trabajo de la pizarra virtual.	
Paso	**Acción**	**Respuesta esperada del sistema**	**Res.**
1	Clic en el servicio pizarra	Escribir con el rotulador la letra X en cada esquina de la pizarra	OK
2	Mostrar la pizarra a la cámara web	Presentar la detección y reconocimiento de las esquinas de la pizarra	OK

Elaborado por: Jimena Caguana

La Tabla 4.14, a continuación muestra la verificación de la selección modo rotulador de RA – modo marcador de RA.

Tabla 4.14: Verificación selección modo rotulador de RA – modo marcador de RA

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA			
Nro.		3	
Nombre		Selección modo rotulador de RA – modo marcador de RA	
Realizada por		Usuario	
Descripción		Testear la solicitud del usuario en el modo de trabajo del software	
Precondiciones		1. Capturar y visualizar el video 2. Detectar y reconocer las esquinas de la pizarra 3. Seleccionar el modo rotulador o modo marcador de RA	
Postcondiciones		1. Modo rotulador de RA: Escribir una letra en la pizarra 2. Modo marcador de RA: Mostrar un marcador de RA	
Paso	**Acción**	**Respuesta esperada del sistema**	**Res.**
1	Clic en el servicio rotulador de RA	Cargando el servicio modo rotulador para escribir una letra en la pizarra y experimentar la Realidad Aumentada	OK
2	Clic en el servicio marcador de RA	Cargando el servicio modo marcador de RA para mostrar una marcador de RA junto a la pizarra y experimentar la Realidad Aumentada	OK

Elaborado por: Jimena Caguana

La Tabla 4.15, a continuación muestra la verificación de la detección y reconocimiento letra de RA o marcador de RA.

Tabla 4.15: Verificación detección y reconocimiento letra de RA o marcador de RA

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA			
Nro.		4	
Nombre		Detección y reconocimiento letra de RA o marcador de RA	
Realizada por		Usuario	
Descripción		Testear la detección y reconocimiento del marcador de RA o la letra de RA	
Precondiciones		1. Capturar y visualizar el video 2. Detectar y reconocer las esquinas de la pizarra 3. Seleccionar el modo rotulador o modo marcador de RA 4. Modo rotulador de RA: Escribir una letra en la pizarra 5. Modo marcador de RA: Mostrar un marcador de RA	
Postcondiciones		1. Seleccionar el modelo de proyección imagen 2D o imagen 3D 2. Seleccionar el modo de uso modo pizarra – modo presentador	
Paso	**Acción**	**Respuesta esperada del sistema**	**Res.**
1	Clic en el servicio modelo de proyección	Cargando el modelo de proyección imagen tipo 2D o imagen tipo 3D.	OK
2	Modo de uso	Mantener el modo de uso pizarra o clic en el servicio modo de uso presentador.	OK

Elaborado por: Jimena Caguana

La Tabla 4.16, a continuación muestra la verificación de la proyección y visualización de Realidad Aumentada.

Tabla 4.16: Verificación proyección y visualización de Realidad Aumentada

VERIFICACIÓN PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA			
Nro.		5	
Nombre		Proyección y visualización de Realidad Aumentada	
Realizada por		Usuario	
Descripción		Testear la proyección y visualización de Realidad Aumentada	
Precondiciones		1. Capturar y visualizar el video 2. Detectar y reconocer las esquinas de la pizarra 3. Seleccionar el modo rotulador o modo marcador de RA 4. Modo rotulador de RA: Escribir una letra en la pizarra 5. Modo marcador de RA: Mostrar un marcador de RA 6. Seleccionar el modelo de proyección imagen 2D o imagen 3D 7. Seleccionar el modo de uso modo pizarra – modo presentador	
Postcondiciones		1. Utilizar un proyector multimedia	
Paso	**Acción**	**Respuesta esperada del sistema**	**Res.**
1	Encender el proyector multimedia	Cargando la visión de Realidad Aumentada al área de exposición del proyector multimedia.	OK

Elaborado por: Jimena Caguana

4.8.10 Mantenimiento del sistema PIVRA

El estándar IEEE 1219 define el Mantenimiento del Software como "la modificación de un producto software con el fin de corregir defectos, mejorar el rendimiento u otros atributos". La Tabla 4.17 muestra el mantenimiento correctivo del sistema PIVRA.

Tabla 4.17: Mantenimiento correctivo

MANTENIMIENTO PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA							
Nombre		Mantenimiento correctivo					
Realizada por		Programadora					
Descripción		Localizar y eliminar posibles defectos del software					
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Procesamiento**	**Rendimiento**	**Programación**	**Documentación**
1	Visión Artificial	Captura y visualización de video		---	OK	OK	OK
2	Visión Artificial	Detección y reconocimiento de esquinas		---	OK	OK	OK
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	---	OK	OK	OK
			Marcador RA	---	OK	OK	OK
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK	OK	OK	OK
			Marcador de RA	OK	OK	OK	OK
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	---	OK	OK	OK
			Modelo 3D	---	OK	OK	OK

Elaborado por: Jimena Caguana

El mantenimiento del sistema PIVRA de tipo correctivo postularon la iniciativa ideal para programar de manera distributiva las líneas de código que constituyen toda la interactividad de la pizarra. Inicialmente la pizarra necesitaba una gran cantidad de memoria RAM y velocidad del procesamiento; pero a partir de una depuración con procesos correctivos en cada categoría, la funcionalidad en visión artificial y RA del prototipo, se ejecutó eficientemente.

Después del mantenimiento correctivo, se realizó el mantenimiento adaptativo; todos los tipos de mantenimiento tuvieron su respectiva retroalimentación, en virtud a la performance de los casos de uso. La Tabla 4.18, a continuación muestra el mantenimiento adaptativo del sistema PIVRA.

Tabla 4.18: Mantenimiento adaptativo

MANTENIMIENTO PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA				
Nombre		Mantenimiento adaptativo		
Realizada por		Programadora		
Descripción		Identificar y modificar el programa debido a cabios en el entorno operativo		
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Entorno en los procesos**
1	Visión Artificial	Captura y visualización de video		OK
2	Visión Artificial	Detección y reconocimiento de letra		OK
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	OK
			Marcador RA	OK
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK
			Marcador de RA	OK
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	OK
			Modelo 3D	OK

Elaborado por: Jimena Caguana

La Tabla 4.19, a continuación muestra el mantenimiento perfectivo del sistema PIVRA.

Tabla 4.19: Mantenimiento perfectivo

MANTENIMIENTO PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA					
Nombre		Mantenimiento perfectivo			
Realizada por		Programadora			
Descripción		Cambiar la especificación en base a los requisitos del software			
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Perfectivo de ampliación**	**Perfectivo de eficiencia**
1	Visión Artificial	Captura y visualización de video		OK	OK
2	Visión Artificial	Detección y reconocimiento de letra		OK	OK
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	OK	OK
			Marcador RA	OK	OK
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK	OK
			Marcador de RA	OK	OK
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	OK	OK
			Modelo 3D	OK	OK

Elaborado por: Jimena Caguana

La Tabla 4.20, a continuación muestra el mantenimiento preventivo del sistema PIVRA.

Tabla 4.20: Mantenimiento preventivo

MANTENIMIENTO PIZARRA VIRTUAL USANDO REALIDAD AUMENTADA				
Nombre		Mantenimiento preventivo		
Realizada por		Programadora		
Descripción		Modificar el software para mejorar sus propiedades sin alterar sus especificaciones funcionales		
Nro.	**Categoría**	**Respuesta esperada del sistema PIVRA**		**Reutilización**
1	Visión Artificial	Captura y visualización de video		OK
2	Visión Artificial	Detección y reconocimiento de letra		OK
3	Visión Artificial – Realidad Aumentada	Selección modo de trabajo	Rotulador RA	OK
			Marcador RA	OK
4	Realidad Aumentada	Detección y Reconocimiento	Letra de RA	OK
			Marcador de RA	OK
5	Realidad Aumentada	Proyección y visualización de RA	Modelo 2D	OK
			Modelo 3D	OK

Elaborado por: Jimena Caguana

Tanto el mantenimiento correctivo, mantenimiento adaptativo, mantenimiento perfectivo y mantenimiento preventivo lograron imponer la interactividad y las técnicas de visión computacional con realidad aumentada en el sistema PIVRA, al proponer en cada categoría de desarrollo del prototipo una depuración informática en las líneas de código, hasta la consecución de un sistema concurrente a los objetivos de la propuesta.

4.8.11 Evaluación con el sistema PIVRA

El sistema PIVRA consta de interfaces evaluativas, ejecutadas a través de la barra de menú del formulario principal, con las cuales el docente podrá evaluar el nivel de percepción de los alumnos en la clase dictada según su cronograma de enseñanza, mejorando la interactividad docente, alumno y educación. Para ello se establece siete formularios con cuestionarios estipulados de la siguiente manera:

Cerebro:

La Tabla 4.21, a continuación muestra las preguntas aplicadas en la evaluación de la clase del cerebro.

Tabla 4.21: Preguntas y respuestas – Cerebro

Pregunta	Respuestas
1.- Son partes principales del cerebro:	a.- Parietal y encéfalo b.- Retícula e hipocampo c.- Hemisferios y surcos d.- Cerebelo e hipocampo
2.- Los ventrículos cerebrales están constituidos por:	a.- Pineal, tercer y cuarto ventrículo b.- Ventrículos laterales y encéfalo c.- Ventrículos laterales, segundo y tercer ventrículo d.- Ventrículos laterales, tercer y cuarto ventrículo
3.- Son sistemas de fibras mielinicas que conectan un hemisferio con el contralateral:	a.- Los ventrículos b.- El cuerpo calloso c.- Hemisferios d.- El encéfalo
4.- La parte del cerebro que se encarga de controlar y manejar el movimiento corporal es:	a.- Subtalamo b.- Núcleo estriado c.- Glándula pineal d.- Caudado

Elaborado por: Jimena Caguana

Corazón:

La Tabla 4.22, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Corazón.

Tabla 4.22: Preguntas y respuestas – Corazón

Pregunta	Respuestas
1.- ¿Estructura que se halla internamente en la cara anterior de la aurícula derecha?	a.- Miocardio b.- Válvula de tebesio c.- Válvula tricúspide
2.- ¿Músculos que se encuentran internamente en las caras externas de cada aurícula?	a.- Pectorales b.- Pectíneos c.- Perforantes
3.- ¿Qué estructura interna pertenece a la cara media de la aurícula izquierda?	a.- Orejuela b.- Válvula de parchappe c.- Válvula mitral
4.- La función del pericardio es:	a.- Permitir el movimiento cardiaco b.- Cubrir a la arteria pulmonar c.- Cubrir a la aorta

Elaborado por: Jimena Caguana

Estomago:

La Tabla 4.23, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Estómago.

Tabla 4.23: Preguntas y respuestas – Estómago

Pregunta	Respuestas
1.- ¿Cuáles son las partes del estómago?	a.- Fundus, cuerpo, antro y píloro b.- Retícula e hipocampo c.- Hemisferios y Surcos d.- Cerebelo e hipocampo
2.- ¿Cuáles son las dimensiones del estómago?	a.- Mide 12 cm de largo, 12 cm de ancho. b.- Mide 25 cm de largo, 12 cm de ancho y 8 cm antero posteriormente c.- Mide 15 cm de largo, 8 cm antero posteriormente d.- Mide 35 cm de largo, 10 cm de ancho y 4 cm antero posteriormente
3.- ¿Qué es el cardias?	a.- Es la bifurcación de la parte inferior de la tráquea b.- Es una estructura situada en la parte media del cuello c.- Es un repliegue o esfínter mucoso fisiológico d.- Es una estructura impar de forma tabula
4.- ¿Qué es el píloro?	a.- Es un repliegue o esfínter mucoso fisiológico b.- Es el esquema de un modular c.- Es el nombre que recibe el conjunto de dientes d.- Es un esfínter anatómico que une al duodeno del estómago

Elaborado por: Jimena Caguana

Hígado:

La Tabla 4.24, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Hígado.

Tabla 4.24: Preguntas y respuestas – Hígado

Pregunta	Respuestas
1.- ¿El hígado recibe sangre mediante 2 vías, escoja la opción correcta?	a.- Vía de la vena porta y vena cava inferior b.- Vía de las arterias hepáticas y vena porta c.- Vía de la vena cava inferior y arteria hepática
2.- ¿Cuál de las siguientes funciones cumple el hígado?	a.- Producción de amonio y urea b.- Almacenamiento de vitaminas c.- Todas las anteriores
3.- Escoja lo correcto, en el Mecanismo fisiopatológico de la cirrosis sucede:	a.- Eliminación de glucosa b.- Proliferación de leucocitos c.- Necrosis de los hepatocitos
4.- Escoja lo correcto. Colangitis es:	a.- Es la obstrucción de la vena porta b.- Causada por parasitosis c.- Es la infección del conducto colédoco

Elaborado por: Jimena Caguana

Intestino:

La Tabla 4.25, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Intestino.

Tabla 4.25: Preguntas y respuestas – Intestino.

Pregunta	Respuestas
1.- ¿Qué es el apéndice vermiforme?	a.- Prolongación del ciego que nace de su pared medial 2 o 3 cm inferiormente al orificio cecal b.- Membranas de naturaleza serosa que envuelven y protegen el Intestino c.- Órganos respiratorios pares, voluminosos y de forma cónica
2.- ¿Cuánto mide el apéndice vermiforme?	a.- 5 a 6 cm longitud y de 4 a 8 Mm diámetro b.- 7 a 8 cm longitud y de 4 a 8 Mm diámetro c.- 7 a 8 cm longitud y de 1 a 2 Mm diámetro
3.- ¿Cuál es la causa más frecuente de obstrucción del intestino delgado?	a.- Neoplásica b.- Hernias c.- Bridas
4.- ¿Cuál es la causa más frecuente de obstrucción de intestino grueso?	a.- Neoplásica b.- Hernias c.- Bridas

Elaborado por: Jimena Caguana

Pulmones:

La Tabla 4.26, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Pulmones.

Tabla 4.26: Preguntas y respuestas – Pulmones

Pregunta	Respuestas
1.- ¿Dónde están situados los pulmones?	a.- Ambos lados del tórax. b.- Ambos bronquios c.- En la tráquea
2.- ¿Cómo se llama el espacio que divide los dos pulmones?	a.- Cisuras b.- Mediastino c.- Tráquea
3.- El pulmón derecho se divide en tres partes:	a.- Primaria, Secundaria, Terciaria b.- Fibra, Alvéolo, Bronquiolo c.- Superior, Medio, Inferior
4.- El pulmón izquierdo se divide en dos partes:	a.- Superior, Inferior b.- Primario, Secundario c.- Arriba, Abajo

Elaborado por: Jimena Caguana

Riñón:

La Tabla 4.27, a continuación muestra las preguntas aplicadas en la evaluación de la clase del Riñón.

Tabla 4.27: Preguntas y respuestas – Riñón

Pregunta	Respuestas
1.- ¿El riñón es un órgano par de?	a.- Cavidad pleural b.- Naturaleza fundamentalmente excretora c.- El Riñón
2.- ¿Los dos riñones están situados junto a la?	a.- Membrana cerosa b.- Corteza Renal c.- Pared dorsal del cuerpo
3.- El riñón tiene forma de:	a.- Un Corazón b.- Habichuela c.- Pera
4.- Cada riñón pesa alrededor de:	a.- 150 g b.- 50 g c.- 100 g

Elaborado por: Jimena Caguana

NOTA: Cada interrogante que interviene en el proceso evaluativo para las partes biológicas internas del ser humano, son plateadas por el docente de la asignatura de Ciencias Naturales, en performance al contenido teórico de la clase dictada.

4.9. Estudio Económico

El presente Trabajo de Graduación no está dirigido a generar beneficios tangibles o económicos; pero es imperante realizar un análisis económico del capital invertido en el desarrollo e implementación de la Pizarra Virtual usando Realidad Aumentada (PIVRA). En el estudio económico, se contempla:

- Descripción general.
- Precios unitarios.
- Sumas parciales.
- Presupuesto general.
- Recuperación del capital invertido en el sistema PIVRA.

4.9.1 Descripción General

Los valores financieros necesarios para la implementación del sistema PIVRA; los costos pueden variar al cambiar los modelos y/o marcas del hardware, así como el tipo de licencias de software utilizados en el desarrollo.

Hardware:

La Tabla 4.28 muestra el hardware utilizado en el desarrollo e implementación del sistema PIVRA:

Tabla 4.28: Hardware

Hardware	Cantidad	Horas de proyecto	Horas de uso al año
Ordenador Intel core i3, 2GHz, 4G RAM.	1	960	1440
Cámara Web Genius Face Cam 321, 8MP.	1	800	1136
Impresora HP LaserJet CP1515n	1	56	56
Proyector Multimedia	1	160	173

Elaborado por. Jimena Caguana

Software:

La Tabla 4.29 muestra en detalle el software utilizado en el diseño y desarrollo del sistema PIVRA:

Tabla 4.29: Software

Software	Cantidad	Horas de proyecto	Horas de uso al año
Sistema Operativo Microsoft Windows 8.1	1	960	1440
Microsoft Visual Studio Professional 2010	1	800	1136
Librería OpenCV 2.4.9	1	800	1136
Librerías EmguCV 2.4.9 – AForge.NET 2.2.5	1	800	1136
Programa GIMP 2.8	1	80	80
Programa Autodesk 3ds Max 2010	1	480	600
Microsoft Office 2010	1	960	1440

Elaborado por. Jimena Caguana

Herramientas:

La Tabla 4.30 muestra las herramientas utilizadas para la implementación y funcionamiento del sistema PIVRA:

Tabla 4.30: Herramientas

Elemento	Cantidad	Horas de proyecto	Horas de uso al año
Pizarra de vidrio	1	800	1136
Roturador color rojo	2	640	851

Elaborado por. Jimena Caguana

Mano de obra directa:

La Tabla 4.31 muestra las horas de mano de obra directa requerida en el proceso de implementación del sistema PIVRA:

Tabla 4.31: Mano de obra directa

Actividad	Horas
Ingeniería de software	40
Programación del software PIVRA	640
Estado del arte de los marcadores para Realidad Aumentada	40
Modelado 2D y 3D	320
Ingeniería de usabilidad	24
Implementación, pruebas y solución de problemas	40
Documentación del proyecto	360
Horas totales	1464

Elaborado por. Jimena Caguana

4.9.2 Precios Unitarios

Se detallan los precios de cada uno de los elementos necesarios para llevar a cabo la implementación del sistema PIVRA, los cuales se han citado anteriormente.

Hardware:

La Tabla 4.32 muestra el hardware del sistema PIVRA y su precio unitario:

Tabla 4.32: Hardware - Precio Unitario

Hardware	Cantidad	Precio ($/unid)
Ordenador Intel core i3, 2GHz, 4G RAM.	1	780,00
Cámara Web Genius Face Cam 321, 8MP.	1	36,00
Impresora HP LaserJet CP1515n	1	280,00
Proyector Multimedia	1	620,00

Elaborado por. Jimena Caguana

Software:

La Tabla 4.33 detalla el software utilizado en el sistema PIVRA con sus respectivos precios unitarios:

Tabla 4.33: Software - Precio Unitario

Software	Cantidad	Precio ($/unid)
Sistema Operativo Microsoft Windows 8.1	1	150,00
Microsoft Visual Studio Professional 2010	1	4,50
Librería OpenCV 2.4.9	1	3,50
Librerías EmguCV 2.4.9 – AForge.NET 2.2.5	1	3,50
Programa GIMP 2.8	1	1,50
Programa Autodesk 3ds Max 2010	1	4,50
Microsoft Office 2010	1	4,50

Elaborado por. Jimena Caguana

Herramientas:

La Tabla 4.34 describe las herramientas necesarias para el funcionamiento del sistema PIVRA con sus precios unitarios:

Tabla 4.34: Herramientas - Precio Unitario

Elemento	Cantidad	Precio ($/unid)
Pizarra de vidrio	1	13,00
Roturador color rojo	2	4,80

Elaborado por. Jimena Caguana

Mano de obra directa:

La Tabla 4.35 muestra la mano de obra directa útil para la implementación del sistema PIVRA, con sus respectivos precios unitarios:

Tabla 4.35: Mano de obra directa - Precio Unitario

Actividad	Horas	Precio ($/hora)
Ingeniería de software	40	5,00
Programación del software PIVRA	640	8,00
Estado del arte de los marcadores para Realidad Aumentada	40	5,00
Modelado 2D y 3D	320	5,00
Ingeniería de usabilidad	24	3,00
Implementación, pruebas y solución de problemas	40	3,00
Documentación del proyecto	360	3,00

Elaborado por. Jimena Caguana

4.9.3 Sumas Parciales

Se detallan los importes parciales de cada uno de los elementos que componen las distintas categorías, que han sido calculados a partir de la descripción general y los precios unitarios.

Hardware:

La Tabla 4.36 muestra el hardware del sistema PIVRA con su respectiva suma parcial:

Tabla 4.36: Hardware - Suma Parcial

Hardware	Cantidad	Horas de proyecto	Horas de uso al año	Precio ($/unid)	Amortización anual	Coste ($)
Ordenador Intel core i3, 2GHz, 4G RAM.	1	960	1440	780,00	26%	135,20
Cámara Web Genius Face Cam 321, 8MP.	1	800	1136	36,00	26%	6,59
Impresora HP LaserJet CP1515n	1	56	56	280,00	26%	72,80
Proyector Multimedia	1	160	173	620,00	26%	10,27
					SUBTOTAL	224,86

Elaborado por. Jimena Caguana

Software:

La Tabla 4.37 muestra el software utilizado para el diseño y desarrollo del sistema PIVRA con la respectiva suma parcial:

Tabla 4.37: Software - Suma Parcial

Software	Cantidad	Horas de proyecto	Horas de uso al año	Precio ($/unid)	Amortización anual	Coste ($)
Sistema Operativo Microsoft Windows 8.1	1	960	1440	150,00	26%	26,00
Microsoft Visual Studio Professional 2010	1	800	1136	4,50	26%	0,82
Librería OpenCV 2.4.9	1	800	1136	3,50	26%	0,64
Librerías EmguCV 2.4.9 – AForge.NET 2.2.5	1	800	1136	3,50	26%	0,64
Programa GIMP 2.8	1	80	80	1,50	26%	0,39
Programa Autodesk 3ds Max 2010	1	480	600	4,50	26%	0,94
Microsoft Office 2010	1	960	1440	4,50	26%	0,78
					SUBTOTAL	30,21

Elaborado por. Jimena Caguana

Herramientas:

La Tabla 4.38 muestra las herramientas utilizadas en la implementación y operatividad del sistema PIVRA con la suma parcial:

Tabla 4.38: Herramientas - Suma Parcial

Elemento	Cantidad	Horas de proyecto	Horas de uso al año	Precio ($/unid)	Amortización anual	Coste ($)
Pizarra de vidrio	1	800	1136	13,00	30%	2,75
Roturador color rojo	2	640	851	4,80	30%	1,08
					SUBTOTAL	3,83

Elaborado por. Jimena Caguana

Mano de obra directa:

La Tabla 4.39 muestra el coste de las horas de mano de obra directa, requerida en el proceso de implementación del sistema PIVRA:

Tabla 4.39: Mano de obra directa - Suma Parcial

Actividad	Horas	Precio ($/hora)	Coste ($)
Ingeniería de software	40	5,00	200,00
Programación del software PIVRA	640	8,00	5.120,00
Estado del arte de los marcadores para Realidad Aumentada	40	5,00	200,00
Modelado 2D y 3D	320	5,00	1.600,00
Ingeniería de usabilidad	24	3,00	72,00
Implementación, pruebas y solución de problemas	40	3,00	120,00
Documentación del proyecto	360	3,00	1.080,00
		SUBTOTAL	8.392,00

Elaborado por. Jimena Caguana

4.9.4 Presupuesto General

Sumando la contribución de todas las categorías anteriores, se obtiene que el coste total del proyecto; impuestos incluidos, asciende al valor estipulado en la Tabla 4.40:

Tabla 4.40: Presupuesto General

Concepto	Coste Subtotal ($)
Hardware	224,86
Software	30,21
Herramientas	3,83
Mano de obra directa	8.392,00
Imprevistos (5%)	432,54
TOTAL	**9.083,44**

Elaborado por. Jimena Caguana

El presupuesto del presente Trabajo de Graduación, asciende a la cantidad de ***NUEVE MIL OCHENTA Y TRES DÓLARES*** *con* ***CUARENTA Y CUATRO CENTAVOS****. Dinero financiado por la Unidad Educativa "Tirso de Molina" y por la Investigadora.*

4.9.5 Recuperación del capital invertido en el sistema PIVRA

Tras haber realizado el presupuesto general que supone el coste de desarrollo del proyecto, se procede a estudiar económicamente la viabilidad del proyecto. En primer lugar, se debe destacar que observado el presupuesto, la primera unidad del sistema

PIVRA tiene un coste de 9.083,44 dólares. Lógicamente es un precio excesivo, debido a ser el primer prototipo, en el cual se ha invertido mucho tiempo para su estudio y desarrollo.

Antes de realizar el análisis de rentabilidad económico, se asumirán algunas premisas:

- El sistema PIVRA, es un software con propósito educativo que no tiene ingresos, solo se considerarán los costes necesarios para dar soporte y mantenimiento.
- Se ha considerado que para dar soporte y mantenimiento a un volumen de 4 licencias GPL, se contará con la participación de una persona con perfil técnico.

La Tabla 4.41 muestra el destino de las licencias GPL del sistema PIVRA.

Tabla 4.41: Licencias GPL del sistema PIVRA

Entorno	Nº de Licencias
Laboratorio de Computación Nº 1	1
Laboratorio de Computación Nº 2	1
Exposición en el aula	2

Elaborado por. Jimena Caguana

La Tabla 4.42 detalla el coste del soporte y mantenimiento analizado para el funcionamiento del sistema PIVRA en la Unidad Educativa "Tirso de Molina".

Tabla 4.42: Coste del soporte y mantenimiento

Coste	Mensual ($)	Anual ($)
Salario Analista Programador (Soporte segundo nivel)	20,00	240,00
Salario Técnico Informático (Soporte primer nivel)	10,00	120,00
Gastos marcadores y roturador		50,00
Coste Total copia licenciada/año		**102,50**

Elaborado por. Jimena Caguana

Cálculo del precio de venta por copia:

El coste del mantenimiento anual de la licencia para cada copia es de 102,50 dólares, al que se incorporará el coste de desarrollo del producto. La Tabla 4.43 detalla el precio de licencia unitario anual que acredita el sistema PIVRA.

Tabla 4.43: Precio de licencia unitario anual

Concepto	Coste ($)
Coste soporte anual por copia para 4 unidades	410,00
Coste desarrollo sistema PIVRA para 4 unidades	9.083,44
COSTE TOTAL	9.493,44
Precio de alquiler unitario para 4 licencias	2.373,00
Margen comercial 30 %	712,00
PRECIO DE LICENCIA UNITARIO ANUAL	**3.085,00**

Elaborado por. Jimena Caguana

Dado un precio de la licencia anual de $ 3.085/licencia a un margen del 30%; con un coste, más que razonable para el usuario de $ 257,08/mes; soporte técnico incluido, lo que constituye una oferta interesante calidad/precio para este tipo de aplicaciones de visión artificial.

Análisis económico:

En el análisis económico se realiza el cálculo de los siguientes factores:

- VAN (Valor Actual Neto).
- TIR (Tasa Interna de Retorno).
- PRC (Periodo de Recuperación de Capital).

Estos factores proporcionan la idea general de rentabilidad del sistema PIVRA. En la rentabilidad la Unidad Educativa "Tirso de Molina" es beneficiada al no pagar el coste de licencia anual, únicamente el coste anual de soporte y mantenimiento para 4 licencias GPL (410,00 dólares). El ahorro para la institución se plantea con los siguientes valores en cada uno de los años establecidos en la Tabla 4.44.

Tabla 4.44: Descripción de valores económicos anuales pretendidos

Año	Valor de licencia	Margen comercial de operación (30 %)	Usabilidad (6,5 %)	Total ingresos ($)
1	2.373,00	712,00	0,00	3.085,00
2	3.085,00	925,50	260,68	4.271,18
3	4.271,18	1.281,35	360,91	5.913,44

4	5.913,44	1.774,03	499,68	8.187,15
5	8.187,15	2.456,14	691,81	11.335,10
6	11.335,10	3.400,53	957,82	15.693,45

Elaborado por. Jimena Caguana

Valor Actual Neto (VAN): Este factor, es el encargado de determinar el valor de los flujos de costos e ingresos generados a través de la vida útil del sistema PIVRA. Alternativamente está actualización puede aplicarse al flujo neto de los ingresos y gastos que se utilizarán en todos y cada uno de los años de operación económica de la Pizarra Virtual usando Realidad Aumentada. El VAN representa en valores actuales, el retorno líquido actualizado generado por el sistema PIVRA.

Si el VAN es mayor que cero, el sistema PIVRA es conveniente; si el VAN es igual a cero, el sistema PIVRA resulta indiferente y si el VAN es menor a cero, el sistema PIVRA no es viable; para determinar este valor se utiliza la ecuación:

$$\boldsymbol{VAN} = \sum_{t=1}^{T} \left[\frac{FC_t}{(1+i)^t}\right] - I_0$$

Donde:

$FC =$ Factor de actualización.

$i =$ Tasa de rentabilidad de la institución (11,15 %).

$T =$ Periodo durante el cual se requiere capitalizar la inversión (6 años).

$I_0 =$ Inversión inicial del sistema PIVRA (**9.083,44**).

A continuación se detalla en la Tabla 4.45 el cálculo del VAN.

Tabla 4.45: Descripción del cálculo del VAN

Año	Ingresos ($)	Egresos ($)	Egre. IVA ($)	Flujo Caja ($)	Fac. Act. (0,1115)	Fondos
0	0	9.083,44	0	-9.083,44		-9.083,44
1	3.085,00	1.677,51	370,20	1.037,29	0,8997	933,25
2	4.271,18	2.322,51	512,54	1.436,13	0,8094	1.162,40
3	5.913,44	3.215,51	709,61	1.988,32	0,7282	1.447,89

4	8.187,15	4.451,87	982,46	2.752,82	0,6552	1.803,65
5	11.335,10	6.163,62	1.360,21	3.811,27	0,5895	2.246,74
6	15.693.45	8.533,53	1.883,21	5.276,71	0,5303	2.798,24
$VAN = (933{,}2 + 1162{,}4 + 1447{,}9 + 1803{,}6 + 2246{,}7 + 2798{,}2) - 9083{,}44$						
$\boldsymbol{VAN = 1.308,73}$						

Elaborado por. Jimena Caguana

Previo al análisis de la Tasa Interna de Retorno (TIR), se define una tasa de actualización, que se encuentra en función a la inflación proyectada al país y razón social del proyecto, puesto que esta es una estimación de riesgo sobre proyectos. La Tabla 4.46 muestra esta tasa de actualización para proyectos.

Tabla 4.46: Tasa de actualización para proyectos

Clasificación de proyectos	Tasa de actualización	
Proyectos sociales sin fines de lucro	7 %	10 %
Proyectos bajo financiamiento estatal	12 %	14 %
Proyectos bajo financiamiento privado	11 %	13 %
Proyectos mixtos	13 %	15 %

Fuente. Ley de Régimen Tributario del Ecuador
Elaborado por. Jimena Caguana

El proyecto del Trabajo de Graduación, se ubica entre un rango del 13 % y 15 %; limites que son de utilidad para obtener un VAN menor y un VAN mayor indispensables en el cálculo de la TIR.

Tasa Interna de Retorno (TIR): Se puede interpretar a la TIR, como la más alta tasa de interés que se podría pagar por un préstamo que financiará la inversión, si el préstamo con los intereses acumulados a esta tasa dada, se fuese abonando con los ingresos provenientes del proyecto, a medida que estos van siendo generados a través de toda la vida útil del proyecto. Si la TIR es mayor que el costo del capital debe aceptarse el proyecto.

La siguiente ecuación, se utiliza para calcular la TIR:

$$TIR = Tasa\ Menor + \left[Difer.de\ tasas * \frac{VAN\ tasa\ menor}{VAN\ menor - VAN\ mayor}\right]$$

A continuación se detalla en la Tabla 4.47 el cálculo de la TIR.

Tabla 4.47: Descripción del cálculo de la TIR

		Actualización			
Año	**Flujo Neto**	**Fac. Act. 13 %**	**VAN Menor**	**Fac. Act. 15 %**	**VAN Mayor**
0	9.083,44				
1	1.037,29	0,8849	917,90	0,8696	902,03
2	1.436,13	0,7831	1.124,63	0,7561	1.085,86
3	1.988,32	0,6930	1.377,91	0,6575	1.307,32
4	2.752,82	0,6133	1.688,30	0,5717	1.573,79
5	3.811,27	0,5428	2.068,76	0,4972	1.894,96
6	5.276,71	0,4803	2.534,40	0,4323	2.281,12
		TOTAL	**9.711,90**	**TOTAL**	**9.045,08**

Elaborado por. Jimena Caguana

$$TIR = 13 + \left[2 + \frac{9.711,90}{9.711,90 - 9.045,08}\right]$$

$$TIR = 13 + 29,13$$

$$\boldsymbol{TIR = 42,13\ \%}$$

Periodo de Recuperación de Capital (PRC): Es el tiempo requerido para recuperar la inversión original, en una medida de la rapidez con que el proyecto reembolsará el capital inicial. El periodo de recuperación consiste en el número de años requeridos para recobrar la inversión preliminar; en la Tabla 4.48 se demuestra el tiempo requerido para que el proyecto recupere la inversión inicial de capital.

Tabla 4.48: Descripción del cálculo de la PRC

Año	**Flujo Caja**	**Fac. Act. (11,15 %)**	**Fondos**	**Acumulado**
0	- 9.083,44			-9.083,44
1	1.037,29	0,8997	933,25	-8.150,19
2	1.436,13	0,8094	1.162,40	-6.987,79
3	1.988,32	0,7282	1.447,89	-5.539,90
4	2.752,82	0,6552	1.803,65	-3.736,25
5	3.811,27	0,5895	2.246,74	-1.489,51
6	5.276,71	0,5303	2.798,24	1.308,73

Elaborado por. Jimena Caguana

Para el cálculo de la PRC, se utiliza la expresión de la ecuación:

$$PRC = Año\ que\ supera\ la\ invers. + \frac{Inversión - \sum primeros\ flujos}{Flujo\ neto\ del\ año\ que\ supera\ la\ invers.}$$

$$PRC = 6 + \frac{9083{,}44 - (2798{,}2 + 2246{,}7 + 1803{,}6 + 1447{,}9 + 1162{,}4 + 933{,}2)}{2798{,}24}$$

$$PRC = 6 + \frac{9083{,}44 - (10392)}{2798{,}24}$$

$$PRC = 6 + (-0{,}4676)$$

$$\boldsymbol{PRC = 5{,}5324} \qquad \textbf{Indica 5 años.}$$

$$PRC = 5{,}5324 - 5 = 0{,}5324$$

$$\boldsymbol{PRC = 0{,}5324 * 12 = 6{,}3888} \qquad \textbf{Indica 6 meses.}$$

$$PRC = 6{,}3888 - 6 = 0{,}3888$$

$$\boldsymbol{PRC = 0{,}3888 * 30 = 11{,}66} \qquad \textbf{Indica 11 días.}$$

El retorno de capital invertido es **5 años, 6 meses y 11 días**; finalmente de acuerdo al análisis de factibilidad se pudo determinar la evaluación económica del proyecto en donde se pudo establecer que el VAN es mayor que cero **$ 1.308,73**; por ende el proyecto es conveniente. En lo que respecta a la TIR se encuentra en un **42,13 %** dando como resultado que la TIR es mayor que el costo del capital (11,15 %) asegurando la viabilidad del sistema PIVRA implementado en la Unidad Educativa "Tirso de Molina".

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Durante el desarrollo de este Trabajo de Graduación se han presentado aspectos relevantes en lo concerniente al diseño e implementación de la Pizarra Virtual usando Realidad Aumentada (PIVRA), detallados en las siguientes conclusiones:

- Se diseñó, desarrolló, implementó y evaluó una aplicación Windows Form de escritorio basada en Realidad Aumentada para apoyar el proceso de aprendizaje de Ciencias Naturales, en los alumnos de octavo año de Educación General Básica de la Unidad Educativa "Tirso de Molina". La Interfaz Gráfica de Usuario (GUI) es robusta, y asegura que el usuario no pueda realizar acciones no soportadas de forma inadvertida; es decir, se ha construido una GUI, adaptada al dominio de la aplicación.
- El análisis cuantitativo de la situación tecnológica de la Realidad Aumentada destinada a la educación interactiva en la Unidad Educativa "Tirso de Molina"; se efectuó a partir de técnicas e instrumentos de investigación como la observación y la entrevista, dejando prever un desconocimiento parcial y muchas veces total, de la existencia de esta directriz aplicativa de la visión artificial y la Ingeniería en Sistemas Computacionales e Informática. Este establecimiento educativo a pesar de ser tangible a la tecnología informática ha rezagado la experiencia de la realidad aumentada, al igual que otras entidades de educación básica, educación media y educación superior en todo el Ecuador; la interactividad en las aulas del país, se ha limitado al uso de proyectores multimedia, reproducciones de audio y video; además de internet. El análisis cualitativo de la situación tecnológica de la realidad aumentada, se concertó con la investigación de indicios sobre visión artificial, caracterización de librerías destinadas al paradigma; algoritmos y seudo-algoritmos

SURF, OCR y Stitching; además de teorías, herramientas y procesos para el entrenamiento de una cascada de Haar; contextos necesarios para el autoaprendizaje sustancial de modelado digital, arte del marcador, programación en Visual C#, uso de las bibliotecas OpenCV, AForge.NET, EmguCV, u otras aplicaciones informáticas como: GIMP, AutoDesk 3DS Max, XNA, Visio, Project, Blender o Unity.

- El sistema PIVRA es capaz de capturar, detectar y reconocer de forma automática marcadores para Realidad Aumentada con un alto nivel de acierto y con ausencia total de marcas o perturbaciones en la imagen digital en movimiento; a partir del uso de las librerías OpenCV (entrenamiento cascada de Haar), reconocimiento de letras (EmguCV) y AForge.NET (reconocimiento de marcadores); las cuales implementan multitud de algoritmos de visión por computador, potentes en el tratamiento de imágenes y seguimiento de zonas de interés, haciendo que el framework de trabajo sea una herramienta eficiente para técnicas de Realidad Aumentada.
- La detección y el reconocimiento de letras del sistema PIVRA, en los casos con los que se ha experimentado, ha sido correcto; obteniendo una tasa de acierto del 95 %, en aproximadamente de 1 a 5 segundos en el tiempo de respuesta; esta condición es óptima para la percepción visual del video que se muestra, tratándose de una secuencia de fotogramas por segundo procesados con técnicas de visón artificial.
- El software PIVRA detecta y reconoce marcadores en blanco y negro de 6 cm x 6 cm a una distancia aproximada de entre 1,5 m a 3,5 m. El rendimiento del sistema PIVRA al ejecutar un ciclo completo de captura, procesamiento y visualización de Realidad Aumentada es menor a 1000 (mSeg) en proyecciones 2D y menor a (5 Seg) en proyecciones 3D parámetros eficaces e inherentes a la sensación de tiempo real.
- El sistema PIVRA permite interactuar con contenidos abstractos mediante una GUI tangible y natural para los usuarios. Los buenos resultados obtenidos con PIVRA hacen posible deducir la factibilidad, de generar una aplicación basada en Realidad Aumentada que sea atractiva para niños/as; que les permita aprender sin adicionar algún nivel de dificultad a su formación integral. La experiencia con la tecnología representa un desarrollo en la educación, deja en exhibición su potencial en áreas tan complejas como la pedagogía.
- Los principales problemas asociados a la implementación del prototipo PIVRA y su uso de marcadores, es el tracking e inclinación del marcador; el sistema atenuó dicho

inconveniente con su estructura algorítmica y sus sentencias de programación, no representaron mayor dificultad para los usuarios. Existe un rápido aprendizaje por parte de los docentes y alumnos respecto de cómo utilizar el software PIVRA y cómo experimentar con la tecnología congénita a Realidad Aumentada.

- Otros inconvenientes que se han encontrado para el desarrollo del sistema PIVRA es lidiar con la iluminación del entorno, la falta de resolución y el contraste o calidad del video en las zonas de interés; ya que a la vista del ojo humano, los colores y los objetos parecen no cambiar, pero al captar la imagen en movimiento por la cámara web; éstas variaciones son significativas. Dichos problemas se han conseguido disminuir mediante la conversión de imágenes, umbralización, filtrado, bordes y localización de la zona de interés sobre la que se delimita los cálculos.
- El software PIVRA es un sistema interactivo de Realidad Aumentada aplicado a objetos de aprendizaje, debido a que se pueden mostrar de forma dinámica los tópicos de Ciencias Naturales; transformando las clases a una forma inmersiva. El prototipo PIVRA puede trascender a niveles robóticos, con el reconocimiento de objetos a partir de un brazo robótico a la par con la Realidad Aumentada; además, al estar desarrollado bajo licencia de código abierto reconocida por la Open Source Iniciative (compatible con GPL), el software es distribuible bajo licencia GNU (General Public License), contención útil al contexto educacional progresivo y de investigación.
- El prototipo PIVRA en su versión 2.0 puede adherir mayores asignaturas del pensum de Educación Básica a efecto de concatenar los procesos de enseñanza y aprendizaje con modelos 3D totalmente realistas y de alta calidad; para virtuar la teoría con la práctica dentro de un mismo entorno.
- La visión artificial del sistema PIVRA cimentada en la detección y reconocimiento de letras y marcadores, en sus próximos prototipos puede involucrar algoritmos o módulos de audio digital; además del lenguaje de señas para una educación incluyente.

5.2. Recomendaciones

A continuación se presentan las principales recomendaciones efectuadas en el desarrollo y la implementación de la Pizarra Virtual usando Realidad Aumentada (PIVRA):

- Ejecutar el prototipo PIVRA implica tomar en cuenta las condiciones de iluminación, así como la nitidez de los marcadores para ser detectados y reconocidos por el sistema, a través de la cámara web; ya que de esto depende la calidad y rapidez del procesamiento y visualización de los recursos multimedia.
- La existencia de herramientas de desarrollo .NET, tal es el caso de Visual C#; agrupado a las librerías de visión por computadora como OpenCV, EmguCV y AForge.NET permiten la implementación de una software PIVRA amigable al docente y al alumno, de mucha portabilidad para facilitar la experiencia de la Realidad Aumentada en la educación de Ciencias Naturales.
- El entrenamiento de una cascada de Haar, es un proceso de alto coste y tiempo, razón que propone el uso de un computador de altas prestaciones para lograr un archivo .xml que permita la creación de algoritmos eficientes en el reconocimiento de objetos vía webcam, el sistema PIVRA se sometió a esta técnica logrando excelentes resultados para el procesamiento de imágenes en movimiento.
- El uso de librerías externas de licencia de código abierto, facilitaron el diseño y depuración de estructuras y algoritmos más complejos para el procesado de imágenes en movimiento, reconocimiento de marcadores y proyección de modelos 2D o 3D con técnicas de Realidad Aumentada del prototipo PIVRA.
- El proceso de visión artificial y reconocimiento de marcadores para Realidad Aumentada, consume en el sistema PIVRA una gran capacidad de procesamiento; esto demanda la utilización de un ordenador de excelentes prestaciones en memoria RAM, disco duro y microprocesador.
- El software PIVRA entra en la categoría de una aplicación reconocedora de patrones, pero existen otras opciones para el desarrollo de una herramienta que aplique el reconocimiento de marcadores, tales como redes neuronales y sistemas expertos.
- Utilizar la metodología de herramientas de modelado UML (Lenguaje Unificado de Modelado) y herramientas CASE (Ingeniería de Software Asistida por Computador) para constituir una aplicación coherente con la Ingeniería en Informática y Sistemas Computacionales.

BIBLIOGRAFÍA

[1] Esqueda, J., "Fundamentos de procesamiento de imágenes y Realidad Aumentada". CONACTEC, Instituto Tecnológico de Ciudad Madero, México, 2011, pp. 10-15.

[2] R. A. Saraguro Bravo, Tesis de Ingeniería en Sistemas Informáticos y Computación. "Implementación de una aplicación Android basada en Realidad Aumentada aplicada a puntos de interés de la UTPL", Universidad Técnica Particular de Loja, Ecuador, 2012, recuperado 07 de noviembre del 2014, de: http://dspace.utpl.edu.ec/bitstream/123456789/4939/1/Tesis_RodrigoSaraguro.pdf.

[3] J. S. Duque Álvarez, Tesis de Ingeniería en Sistemas y Telecomunicaciones. "Investigación y desarrollo de una aplicación en Realidad Aumentada para la empresa PLUGAR", Universidad Católica de Pereira, Facultad de Ciencias, Perú, 2011, recuperado 07 de noviembre del 2014, de: http://ribuc.ucp.edu.co:8080/jspui/bitstream/handle/10785/1022/CDPEIST68.pdf?sequence=1.

[4] C. Alcarria Izquierdo, Tesis de Ingeniería Informática. "Desarrollo de un sistema de Realidad Aumentada en dispositivos móviles", Universidad Politécnica de Valencia, Escuela Técnica Superior de Ingeniería Informática, España, 2010, recuperado 07 de noviembre del 2014, de: http://riunet.upv.es/bitstream/handle/10251/8597/PFC%20-%20Desarrollo%20de%20un%20sistema%20de%20Realidad%20Aumentada%20en%20dispositivos%20m%C3%B3viles.pdf

[5] C., Gonzáles Morcillo, D. Vallejo, J. Albusac, J. Castro, Realidad Aumentada, "Realidad Aumentada. Un Enfoque Práctico con ARToolKit y Blender", España, Ciudad Real, Primera Edición, Editorial Bubok Publishing S.L., 2012, pp. 4-20.

[6] B. Nievas, Realidad Aumentada, "La Realidad Aumentada", España, Madrid, Primera Edición, Editorial Hermida, 2011, pp. 10-25.

[7] G. Ros, G. Garc, Realidad Aumentada, "Realidad Aumentada basada en características naturales: Un enfoque práctico", España, Madrid, Primera Edición, Editorial Académica Española, 2012, pp. 7-15.

[8] ANOBIUM.ES., X. Basogain, M. Olabe., K. Espinosa., C. Roueche y J. Olabe (2012, Agosto) "Realidad Aumentada en la Educación: una tecnología emergente", Paper (2011), recuperado 6 de diciembre del 2014, de: http://www.anobium.es/docs/gc_fichas/doc/6CFJNSalrt.pdf.

[9] INFORMATICA2013.SLD.CU., J. Gaviria, G. Castaño, B. Rosero, J. Sierra. (2013, Enero). "Realidad Aumentada en el tratamiento de las enfermedades mentales y adiciones", recuperado 6 de diciembre del 2014, de: http://www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/viewFile/428/252.

[10] UNILIBREBAQ.EDU.CO., J. Marino e I. De León, (2012, Junio). "Uso de Realidad Aumentada para Enseñanza de Conceptos básicos de Física y Mecánica", Web-Site (2012), recuperado 6 de diciembre del 2014, de: http://www.unilibrebaq.edu.co/unilibrebaq/revistas2/index.php/ingeniare/article/viewFile/289/260.

[11] WORDPRESS.COM., M. Silva, R. Baños. "Educación Interactiva", recuperado 13 de diciembre del 2014, de: http://rosariobanos.files.wordpress.com/2010/06/educacion-interactiva1.doc

[12] L. Salgado, Grupo de Tratamiento de Imágenes. "Visión Artificial: Fundamentos y Aplicaciones", Universidad Politécnica de Madrid, E.T.S. Ingenieros de Telecomunicación, España, Madrid, 2013, recuperado 21 de marzo del 2015, de: http://arantxa.ii.uam.es/jms/seminarios_doctorado/abstracts2013/200503LSalagado.pdf

[13] M. José, Documento. "Visión Artificial", Universidad Pontificia Comillas, Escuela Técnica Superior de Ingeniería (ICAI), España, Madrid, 2009, recuperado 24 de marzo del 2015, de: http://www.etitudela.com/celula/downloads/visionartificial.pdf

[14] VIRTUALAMA.COM., R. Gómez. "Realidad Aumentada y las Pizarras virtuales", recuperado 13 de diciembre del 2014, de: http://www.virtualama.com/blog/realidad-aumentada-y-educacion/

[15] Azuma Ronald, T., "A Survey of Augmented Reality". Un estudio de la Realidad Aumentada. Estados Unidos, Malibu, Primera Edición, Editorial THOMSON, CA 90365, 2010, pp. 13-21.

[16] GENBETA.COM. "La empresa Metaido trae la Realidad Aumentada a los libros educativos", Web-Site (2011), recuperado 10 de noviembre del 2014, de: http://www.genbeta.com/metaio-trae-la-realidad-aumentada-a-los-libros.

[17] MADRIMASD.ORG. "Sistemas inteligentes, un contexto de la Realidad Aumentada", Blog (2011), recuperado 10 de noviembre del 2014, de: http://www.madrimasd.org/blogs/sistemas_inteligentes/2010/03/21/87063.

[18] REALIDAD VIRTUAL.COM. "Fundamentación teórica y estudio tecnológico sobre la Realidad Aumentada", RA (2010), recuperado 10 de noviembre del 2014, de: www.realidadvirtual.com.

[19] BUILDAR.CO. The Software of Augmented Reality. "El Software de Realidad Aumentada", BuildAR (2011), recuperado 07 de noviembre del 2014, de: http://www.buildar.co.nz/.

[20] GATECH.EDU. The Software of Augmented Reality. "El Software de Realidad Aumentada", DART (2012), recuperado 07 de noviembre del 2014, de: http://www.cc.gatech.edu/dart/.

[21] J. V. Rivadeneira Herrera, Tesis de Ingeniería en Electrónica y Telecomunicaciones. "Desarrollo de una aplicación de Realidad Aumentada, para Educación y Tele-Educación", Escuela Politécnica del Ejército, Departamento de Eléctrica y Electrónica, Ecuador, 2013, recuperado 07 de marzo del 2015, de: http://repositorio.espe.edu.ec/bitstream/21000/6684/1/T-ESPE-047214.pdf

[22] VIRTUALAMA.COM., R. Gómez. "Realidad Aumentada y las Pizarras virtuales", recuperado 13 de diciembre del 2014, de: http://www.virtualama.com/blog/realidad-aumentada-y-educacion/

[23] GARRIDO, R. y GARCÍA ALONSO, A. Técnicas de interacción para sistemas de realidad aumentada, Unidad de Construcción y Desarrollo del Territorio, LABEIN – Tecnalia, Parque Tecnológico de Bizkaia, 2010.

[24] A. Siracusa, Proyecto de Graduación. "Realidad Aumentada", Universidad de Palermo, Facultad de Diseño y Comunicación, Buenos Aires, Argentina, 2013, recuperado 08 de marzo del 2015, de: http://repositorio.espe.edu.ec/bitstream/21000/6684/1/T-ESPE-047214.pdf

[25] F. Rolando, Material de lectura. "De la realidad virtual a la realidad aumentada", Universidad de Palermo, Taller Perteneciente al Ciclo Open DC, Buenos Aires, Argentina, 2012, recuperado 13 de marzo del 2015, de: http://fido.palermo.edu/servicios_dyc/opendc/archivos/4674_open.pdf

[26] J. V. Rivadeneira Herrera, Tesis de Ingeniería en Electrónica y Telecomunicaciones. "Desarrollo de una aplicación de Realidad Aumentada, para Educación y Tele-Educación", Escuela Politécnica del Ejército, Departamento de Eléctrica y Electrónica, Ecuador, 2013, recuperado 07 de marzo del 2015, de: http://repositorio.espe.edu.ec/bitstream/21000/6684/1/T-ESPE-047214.pdf

[27] L. F. Bautista Moreta, Tesis de Ingeniería en Electrónica y Control. "Diseño e implementación de un sistema de tele operación para un robot móvil mediante reconocimiento de movimientos de la cabeza", Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Ecuador, Quito, 2014, recuperado 09 de marzo del 2015, de: http://bibdigital.epn.edu.ec/bitstream/15000/7363/1/CD-5512.pdf

[28] OPENCV.ORG, Manual de Usuario. "The OpenCV Reference Manual", recuperado 23 de marzo del 2015, de: http://docs.opencv.org/opencv2refman.pdf

[29] L. A. Aranda Barjola, Tesis de Ingeniería Industrial. "Sistema de seguridad basado en el cálculo de las trayectorias de los vehículos mediante cámara embarcada", Universidad Carlos III de Madrid, Escuela Politécnica Superior, Departamento de Ingeniería de Sistemas y Automática, España, Madrid, 2012, recuperado 18 de abril del 2015, de: http:// e-archivo.uc3m.es/Memoria%20PFC_LuisAlberto_Aranda_Barjola.pdf

[30] Shin Shi, Emgu CV Essentials, "Develop your own computer visión application using the power of Emgu CV - Desarrolle su propia aplicación de visión por computador utilizando el poder de Emgu CV", Reino Unido, Birmingham, Primera Edición, PACKT Publishing, 2013, pp. 11-99.

[31] AFORGENET.COM, Sitio Web Oficial. "Framework AForge.NET", recuperado 27 de abril del 2015, de: http://www.aforgenet.com/framework/

[32] J. P. Rodríguez Lomuscio, Tesis de Ingeniería Civil en Computación. "Realidad aumentada para el aprendizaje de Ciencias en niños de Educación General Básica", Universidad de Chile, Facultada de Ciencias Físicas y Matemáticas, Chile, 2011, recuperado 23 de febrero del 2015, de: http://repositorio.uchile.cl/tesis/uchile/2011/cf-rodriguez_jl/pdfAmont/cf-rodriguez_jl.pdf

[33] L. A. Aranda Barjola, Tesis de Ingeniería Industrial. "Sistema de seguridad basado en el cálculo de las trayectorias de los vehículos mediante cámara embarcada", Universidad Carlos III de Madrid, Escuela Politécnica Superior, Departamento de Ingeniería de Sistemas y Automática, España, Madrid, 2012, recuperado 18 de abril del 2015, de: http:// e-archivo.uc3m.es/Memoria%20PFC_LuisAlberto_Aranda_Barjola.pdf

[34] AFORGENET.COM, Sitio Web Oficial. "Framework AForge.NET", recuperado 20 de mayo del 2015, de: http://www.aforgenet.com/framework/

[35] J. P. Rodríguez Lomuscio, Tesis de Ingeniería Civil en Computación. "Realidad aumentada para el aprendizaje de Ciencias en niños de Educación General Básica", Universidad de Chile, Facultada de Ciencias Físicas y Matemáticas, Chile, 2011, recuperado 23 de febrero del 2015, de: http://repositorio.uchile.cl/tesis/uchile/2011/cf-rodriguez_jl/pdfAmont/cf-rodriguez_jl.pdf

ANEXOS

ÍNDICE DE ANEXOS

ANEXO A: Manual de Usuario PIVRA

ÍNDICE MANUAL DE USUARIO

1. INTRODUCCIÓN

La Pizarra Virtual usando Realidad Aumentada (PIVRA) en su entorno de herramienta educativa emplea, por un lado; técnicas de RA como tecnología base para dotar a la realidad, de contenidos digitales interactivos, y por otro lado; una pizarra y software de escritorio para la visualización e interacción con estos contenidos digitales. En este entorno se da la posibilidad de interactuar y preguntar al alumno sobre las cuestiones planteadas en los recursos digitales empleados; de esta forma no sólo la visualización y representación de éstos, ayuda a entender de una mejor manera aquello que se pretende enseñar sino que se refuerza la comprensión y el aprendizaje de una forma más amena y transparente.

PIVRA focaliza su operatividad en el análisis de una imagen mediante un sistema de visión artificial, para identificar objetos y regiones de interés dibujado o impreso por el usuario. El sistema PIVRA comprueba si existe algún dibujo u objeto relevante a detectar; si existiera algún dibujo u objeto, éste es cambiado por una imagen digital o modelo 3D almacenado en el sistema; a este dibujo u objeto se le conoce como marcador para RA. Un marcador para RA en el sistema PIVRA, es una imagen dibujada con un rotulador en una pizarra ordinaria de tiza líquida o una imagen impresa en una hoja; la característica distintiva de esta imagen dibujada o impresa, es la combinación de color blanco y negro; y que representa la marca de reconocimiento de PIVRA para la respectiva proyección del modelo 3D u imagen digital.

2. OBJETIVOS DEL MANUAL

- El objetivo primordial es ayudar a utilizar el sistema PIVRA obteniendo información de cada uno de los formularios que posee la aplicación, a manera de guía de usuario.
- Conocer cómo utilizar el sistema PIVRA mediante la descripción detallada e ilustrada de todas las opciones.

3. REQUISITOS DEL SISTEMA

El sistema funcionará sin problemas con las siguientes características mínimas:

Sistemas operativos admitidos:

- Microsoft Windows 8.1. SP1.
- Microsoft Windows 8.
- Microsoft Windows 7. SP1.

Procesador:

- Arquitecturas de 32 y 64 bits.

Memoria RA:

- Al menos 2GB para SO: Windows 7 SP1, Windows 8 y Windows 8.1 SP1.

Dispositivo gráfico:

- Dispositivo gráfico compatible con DirectX 9.
- Memoria gráfica de al menos 2 GB compartida o dedicada.

Dispositivos:

- Proyector multimedia.

Herramientas y suministros:

- Pizarra de marcador.
- Rotulador.

- Marcadores de realidad aumentada impresos o videos en el que contengan los marcadores.

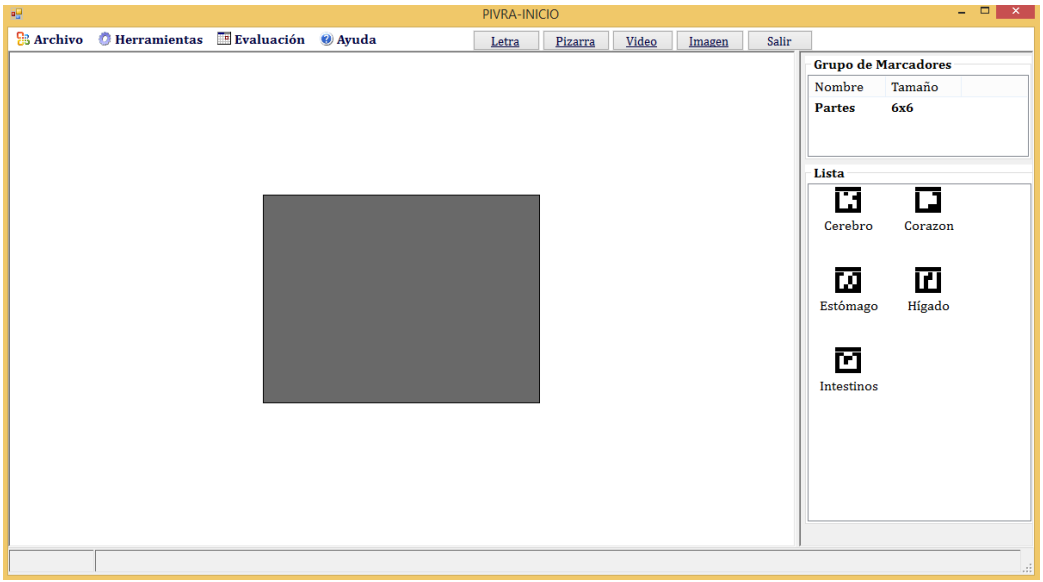
4. CONOCIMIENTOS PREVIOS

Los conocimientos mínimos que deben tener las personas que van a operar la aplicación y deberán utilizar este manual son:

- Conocimientos básicos acerca de Programas Utilitarios.
- Conocimientos básicos de aplicaciones Windows Forms.

5. SISTEMA PIVRA

Para ingresar al sistema, el usuario necesita ejecutar la aplicación, accediendo a la siguiente interfaz:



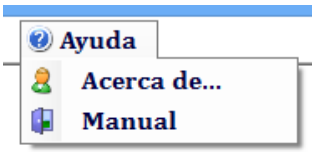
Interfaz. 1: Ventana inicio aplicación PIVRA

A continuación se describirá cada ítem del menú principal. Mismo que se compone de tres partes importantes como son: Archivo, herramientas y Ayuda.

Interfaz. 2: Sub-Ítems opción Archivo

Interfaz. 3: Sub-Ítems opción Herramientas

Interfaz. 4: Sub-Ítems opción Evaluación



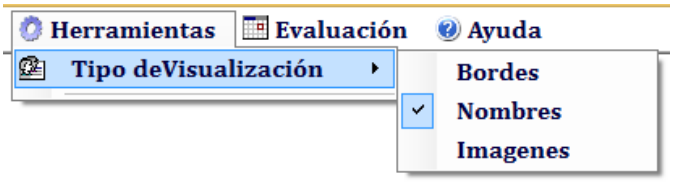
Interfaz. 5: Sub-Ítems opción Ayuda

Detección por cámara: Permite la captura directa en vivo de los marcadores.

Abrir video: Permite la opción de abrir un video ya estructurado, que contenga los marcadores respectivos.

Salir: Cierra el sistema y todas las ventanas.

Tipo de Visualización: Esta opción permite tres maneras de visualización de realidad aumentada, como son:

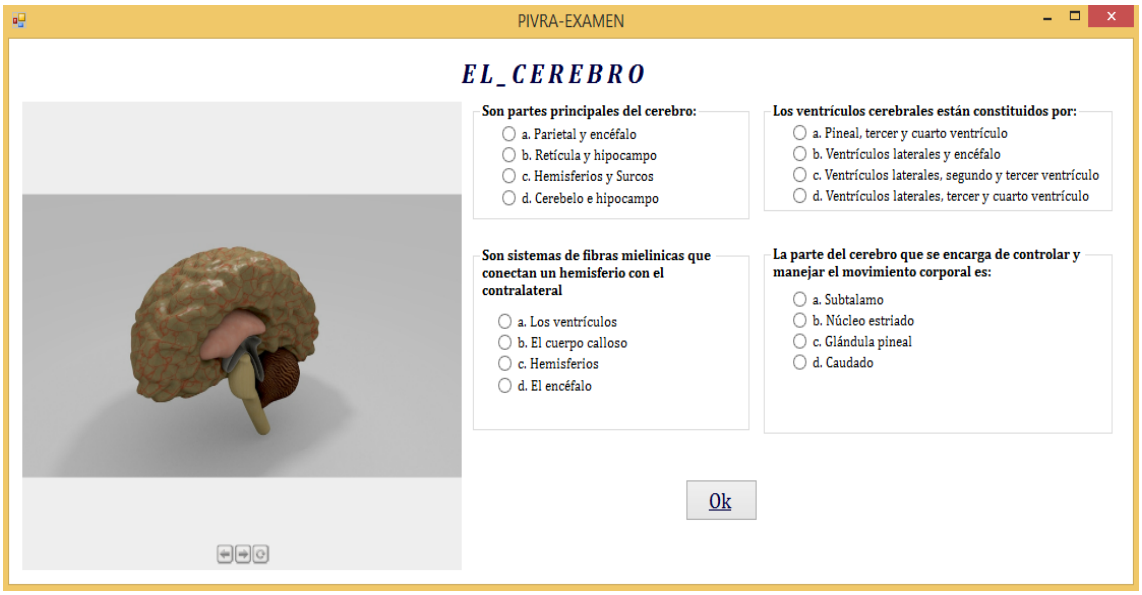


Interfaz. 6: Sub-Ítems opción Tipo de Visualización

- ***Bordes:*** Señala el borde en que se encuentra el marcador.
- ***Nombres:*** Coloca el nombre al que pertenece el marcador.
- ***Imágenes:*** Transforma de lo real a 2D.
- ***Modelo 3D:*** Transforma de lo real a 3D.

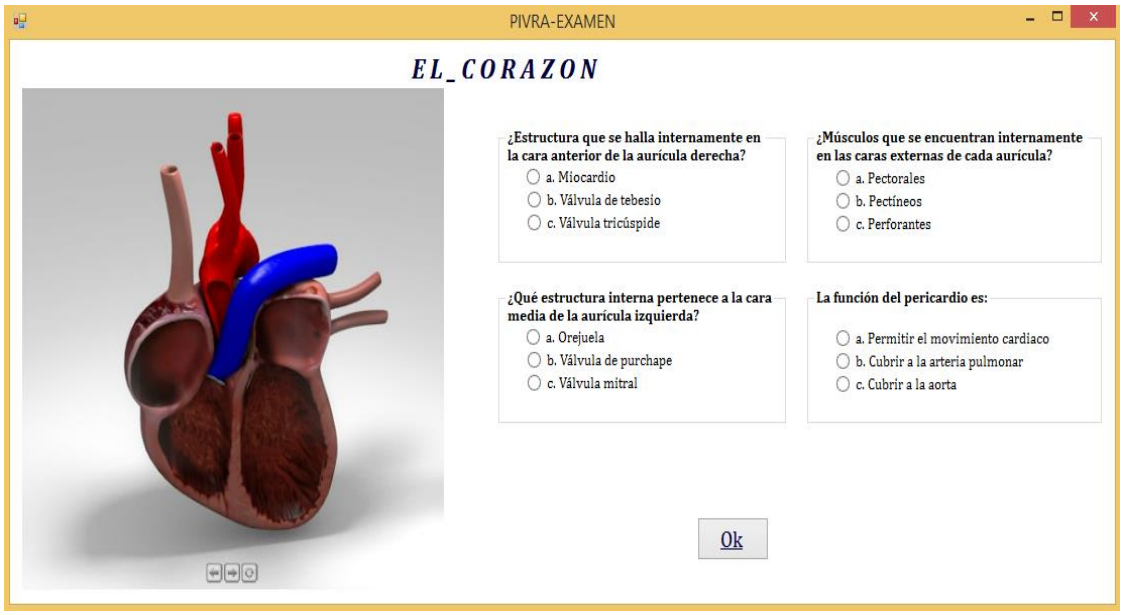
Evaluacion: Evalúa el Aprendizaje del estudiante.

Cerebro:



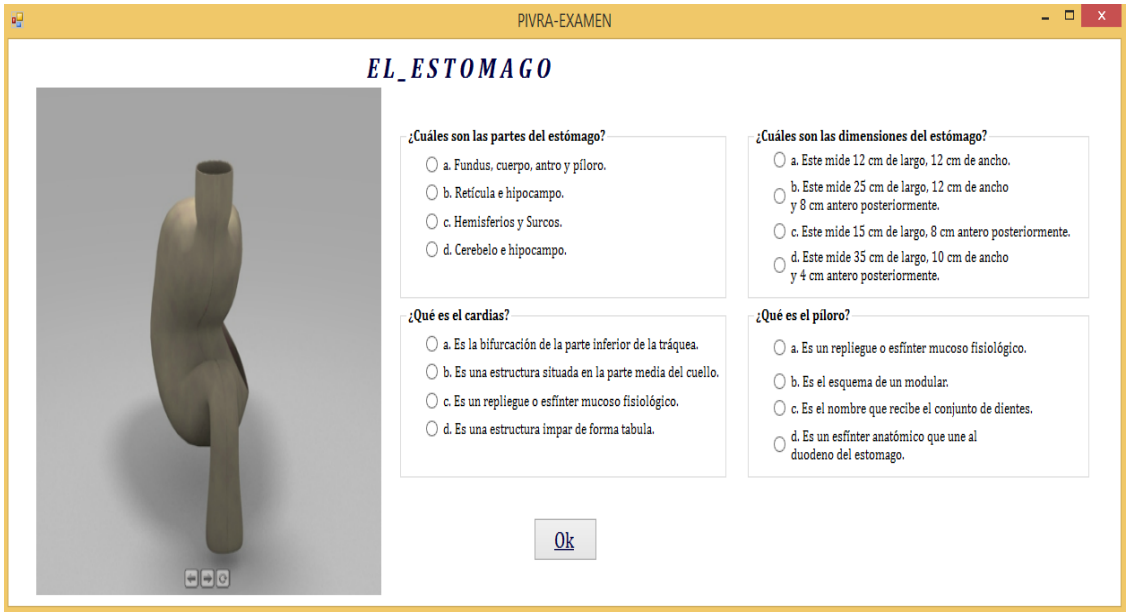
Interfaz. 7: Formulario evaluación Cerebro

Corazón:



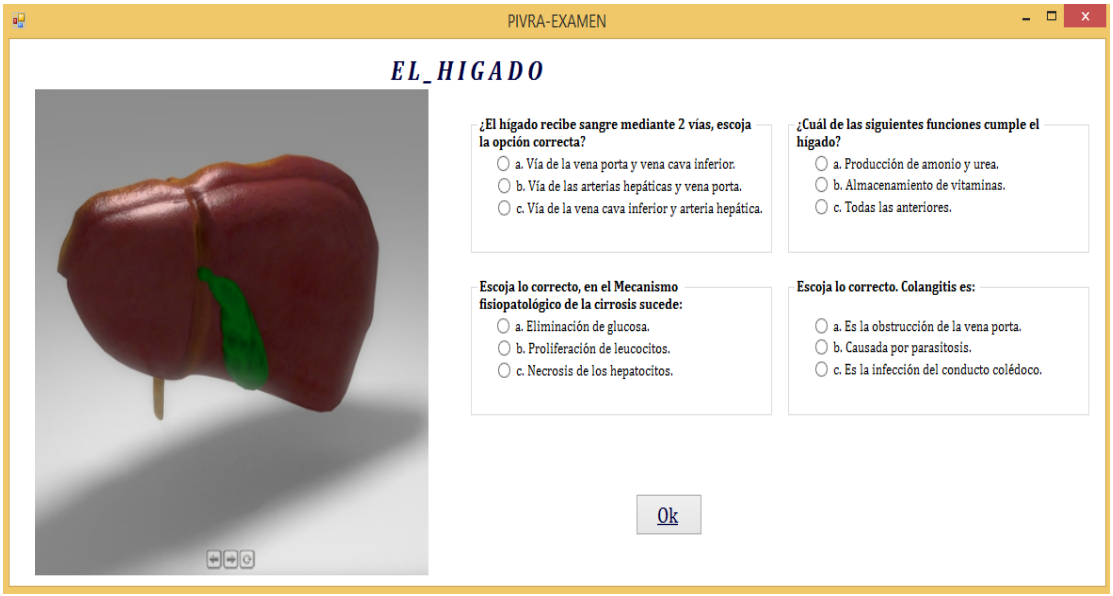
Interfaz. 8: Formulario evaluación Corazón

Estómago:



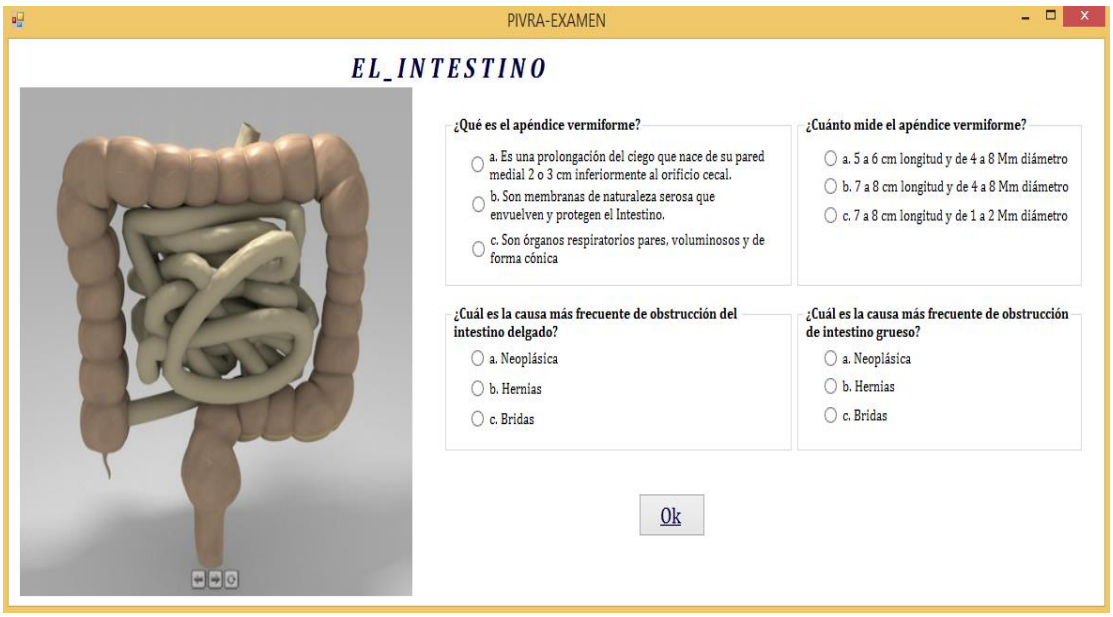
Interfaz. 9: Formulario evaluación Estómago

Hígado:



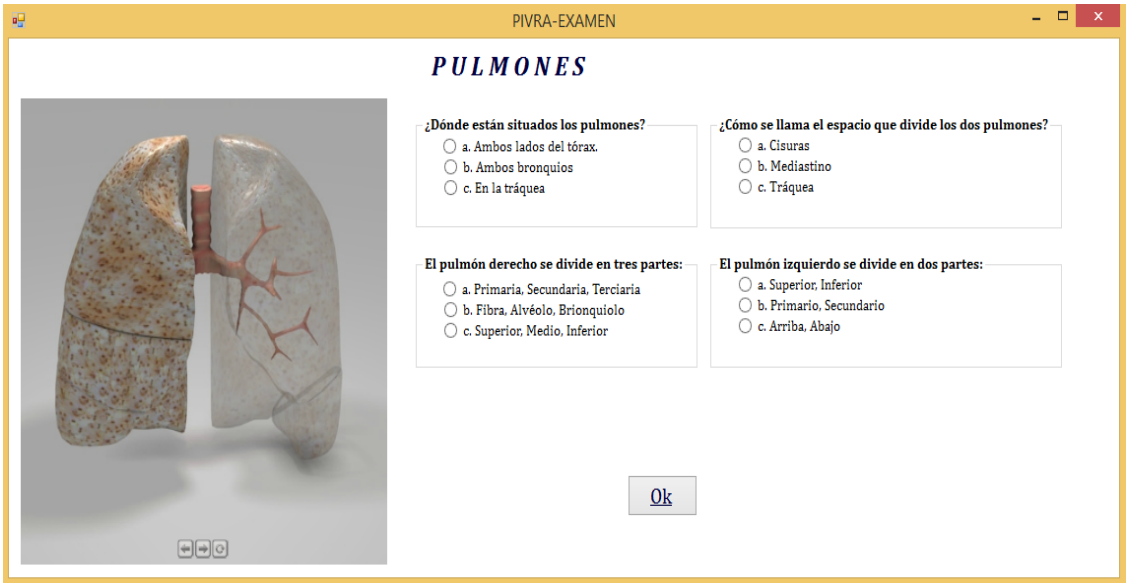
Interfaz. 10: Formulario evaluación Hígado

Intestino:



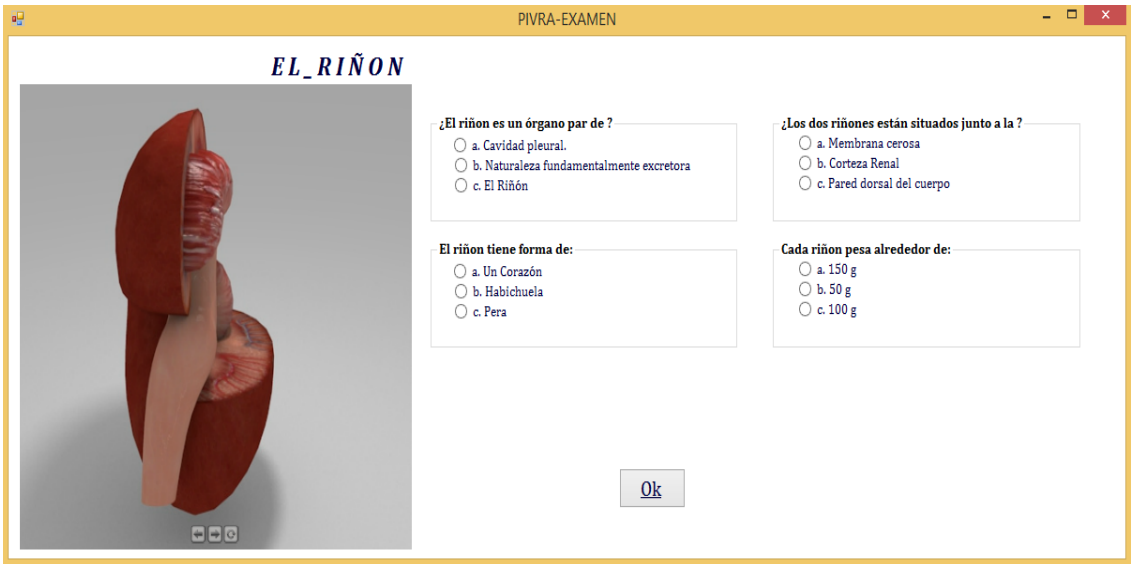
Interfaz. 11: Formulario evaluación Intestino

Pulmones:



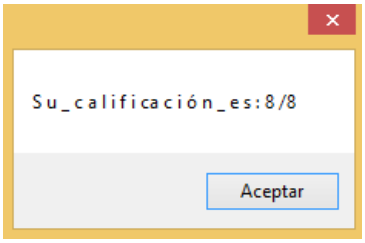
Interfaz. 12: Formulario evaluación Pulmones

Riñón:



Interfaz. 13: Formulario evaluación Riñón

Botón Ok: Finaliza el cuestionario, dando el resultado final sobre 8.

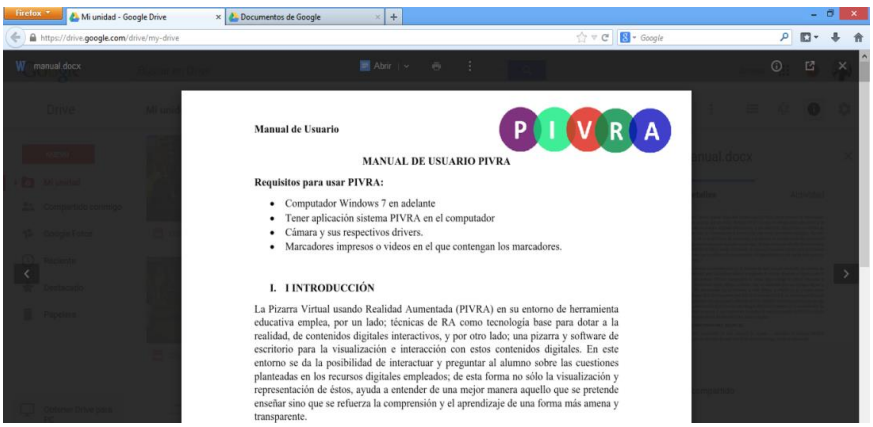


Interfaz. 14: Mensaje Evaluación

Acerca de: Muestra la autoría del sistema.

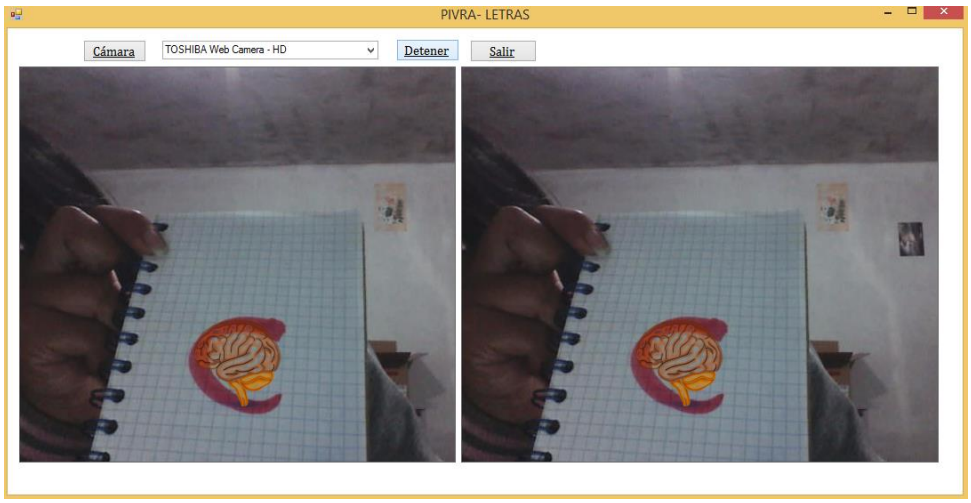
Interfaz. 15: Autoría aplicación PIVRA

Manual: Permite ingresar al link donde estará el manual de usuario.



Interfaz. 16: Manual de usuario PIVRA

Botón Letra: Abre el formulario para ***Detección por letra*.**



Interfaz. 17: Detección de letra

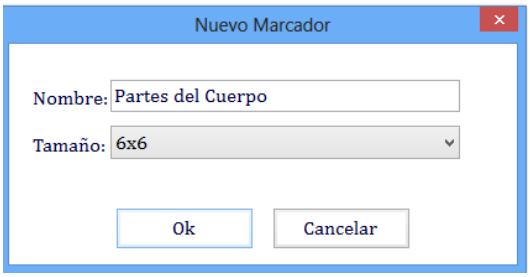
Botón Pizarra: Acceso directo a la opción de ***Detección por cámara.***

Botón Video: Acceso directo a ***Abrir video.***

Botón Imagen: Acceso Directo a ***Imágenes.***

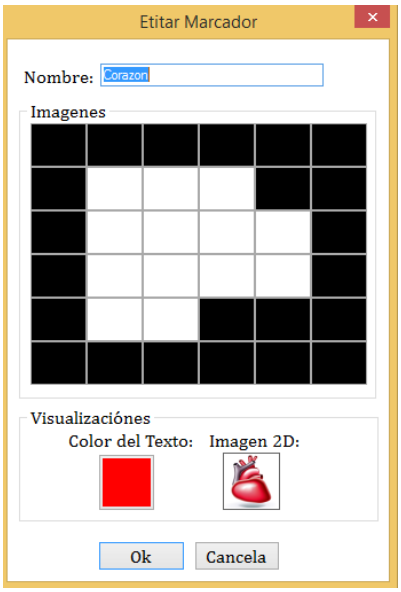
Creación de un grupo de marcadores:

Primero, es dar un nombre a todo este grupo, con un tamaño de 6x6 ya que de esta forma están diseñados los marcadores existentes:



Interfaz.18: Creación grupo de marcadores

Luego se creará marcador a marcador, en este caso, como ejemplo; se va a realizar el diseño de un marcador que muestre el corazón:

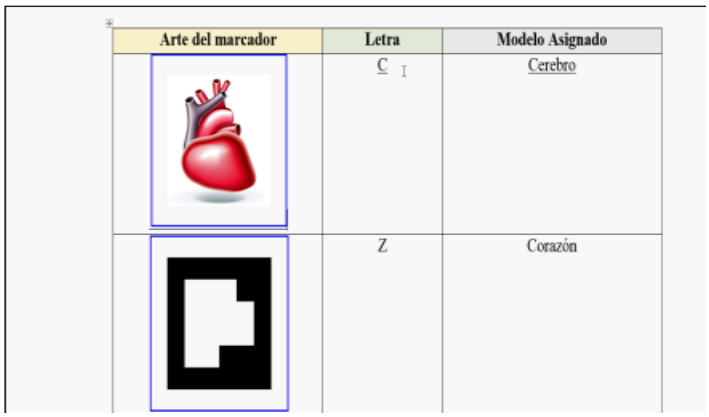


Interfaz. 19: Creación de marcador

Interfaz. 20: Selección Imágenes 2D

Aquí se muestra, como el sistema transforma lo real en realidad aumentada.

2D:



Interfaz. 21: Visualización 2D

ANEXO B: Estudio de Usabilidad

ÍNDICE ESTUDIO DE USABILIDAD

1. INTRODUCCIÓN

El estudio de usabilidad permite determinar la cualidad de un sistema o software respecto a su facilidad de uso, su facilidad de aprendizaje y la satisfacción del usuario final. Para realizar la evaluación de usabilidad de la Pizarra Virtual usando Realidad Aumentada (PIVRA), es imperante definir el término usabilidad y los argumentos de la ingeniería de usabilidad.

1.1. Usabilidad

Es un atributo intangible del software, por lo tanto; es difícil de visualizar, medir y reconocer como un factor determinante de su calidad. **[35]**

1.2. Ingeniería de Usabilidad

La Ingeniería de Usabilidad (IU) se puede definir como un conjunto de técnicas para el desarrollo de sistemas en la que se especifican previamente niveles cuantitativos de usabilidad, y el sistema se construye para alcanzar dichos niveles, que se conocen como métricas. La IU es una propuesta tecnológica para contribuir a la evaluación de la usabilidad de un software o sistema informático. Existe diversidad de cuestionarios métricos para usabilidad, tres de los principales se detallan a continuación:

- ***WAMMI:*** Siglas en inglés de Website Analysis and Measure Ment Inventory. Consiste en un servicio de análisis web para ayudar a cumplir objetivos digitales mediante la medición y análisis de la experiencia de usuario web. **[35]**
- ***SUMI:*** Siglas en inglés de Software Usability Measuring Inventory. Es una herramienta subjetiva que mide la satisfacción y percepción del usuario. **[35]**
- ***QUIS:*** Siglas en inglés de Questionnaire for User Interaction Satisfaction. Es una herramienta de evaluación de usabilidad centrada en el usuario para sistemas de computación interactiva. **[35]**

2. EVALUACIÓN DE USABILIDAD DEL SISTEMA PIVRA

Al evaluar la usabilidad del sistema PIVRA, se aplicaron métodos e instrumentos de investigación dedicados a la cuantificación y cualificación de un sistema informático por parte del usuario. La evaluación se basó en la observación de la forma en que el docente y los alumnos trabajan con el sistema PIVRA. Además, se aplicó un cuestionario métrico de usabilidad para conocer la impresión de los usuarios respeto a la manipulación del sistema PIVRA y su experiencia con la Realidad Aumentada. Al iniciar la evaluación de usabilidad fue necesario conocer los participantes a los cuales se aplicó la metodología de usabilidad.

2.1. Participantes

La respectiva evaluación de usabilidad fue realizada con los docentes de Ciencias Naturales y sus respectivos alumnos del Octavo Año de Educación General Básica de la Unidad Educativa "Tirso de Molina". La tabla 1 muestra el registro de usuarios finales del sistema PIVRA.

Tabla 1: Usuarios finales del sistema PIVRA

USUARIOS FINALES	CANTIDAD
Docentes de Ciencias Naturales	7
Octavo Año de Educación Básica "A"	32
Octavo Año de Educación Básica "B"	32
Octavo Año de Educación Básica "C"	31
Octavo Año de Educación Básica "D"	31
SUBTOTAL	133
MEDIA ($\bar{X}$)	26,6
TOTAL	**27**

Fuente. Unidad Educativa "Tirso de Molina"
Elaborado por. Jimena Caguana

Para la evaluación de usabilidad se contó con 27 usuarios finales. La distribución por géneros fue de 14 hombres y 13 mujeres. La distribución etaria fue de 17 usuarios de 13

años, 3 de 14 años, 2 de 34 años, 3 de 37 años y 2 de 38 años respectivamente. Los usuarios nunca antes habían usado, visto o recibido comentarios del sistema PIVRA y ninguno de ellos afirmó haber visto una aplicación de Realidad Aumentada con anterioridad. El número de usuarios es adecuado para aplicar la metodología planteada para la evaluación de usabilidad del software.

2.2. Métodos e Instrumentos

Observación: Este método consiste en observar al usuario interactuar con el sistema PIVRA en su ambiente de trabajo. Esto es de suma importancia para la usabilidad, ya que permite determinar información in situ del uso de la aplicación. Por otro lado; observar la forma en que los usuarios realizan ciertas tareas entrega una gran cantidad de información respecto de sus modelos de trabajo mental, estrategias individuales de resolución de problemas y decisiones subyacentes a la tarea en particular. Se identifican los siguientes puntos como claves para la interacción:

- Falla la interacción del sistema, el sistema no responde con la acción deseada.
- Número de veces que el usuario obstruye accidentalmente el marcador.
- Número de veces en que el sistema falla en detectar el marcador.
- El usuario deteriora el marcador.

NOTA: Una técnica que es comúnmente usada en el método de observación es la de grabar en video la sesión de trabajo, para revisar posteriormente la interacción y obtener detalles que no fueron captados en el momento de la evaluación. Durante la evaluación realizada se grabó a los usuarios y su interacción a través de un grabador de pantalla.

Cuestionario: Una gran cantidad de aspectos de la usabilidad de un sistema pueden ser estudiados simplemente preguntando a los usuarios. Esto es trascendental cuando se busca medir la satisfacción subjetiva del usuario y sus impresiones acerca de

la interfaz de la aplicación. Una forma de consultar a los usuarios sobre estos aspectos son los principales cuestionarios métricos para usabilidad.

Para el sistema PIVRA se realizó una evaluación de usabilidad de usuario final, adaptando el cuestionario métrico para usabilidad QUIS "Questionnaire for User Interaction Satisfaction"; a las necesidades de la aplicación. Las modificaciones fueron realizadas con el apoyo de una psicóloga con experiencia en la creación y adaptación de cuestionarios, para mantener dentro de lo posible la validez del instrumento. La escala de valoración es de 0 a 9.

2.3. Contexto

La evaluación de usabilidad se trabajó en la Unidad Educativa "Tirso de Molina". Se contó con un ambiente controlado, sin demasiado ruido ni distracciones para los usuarios y con luz moderada para mejorar el funcionamiento de la aplicación. Se trabajó sobre un escritorio con espacio suficiente para interactuar con todos los elementos necesarios y la configuración del entorno de trabajo contó de un notebook, una cámara web externa montada en la parte superior de la pantalla del notebook, un proyector multimedia y 10 marcadores impresos sobre un material rígido y durable.

2.4. Cuestionario de evaluación de usabilidad del Sistema PIVRA

En este apartado del Trabajo de Graduación, se muestra el cuestionario métrico para usabilidad aplicado a los usuarios del sistema PIVRA en la Unidad Educativa "Tirso de Molina".

Indicación: Por favor, marque con una (X) las casillas que mejor represente su opinión (grado de acuerdo) sobre las características del software de la Pizarra Virtual usando Realidad Aumentada, de acuerdo a lo que haya experimentado durante el uso de este sistema.

La tabla 2 muestra el cuestionario de evaluación de usabilidad del sistema PIVRA.

Tabla 2: Cuestionario de evaluación de usabilidad del sistema PIVRA

USABILIDAD DEL SISTEMA PIVRA													
INTERROGANTE	**En desacuerdo**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**De acuerdo**
REACCIÓN GENERAL DEL SOFTWARE													
1.- El software es de su agrado	Nada											Suficiente
2.- El software es entretenido	Nada											Suficiente
3.- El software es recomendable a otros niños o jóvenes	Nada											Suficiente
4.- El software le dejó algún aprendizaje	Nada											Suficiente
5.- El software es interactivo	Nada											Suficiente
6.- El software es fácil de utilizar	Nada											Suficiente
7.- El software es motivador	Nada											Suficiente
8.- El software puede ser implementado en la vida cotidiana	No procede											Procede
PANTALLA													
9.- Las imágenes, colores y brillos de la pantalla	No comunica											Comunica
10.- Lectura de caracteres en la pantalla	Dificultoso											Fácil
11.- Organización de la información	Confunde											Claro
TERMINOLOGÍA Y SISTEMA DE INFORMACIÓN													
12.- Terminología relacionada con la tarea	Nunca											Siempre
13.- Secuencia de pantallas	Confunde											Claro
APRENDIZAJE													
14.- Aprender a operar el sistema	Dificultoso											Fácil
15.- Realización de tareas es sencilla	Nunca											Siempre
16.- Los modelos 3D del software	No comunica											Comunica
17.- Experiencia de Realidad Aumentada	Mala											Excelente
SISTEMA DE CAPACIDADES													
18.- Los modelos 3D del software	No identificables											Identificables
19.- Sistema Velocidad	Lento											Rápido
20.- Sistema fiabilidad	Poco fiable											Seguro

Fuente: Cuestionario QUIS – Sistema PIVRA
Elaborado por: Jimena Caguana

3. RESULTADOS USABILIDAD DEL SISTEMA PIVRA

3.1. Observación evaluativa del uso del sistema PIVRA

La tabla 3 detalla la guía de observación con la respectiva evaluación del uso del sistema PIVRA.

Tabla 3: Guía de observación para evaluación del sistema PIVRA

INT	USUARIO	FALLA LA INTERACCIÓN	OBSTRUYE ACCIDENTALMENTE EL MARCADOR	FALLA EN DETECCIÓN DEL MARCADOR	DETERIORO DEL MARCADOR
1	I	✓	✓	✓	✓
2	D	✓	✓	✓	✓
3	D	✓	✓	✓	✓
4	D	✓	✓	✓	✓
5	A	✓	✓	✓	✓
6	A	✓	×	✓	✓
7	A	✓	✓	×	✓
8	A	✓	×	✓	✓
9	A	✓	✓	✓	✓
10	A	✓	✓	✓	✓

A = Alumno/a. **D** = Docente. **I** = Investigadora.

Elaborado por: Jimena Caguana

La tabla 4 muestra los resultados obtenidos en la observación del uso del sistema PIVRA.

Tabla 4: Resultados obtenidos en la observación evaluativa del uso del sistema PIVRA

PUNTOS DE INTERACCIÓN	DIEZ INTERACCIONES	PORCENTAJE DE EFECTIVIDAD
Falla la interacción del sistema, el sistema no responde con la acción deseada	0	100%
Número de veces que el usuario obstruye accidentalmente el marcador	2	80%
Número de veces en que el sistema falla en detectar el marcador	1	90%
El usuario deteriora el marcador	0	100%

Elaborado por: Jimena Caguana

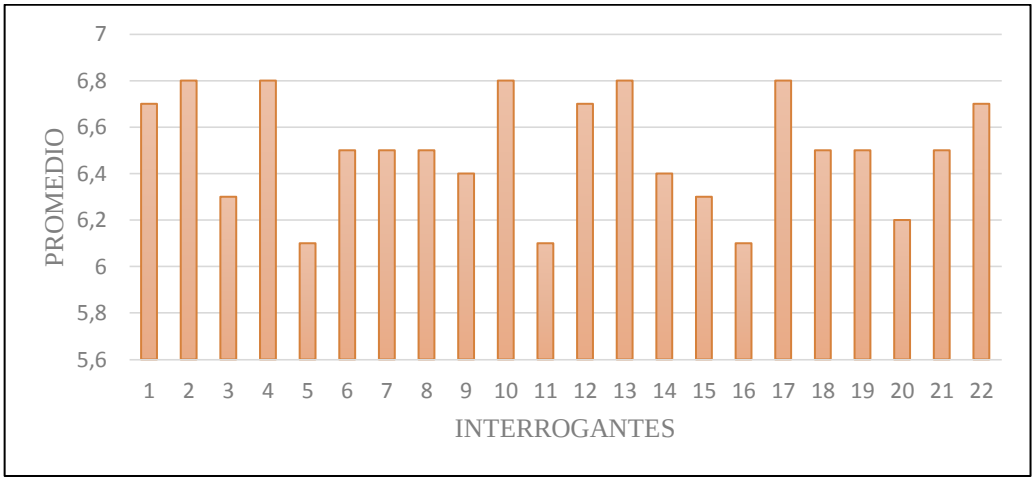
NOTA: La observación evaluativa del uso del sistema PIVRA permitió determinar que el mayor inconveniente en el uso del sistema PIVRA es la obstrucción accidental del marcador frente a la detección y reconocimiento de software, provocando el desvanecimiento instantáneo de el modelo 2D.

3.2. Cuestionario de evaluación de usabilidad del sistema PIVRA

USABILIDAD DEL SISTEMA PIVRA											
USUARIO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**PRM.**
GÉNERO	**F**	**F**	**M**	**M**	**M**	**F**	**F**	**F**	**M**	**M**	
REACCIÓN GENERAL AL SOFTWARE											
1.- Le gusta el software	7	7	5	6	7	6	7	7	7	8	**6,7**
2.- El software es entretenido	7	7	6	5	8	6	7	7	7	8	**6,8**
3.- Recomendaría este software a otros niños o jóvenes	7	7	7	8	8	6	6	6	5	3	**6,3**
4.- Aprendió con este software	8	5	6	7	7	7	7	6	7	8	**6,8**
5.- El software es interactivo	6	6	6	7	7	6	5	5	6	7	**6,1**
6.- El software es fácil de utilizar	5	7	7	7	5	8	6	6	8	6	**6,5**
7.- El software es motivador	8	8	6	6	6	7	5	7	5	7	**6,5**
8.- El software puede ser implementado en la vida cotidiana	7	7	7	6	7	7	7	6	6	5	**6,5**
PANTALLA											
9.- Las imágenes, colores y brillos de la pantalla transmiten información	7	6	7	5	7	7	5	7	6	7	**6,4**
10.- Lectura caracteres en la pantalla	8	6	7	7	7	5	6	8	7	7	**6,8**
11.- Organización de la información	6	6	6	5	6	5	7	8	5	7	**6,1**
12.- Secuencia de pantallas	6	7	7	7	8	5	6	6	7	8	**6,7**
TERMINOLOGÍA Y SISTEMA DE INFORMACIÓN											
13.- El uso de términos en todo el sistema	7	8	7	7	6	7	8	6	6	6	**6,8**
14.- Terminología relacionada con la tarea	6	5	5	5	8	7	7	7	8	6	**6,4**
15.- Posición de los mensajes en pantalla	7	6	6	8	4	7	6	7	5	7	**6,3**
APRENDIZAJE											
16.- Aprender a operar el sistema	7	7	5	5	8	5	5	6	6	7	**6,1**
17.- Recordar nombres y el uso de los comandos	6	7	6	7	7	8	9	5	6	7	**6,8**
18.- Realización de tareas es sencilla	7	7	7	7	5	6	6	8	4	8	**6,5**
19.- Los modelos 3D del software me transmiten información	7	6	6	5	5	8	7	7	8	6	**6,5**
SISTEMA DE CAPACIDADES											
20.- Los modelos 3D del software son claramente identificables	6	7	5	7	6	5	7	7	7	5	**6,2**
21.- Sistema Velocidad	5	7	8	6	5	7	6	8	6	7	**6,5**
22.- Sistema Fiabilidad	7	9	7	7	6	5	4	8	7	7	**6,7**
PROMEDIO POR USUARIO	6,7	6,4	6,3	6,4	6,5	6,4	6,3	6,7	6,3	6,7	**6,5**

Elaborado por: Jimena Caguana

NOTA: El promedio general obtenido en la evaluación de usabilidad del sistema PIVRA es de **6,5 puntos**; lo que significa que la mayoría de los usuarios finales aprecian el desempeño y la interactividad del sistema PIVRA en los procesos de enseñanza y aprendizaje para la asignatura de Ciencias Naturales, en la Unidad Educativa "Tirso de Molina". Los usuarios encontraron agradable e interesante la aplicación y la experiencia con la Realidad Aumentada. A continuación, la gráfica muestra los resultados de la evaluación de usabilidad del sistema PIVRA.



Elaborado por: Jimena Caguana

Del gráfico se puede observar que dieciocho de las veintidós afirmaciones evaluadas obtuvieron puntajes de 6,2 sobre 10; además, cinco interrogantes se encuentran en un puntaje de 6,8; lo cual indica que la satisfacción y aceptación del software, es alta entre los usuarios finales. A partir del cuestionario de evaluación de usabilidad del sistema PIVRA modificado utilizado en la evaluación, es estadísticamente posible considerar cinco índices; reacción general al software, pantalla, terminología y sistema de información, aprendizaje y sistema de capacidades. A continuación, en la tabla 5; se especifica el subconjunto de afirmaciones que están asociadas a cada índice.

Tabla 5: Índices de evaluación de usabilidad del sistema PIVRA

ÍNDICES	INTERROGANTES								PROMEDIO
Reacción general al software	1	2	3	4	5	6	7	8	6,53
	6,7	6,8	6,3	6,8	6,1	6,5	6,5	6,5	
Pantalla	9		10		11		12		6,5
	6,4		6,8		6,1		6,7		
Terminología y sistema de información	13		14		15				6,5
	6,8		6,4		6,3				
Aprendizaje	16		17		18		19		6,47
	6,1		6,8		6,5		6,5		
Sistema de capacidades	20		21		22				6,46
	6,2		6,5		6,7				

Elaborado por: Jimena Caguana

Al analizar los cinco índices se puede observar que los puntajes obtenidos en cada uno de ellos son altos. En todos los casos el puntaje es 6,5. También es posible observar que no existe una diferencia significativa en el puntaje obtenido por la aplicación entre hombres y mujeres, la tabla 6 muestra esta asimilación.

Tabla 6: Evaluación de usabilidad del sistema PIVRA por género

ÍNDICE	GÉNERO	N	MEDIA
Reacción general al software	F	5	6,55
	M	5	6,5
Pantalla	F	5	6,35
	M	5	6,65
Terminología y sistema de información	F	5	6,73
	M	5	6,27
Aprendizaje	F	5	6,7
	M	5	6,25
Sistema de capacidades	F	5	6,53
	M	5	6,4

Elaborado por: Jimena Caguana

Se puede observar que ambos géneros dieron un promedio general superior a 6,25 en todos los índices de evaluación de usabilidad. Este resultado muestra que la aplicación no tiene una tendencia hacia uno u otro tipo de usuario. Esto es atribuible a que durante el diseño del sistema PIVRA se intentó crear una aplicación que fuera neutra, en el sentido de que no incorpora elementos que pudieran ser más llamativos para un género que para otro.

ANEXO C: Código de Programación

ÍNDICE CÓDIGO DE PROGRAMACIÓN

1. INTRODUCCIÓN

El código de programación utilizado en el sistema PIVRA para la consecución de sus objetivos; los cuales; principalmente son: ser un software informática de escritorio con técnicas de Visión Artificial y Realidad Aumentada, que promulgue el uso de esta tecnología en los procesos de enseñanza y aprendizaje, principalmente para el área de Ciencias Naturales.

2. FORMULARIO PRINCIPAL

La programación de la interfaz principal es la siguiente:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Printing;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;
using AForge.Math;
using AForge.Video;
using AForge.Video.DirectShow;
using AForge.Vision.GlyphRecognition;
using AForge.Imaging;
using AForge.Imaging.Filters;

using Xna3DViewer;

namespace PIVRA
{
    public partial class Inicio : Form
    {
        private GlyphDatabases glyphDatabases = new GlyphDatabases( );
        private string activeGlyphDatabaseName = null;
        private GlyphDatabase activeGlyphDatabase = null;
        private Stopwatch stopWatch = null;

        private ImageList glyphsImageList = new ImageList( );

        private        AugmentedRealityForm        arForm        =        null;
```

```
        private GlyphImageProcessor imageProcessor = new GlyphImageProcessor( );
        private bool autoDetectFocalLength = true;

        private object sync = new object( );

        private const string ErrorBoxTitle = "Error";

        #region Configuration Option Names
        private const string activeDatabaseOption = "ActiveDatabase";
        private const string mainFormXOption = "MainFormX";
        private const string mainFormYOption = "MainFormY";
        private const string mainFormWidthOption = "MainFormWidth";
        private const string mainFormHeightOption = "MainFormHeight";
        private const string mainFormStateOption = "MainFormState";
        private const string mainSplitterOption = "MainSplitter";
        private const string glyphSizeOption = "GlyphSize";
        private const string focalLengthOption = "FocalLength";
        private const string detectFocalLengthOption = "DetectFocalLength";
        private       const       string       autoDetectFocalLengthOption       =
"AutoDetectFocalLength";
        #endregion

        public Inicio( )
        {
            InitializeComponent( );
            glyphsImageList.ImageSize = new Size( 32, 32 );
            lstvMarcadores.LargeImageList = glyphsImageList;

            itmBordes.Tag = VisualizationType.BorderOnly;
            itmNombres.Tag   = VisualizationType.Name;
            itmImagenes.Tag  = VisualizationType.Image;
            itm3D.Tag    = VisualizationType.Model;
        }

        private void ShowErrorBox( string message )
        {
            MessageBox.Show(    message,    ErrorBoxTitle,    MessageBoxButtons.OK,
MessageBoxIcon.Error );
        }

        private void OpenVideoSource( IVideoSource source )
        {

            this.Cursor = Cursors.WaitCursor;
            imageProcessor.Reset( );

            vspResultado.SignalToStop( );
            //videoSourcePlayer3.SignalToStop();
            vspResultado.WaitForStop( );
            //videoSourcePlayer3.WaitForStop();

            vspResultado.VideoSource    =    new    AsyncVideoSource(    source    );
```

```
            vspResultado.Start( );
            //videoSourcePlayer3.VideoSource = new AsyncVideoSource(source);
            //videoSourcePlayer3.Start();
            stopWatch = null;

            timer.Start( );

            this.Cursor = Cursors.Default;
        }

        private void timer_Tick( object sender, EventArgs e )
        {
            IVideoSource videoSource = vspResultado.VideoSource;

            if ( videoSource != null )
            {
                int framesReceived = videoSource.FramesReceived;

                if ( stopWatch == null )
                {
                    stopWatch = new Stopwatch( );
                    stopWatch.Start( );
                }
                else
                {
                    stopWatch.Stop( );

                    float fps = 1000.0f * framesReceived /
stopWatch.ElapsedMilliseconds;
                    fpsLabel.Text = fps.ToString( "F2" ) + " fps";

                    stopWatch.Reset( );
                    stopWatch.Start( );
                }
            }
        }

        private void newToolStripMenuItem_Click( object sender, EventArgs e )
        {
            NuevoMarcador newCollectionForm = new NuevoMarcador(
glyphDatabases.GetDatabaseNames( ) );

            if ( newCollectionForm.ShowDialog( ) == DialogResult.OK )
            {
                string name = newCollectionForm.CollectionName;
                int size = newCollectionForm.GlyphSize;

                GlyphDatabase db = new GlyphDatabase( size );

                try
                {
                    glyphDatabases.AddGlyphDatabase( name, db );
```

```
                    ListViewItem lvi = lstvListaMarcadores.Items.Add( name );
                    lvi.SubItems.Add( string.Format( "{0}x{1}", size, size ) );
                    lvi.Name = name;
                }
                catch
                {
                    ShowErrorBox( string.Format( "Ya existe una Base de datos
Marcador con el nombre '{0}' .", name ) );
                }
            }
        }

        private void glyphCollectionsContextMenu_Opening( object sender,
CancelEventArgs e )
        {
            activateToolStripMenuItem.Enabled =
            renameToolStripMenuItem.Enabled =
            deleteToolStripMenuItem.Enabled = (
lstvListaMarcadores.SelectedIndices.Count != 0 );
        }

        private void glyphCollectionContextMenu_Opening( object sender,
CancelEventArgs e )
        {
            newGlyphToolStripMenuItem.Enabled = ( activeGlyphDatabase != null );
            editGlyphToolStripMenuItem.Enabled =
            deleteGlyphToolStripMenuItem.Enabled =
            printPreviewToolStripMenuItem.Enabled =
            printToolStripMenuItem.Enabled = ( activeGlyphDatabase != null ) && (
lstvMarcadores.SelectedIndices.Count != 0 );
        }

        private void newGlyphToolStripMenuItem_Click( object sender, EventArgs e
)
        {
            if ( activeGlyphDatabase != null )
            {

                Glyph glyph = new Glyph( string.Empty, activeGlyphDatabase.Size
);
                glyphNameInEditor = string.Empty;

                EditarMarcador glyphForm = new EditarMarcador(glyph,
activeGlyphDatabase.GetGlyphNames());
                glyphForm.Text = "Nuevo Marcador";

                glyphForm.SetGlyphDataCheckingHandeler( new
GlyphDataCheckingHandeler( CheckGlyphData ) );

                if ( glyphForm.ShowDialog( ) == DialogResult.OK )
                {
```

```
                    try
                    {
                        lock ( sync )
                        {
                            activeGlyphDatabase.Add( glyph );
                        }
                        glyphsImageList.Images.Add(                    glyph.Name,
CrearIconoMarcador( glyph ) );

                        ListViewItem  lvi  =  lstvMarcadores.Items.Add(  glyph.Name
);
                        lvi.ImageKey = glyph.Name;
                    }
                    catch
                    {
                        ShowErrorBox( string.Format( "Un marcador con el nombre
'{0}' ya existe en la base de datos.", glyph.Name ) );
                    }
                }
            }
        }

        private void editGlyphToolStripMenuItem_Click( object sender, EventArgs e
)
        {
            EditSelectedGlyph( );
        }

        private void EditSelectedGlyph( )
        {
            if    (    (    activeGlyphDatabase    !=    null    )    &&    (
lstvMarcadores.SelectedIndices.Count != 0 ) )
            {
                ListViewItem lvi = lstvMarcadores.SelectedItems[0];
                Glyph glyph = (Glyph) activeGlyphDatabase[lvi.Text].Clone( );
                glyphNameInEditor = glyph.Name;
                EditarMarcador    glyphForm    =    new    EditarMarcador(glyph,
activeGlyphDatabase.GetGlyphNames());
                glyphForm.Text = "Etitar Marcador";

                glyphForm.SetGlyphDataCheckingHandeler(                    new
GlyphDataCheckingHandeler( CheckGlyphData ) );

                if ( glyphForm.ShowDialog( ) == DialogResult.OK )
                {
                    try
                    {
                        lock ( sync )
                        {
                            activeGlyphDatabase.Replace( glyphNameInEditor, glyph
);
                        }
```

```
                        lvi.Text = glyph.Name;

                        lvi.ImageKey = null;

                        glyphsImageList.Images.RemoveByKey( glyphNameInEditor );
                        glyphsImageList.Images.Add(                    glyph.Name,
CrearIconoMarcador( glyph ) );

                        lvi.ImageKey = glyph.Name;
                    }
                    catch
                    {
                        ShowErrorBox( string.Format( "Un marcador con el nombre
'{0}' ya existe en la base de datos.", glyph.Name ) );
                    }
                }
            }
        }

        string glyphNameInEditor = string.Empty;

        private bool CheckGlyphData( byte[,] glyphData )
        {
            if ( activeGlyphDatabase != null )
            {
                int rotation;
                Glyph   recognizedGlyph   =   activeGlyphDatabase.RecognizeGlyph(
glyphData, out rotation );

                if ( ( recognizedGlyph != null ) && ( recognizedGlyph.Name !=
glyphNameInEditor ) )
                {
                    ShowErrorBox( "La base de datos ya contiene un marcador con
el mismo diseño." );
                    return false;
                }
            }

            return true;
        }

        private void deleteGlyphToolStripMenuItem_Click( object sender, EventArgs
e )
        {
            if    (    (    activeGlyphDatabase    !=    null    )    &&    (
lstvMarcadores.SelectedIndices.Count != 0 ) )
            {
                ListViewItem lvi = lstvMarcadores.SelectedItems[0];

                lock ( sync )
                {
                    activeGlyphDatabase.Remove(            lvi.Text            );
```

```
                }
                lstvMarcadores.Items.Remove( lvi );
                glyphsImageList.Images.RemoveByKey( lvi.Text );
            }
        }

        private void activateToolStripMenuItem_Click( object sender, EventArgs e
)
        {
            if ( lstvListaMarcadores.SelectedIndices.Count == 1 )
            {
                ActivarMarcadoresenMemoria(
lstvListaMarcadores.SelectedItems[0].Text );
            }
        }

        private void deleteToolStripMenuItem_Click( object sender, EventArgs e )
        {
            if ( lstvListaMarcadores.SelectedIndices.Count == 1 )
            {
                string selecteItem = lstvListaMarcadores.SelectedItems[0].Text;

                if ( selecteItem == activeGlyphDatabaseName )
                {
                    ActivarMarcadoresenMemoria( null );
                }

                glyphDatabases.RemoveGlyphDatabase( selecteItem );
                lstvListaMarcadores.Items.Remove(
lstvListaMarcadores.SelectedItems[0] );
            }
        }

        private void renameToolStripMenuItem_Click( object sender, EventArgs e )
        {
            if ( lstvListaMarcadores.SelectedIndices.Count == 1 )
            {

lstvListaMarcadores.Items[lstvListaMarcadores.SelectedIndices[0]].BeginEdit( );
            }
        }

        private void itmTipodeVisualizacion_Click( object sender, EventArgs e )
        {
            ToolStripMenuItem item = (ToolStripMenuItem) sender;

            if ( item.Tag is VisualizationType )
            {
                imageProcessor.VisualizationType = (VisualizationType) item.Tag;

                lock ( this )
                {
```

```
                    if        (        imageProcessor.VisualizationType        ==
VisualizationType.Model )
                    {
                        if ( arForm == null )
                        {
                            arForm = new AugmentedRealityForm( );

                            arForm.FormClosing  +=  new  FormClosingEventHandler(
arForm_FormClosing );
                            arForm.Show( );
                        }
                    }
                    else
                    {
                        if ( arForm != null )
                        {
                            arForm.Close( );
                        }
                    }
                }
            }
        }

        private void ActivarMarcadoresenMemoria( string name )
        {
            ListViewItem lvi;

            if ( activeGlyphDatabase != null )
            {
                lvi       =       LitaNombresMarcador(       lstvListaMarcadores,
activeGlyphDatabaseName );

                if ( lvi != null )
                {
                    Font font = new Font( lvi.Font, FontStyle.Regular );
                    lvi.Font = font;
            }
            }
            activeGlyphDatabaseName = name;

            if ( name != null )
            {
                try
                {
                    activeGlyphDatabase = glyphDatabases[name];

                    lvi = LitaNombresMarcador( lstvListaMarcadores, name );

                    if ( lvi != null )
                    {
                        Font font = new Font( lvi.Font, FontStyle.Bold );
                        lvi.Font                        =                        font;
```

```
                }
            }
            catch
            {
            }
        }
        else
        {
            activeGlyphDatabase = null;
        }

        imageProcessor.GlyphDatabase = activeGlyphDatabase;
        ActualizarListaMarcadores( );
    }

    private ListViewItem LitaNombresMarcador( ListView lv, string name )
    {
        try
        {
            return lv.Items[name];
        }
        catch
        {
            return null;
        }
    }

    private void ListadeMarcadoresenMemoria( )
    {
        lstvListaMarcadores.Items.Clear( );

        List<string> dbNames = glyphDatabases.GetDatabaseNames( );

        foreach ( string name in dbNames )
        {
            GlyphDatabase db = glyphDatabases[name];
            ListViewItem lvi = lstvListaMarcadores.Items.Add( name );
            lvi.Name = name;

            lvi.SubItems.Add( string.Format( "{0}x{1}", db.Size, db.Size ) );
        }
    }

    private void ActualizarListaMarcadores( )
    {
        lstvMarcadores.Items.Clear( );
        glyphsImageList.Images.Clear( );

        if ( activeGlyphDatabase != null )
        {
            foreach ( Glyph glyph in activeGlyphDatabase )
            {
```

```
                    glyphsImageList.Images.Add(  glyph.Name,  CrearIconoMarcador(
glyph ) );
                    ListViewItem lvi = lstvMarcadores.Items.Add( glyph.Name );
                    lvi.ImageKey = glyph.Name;
                }
            }
        }

        private Bitmap CrearIconoMarcador( Glyph glyph )
        {
            return CrearImagenMarcador( glyph, 32 );
        }

        private Bitmap CrearImagenMarcador( Glyph glyph, int width )
        {
            Bitmap      bitmap      =      new      Bitmap(      width,      width,
System.Drawing.Imaging.PixelFormat.Format32bppArgb );

            int cellSize = width / glyph.Size;
            int glyphSize = glyph.Size;

            for ( int i = 0; i < width; i++ )
            {
                if (i != cellSize){
                int yCell = i / cellSize;

                for ( int j = 0; j < width; j++ )
                {
                    int xCell = j / cellSize;

                    if ( ( yCell >= glyphSize ) || ( xCell >= glyphSize ) )
                    {
                        bitmap.SetPixel( j, i, Color.Transparent );
                    }
                    else
                    {
                        bitmap.SetPixel( j, i,
                            ( glyph.Data[yCell, xCell] == 0 ) ? Color.Black :
Color.White );
                    }
                }
                }
            }

            return bitmap;
        }

        private void arForm_FormClosing( object sender, FormClosingEventArgs e )
        {
            arForm.FormClosing -= new FormClosingEventHandler( arForm_FormClosing
);
            arForm                                =                                null;
```

```
            if ( imageProcessor.VisualizationType == VisualizationType.Model )
            {
                imageProcessor.VisualizationType = VisualizationType.Name;
            }
        }

        private void printToolStripMenuItem_Click( object sender, EventArgs e )
        {
            try
            {
                printDialog.Document = printDocument;
                if ( printDialog.ShowDialog( ) == DialogResult.OK )
                {
                    printDocument.Print( );
                }
            }
            catch ( InvalidPrinterException ex )
            {
                ShowErrorBox( "No se pudo acceder a la impresora.\r\n\r\n" +
ex.Message );
            }
        }

        private void printPreviewToolStripMenuItem_Click( object sender,
EventArgs e )
        {
            try
            {
                printPreviewDialog.Document = printDocument;
                printPreviewDialog.ShowDialog( );
            }
            catch ( InvalidPrinterException ex )
            {
                ShowErrorBox( "No se pudo acceder a la impresora.\r\n\r\n" +
ex.Message );
            }
        }

        private void printDocument_PrintPage( object sender,
System.Drawing.Printing.PrintPageEventArgs e )
        {
            if ( ( activeGlyphDatabase != null ) && (
lstvMarcadores.SelectedIndices.Count != 0 ) )
            {
                ListViewItem lvi = lstvMarcadores.SelectedItems[0];
                Glyph glyph = activeGlyphDatabase[lvi.Text];

                float glyphSizeInches = imageProcessor.GlyphSize / 25.4f;
                int imageWidth = (int) ( glyphSizeInches * 96 );

                Bitmap glyphImage = CrearImagenMarcador( glyph, imageWidth );
```

```
                glyphImage.SetResolution( 96, 96 );

                float xInches = ( (float) e.PageBounds.Width / 100 -
glyphSizeInches ) / 2;
                float yInches = ( (float) e.PageBounds.Height / 100 -
glyphSizeInches ) / 2;

                int xPixels = (int) ( xInches *
e.Graphics.VisibleClipBounds.Width / ( e.PageBounds.Width / 100 ) );
                int yPixels = (int) ( yInches *
e.Graphics.VisibleClipBounds.Height / ( e.PageBounds.Height / 100 ) );

                e.Graphics.DrawImage( glyphImage, xPixels, yPixels );
            }
        }

        private void ItmAcercaDe_Click(object sender, EventArgs e)
        {
            AcercaDe form = new AcercaDe();

            form.ShowDialog();
        }

        private void itmDeteccionCamara_Click(object sender, EventArgs e)
        {
            //VideoCaptureDeviceForm form = new VideoCaptureDeviceForm();

            //if (form.ShowDialog(this) == DialogResult.OK)
            //{
            //    OpenVideoSource(form.VideoDevice);
            //}


            SeleccionCamara form = new SeleccionCamara();

            if (form.ShowDialog(this) == DialogResult.OK)
            {
                VideoCaptureDevice videoSource = new
VideoCaptureDevice(form.VideoDevice);
                videoSource.DesiredFrameRate = 30;

                OpenVideoSource(videoSource);
            }
        }

        private void itmAbrirVideo_Click(object sender, EventArgs e)
        {
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                FileVideoSource fileSource = new
FileVideoSource(openFileDialog.FileName);
```

```
                OpenVideoSource(fileSource);
            }
        }

        private void itmSalir_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private    void    itmTipoVisualizacion_DropDownOpening(object    sender,
EventArgs e)
        {
            itmBordes.Checked      =      (imageProcessor.VisualizationType      ==
VisualizationType.BorderOnly);
            itmNombres.Checked      =      (imageProcessor.VisualizationType      ==
VisualizationType.Name);
            itmImagenes.Checked     =     (imageProcessor.VisualizationType     ==
VisualizationType.Image);
            itm3D.Checked       =       (imageProcessor.VisualizationType       ==
VisualizationType.Model);
        }

        private void itmOpciones_Click(object sender, EventArgs e)
        {
            Opciones optionsForm = new Opciones();

            optionsForm.AutoDetectFocalLength = autoDetectFocalLength;
            optionsForm.CameraFocalLength = imageProcessor.CameraFocalLength;
            optionsForm.GlyphSize = imageProcessor.GlyphSize;
            if (optionsForm.ShowDialog() == DialogResult.OK)
            {
                imageProcessor.GlyphSize = optionsForm.GlyphSize;
                autoDetectFocalLength = optionsForm.AutoDetectFocalLength;

                if (!autoDetectFocalLength)
                {
                    imageProcessor.CameraFocalLength                                =
optionsForm.CameraFocalLength;
                }
            }
        }

        private void itmManual_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("www.manual.edu.ec");
        }

        private void vspResultado_NewFrame(object sender, ref Bitmap image)
        {
            if (activeGlyphDatabase != null)
            {
                if     (image.PixelFormat     ==     PixelFormat.Format8bppIndexed)
```

```
                {
                    GrayscaleToRGB filter = new GrayscaleToRGB();
                    Bitmap temp = filter.Apply(image);
                    image.Dispose();
                    image = temp;
                }

                lock (sync)
                {
                    List<ExtractedGlyphData> glyphs =
imageProcessor.ProcessImage(image);

                    if (arForm != null)
                    {
                        List<VirtualModel> modelsToDisplay = new
List<VirtualModel>();

                        foreach (ExtractedGlyphData glyph in glyphs)
                        {
                            if ((glyph.RecognizedGlyph != null) &&
                                (glyph.RecognizedGlyph.UserData != null) &&
                                (glyph.RecognizedGlyph.UserData is
GlyphVisualizationData) &&
                                (glyph.IsTransformationDetected))
                            {
                                modelsToDisplay.Add(new VirtualModel(

((GlyphVisualizationData)glyph.RecognizedGlyph.UserData).ModelName,
                                    glyph.TransformationMatrix,
                                    imageProcessor.GlyphSize));
                            }
                        }

                        arForm.UpdateScene(image, modelsToDisplay);
                    }
                }
            }
        }

        private void vspResultado_PlayingFinished(object sender,
ReasonToFinishPlaying reason)
        {
            if (arForm != null)
            {
                arForm.UpdateScene(null, new List<VirtualModel>());
            }
        }

        private void lstvListaMarcadores_AfterLabelEdit(object sender,
LabelEditEventArgs e)
        {
            if (e.Label != null)
```

```
            {
                string newName = e.Label.Trim();

                if (newName == string.Empty)
                {
                    ShowErrorBox("El nombre no puede estar vacio.");
                    e.CancelEdit = true;
                    return;
                }
                else
                {
                    string oldName = lstvListaMarcadores.Items[e.Item].Text;

                    if (oldName != newName)
                    {
                        if (glyphDatabases.GetDatabaseNames().Contains(newName))
                        {
                            ShowErrorBox("Ya existe el nombre de marcador");
                            e.CancelEdit = true;
                            return;
                        }
                        glyphDatabases.RenameGlyphDatabase(oldName, newName);

                        if (activeGlyphDatabaseName == oldName)
                            activeGlyphDatabaseName = newName;

                        if (newName != e.Label)
                        {
                            lstvListaMarcadores.Items[e.Item].Text = newName;
                            e.CancelEdit = true;
                        }
                    }
                    else
                    {
                        e.CancelEdit = true;
                    }
                }
            }
        }

        private      void      lstvMarcadores_MouseDoubleClick(object      sender,
MouseEventArgs e)
        {
            EditSelectedGlyph();
        }

        private void Inicio_FormClosed(object sender, FormClosedEventArgs e)
        {
            Configuration config = Configuration.Instance;

            if (WindowState != FormWindowState.Minimized)
            {
```

```
                if (WindowState != FormWindowState.Maximized)
                {
                    config.SetConfigurationOption(mainFormXOption,
Location.X.ToString());
                    config.SetConfigurationOption(mainFormYOption,
Location.Y.ToString());
                    config.SetConfigurationOption(mainFormWidthOption,
Width.ToString());
                    config.SetConfigurationOption(mainFormHeightOption,
Height.ToString());
                }
                config.SetConfigurationOption(mainFormStateOption,
WindowState.ToString());
                config.SetConfigurationOption(mainSplitterOption,
splitContainer.SplitterDistance.ToString());
            }
            config.SetConfigurationOption(activeDatabaseOption,
activeGlyphDatabaseName);

            config.SetConfigurationOption(autoDetectFocalLengthOption,
autoDetectFocalLength.ToString());
            config.SetConfigurationOption(focalLengthOption,
imageProcessor.CameraFocalLength.ToString());
            config.SetConfigurationOption(glyphSizeOption,
imageProcessor.GlyphSize.ToString());

            try
            {
                config.Save(glyphDatabases);
            }
            catch (IOException ex)
            {
                ShowErrorBox("No se puede guardar, revise el nombre, diseño e
Imagen.\r\n\r\n" + ex.Message);
            }

            if (vspResultado.VideoSource != null)
            {
                vspResultado.SignalToStop();
                vspResultado.WaitForStop();
            }
        }

        private void Inicio_Load(object sender, EventArgs e)
        {
            Configuration config = Configuration.Instance;

            if (config.Load(glyphDatabases))
            {
                ListadeMarcadoresenMemoria();

ActivarMarcadoresenMemoria(config.GetConfigurationOption(activeDatabaseOption));
```

```
                try
                {
                    Location = new Point(
int.Parse(config.GetConfigurationOption(mainFormXOption)),
int.Parse(config.GetConfigurationOption(mainFormYOption)));
                    Size = new Size(
int.Parse(config.GetConfigurationOption(mainFormWidthOption)),
int.Parse(config.GetConfigurationOption(mainFormHeightOption)));
                    WindowState                                          =
(FormWindowState)Enum.Parse(typeof(FormWindowState),
                        config.GetConfigurationOption(mainFormStateOption));
                    splitContainer.SplitterDistance                      =
int.Parse(config.GetConfigurationOption(mainSplitterOption));
                    autoDetectFocalLength                                =
bool.Parse(config.GetConfigurationOption(autoDetectFocalLengthOption));
                    imageProcessor.GlyphSize                             =
float.Parse(config.GetConfigurationOption(glyphSizeOption));
                    if (!autoDetectFocalLength)
                    {
                        imageProcessor.CameraFocalLength                 =
float.Parse(config.GetConfigurationOption(focalLengthOption));
                    }
                }
                catch
                {
                }
            }
        }

        private void btnTransformar_Click(object sender, EventArgs e)
        {
            //Sobrepone las imagenes 2D sobre el rotulador
            imageProcessor.VisualizationType = VisualizationType.Image;
        }


        private void btnLetra_Click(object sender, EventArgs e)
        {
            Letras a = new Letras();
            a.Show();
            this.Hide();
        }

        private void corazónToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            Cerebro a = new Cerebro();
            a.Show();
        }

        private void corazónToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            Corazon a = new Corazon();
            a.Show();
        }

        private void estomagoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Estomago a = new Estomago();
            a.Show();
        }

        private void higadoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Higado a = new Higado();
            a.Show();
        }

        private void intestinoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Intestino a = new Intestino();
            a.Show();
        }

        private void pulmonesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Pulmones a = new Pulmones();
            a.Show();
        }

        private void riñonToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Riñon a = new Riñon();
            a.Show();
        }


    }
}
```

3. FORMULARIO AYUDA

La programación de la interfaz ayuda es la siguiente:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace PIVRA
{
    public partial class AcercaDe : Form
    {
        public AcercaDe( )
        {
            InitializeComponent( );

        }

        private void label_LinkClicked( object sender,
LinkLabelLinkClickedEventArgs e )
        {
            System.Diagnostics.Process.Start( e.Link.LinkData.ToString( ) );
        }
        private void VisitLink(Int32 a)
        {
            if (a == 1)
            {
                // cambia de color
                linkLabel2.LinkVisited = true;
                //llama al procedimiento y abre la pagina
                System.Diagnostics.Process.Start("www.uta.edu.ec");
            }
            if (a == 2)
            {
                // cambia de color
                linkLabel1.LinkVisited = true;
                //llama al procedimiento y abre la pagina
                System.Diagnostics.Process.Start("http://fisei.uta.edu.ec/");
            }
        }


        private void linkLabel1_LinkClicked_1(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            try
            {
```

```
                VisitLink(2);
            }
            catch (Exception ex)
            {
                MessageBox.Show("No se puede Abrir esta página.");
            }
        }

        private void linkLabel2_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            VisitLink(1);
        }

        private void btnOk_Click(object sender, EventArgs e)
        {

        }


    }
}
```

ANEXO D: Guía de entrevista y observación

CUESTIONARIO ENTREVISTA
Pregunta 1: ¿Piensa usted que la Unidad Educativa "Tirso de Molina" está a la vanguardia de la tecnología informática?
Pregunta 2: ¿Sabe usted sobre la importancia de las TIC's en la educación?
Pregunta 3: ¿Conoce usted la tecnología denominada Visión Artificial?
Pregunta 4: ¿Ha escuchado usted sobre el término Realidad Aumentada?
Pregunta 5: ¿Qué descripción tecnológica modela su mente con la palabra Realidad Aumentada?
Pregunta 6: ¿Impulsaría usted el uso de software con tecnología de Realidad Aumentada para fortalecer los procesos de enseñanza y aprendizaje en la educación?
Pregunta 7: ¿Considera usted preponderante la incursión de la tecnología Realidad Aumentada en la calidad perceptiva de conocimientos para el aprovechamiento de los alumnos?

Elaborado por: Jimena Caguana

GUÍA DE OBSERVACIÓN
Permite determinar información in situ de la experiencia existente en el uso de alguna aplicación de Realidad Aumentada por parte de los docentes, alumnos o administrativos de la Unidad Educativa "Tirso de Molina". Además, es utilizada en la Ingeniería de Usabilidad (UI) del software, con el propósito de observar al usuario interactuar con el sistema PIVRA en su ambiente de trabajo. Se identifican los siguientes puntos claves para la observación:
Punto 1: Auditoría informática de bajo nivel en los ordenadores de la institución (programas – websities).
Punto 2: Experiencia con la Realidad Aumentada - Entrevista
Punto 3: Falla la interacción del sistema PIVRA, el sistema no responde con la acción deseada
Punto 4: Número de veces que el usuario obstruye accidentalmente el marcador
Punto 5: Número de veces en que el sistema falla en detectar el marcador
Punto 6: El usuario deteriora el marcador

Elaborado por: Jimena Caguana

ANEXO E: Fotografías

SISTEMA PIVRA

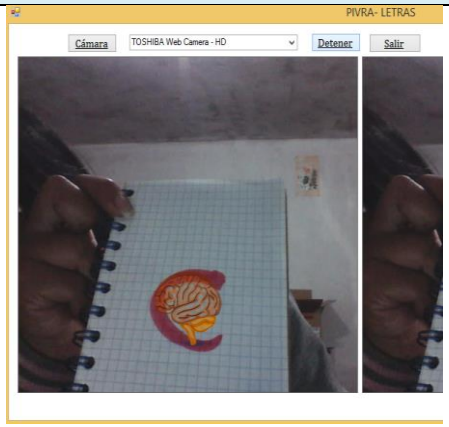


Foto 1: RECONOCIMIENTO DE LETRA C

Foto 2: RECONOCIMIENTO DE LOS MARCADORES DE EJECUCIÓN

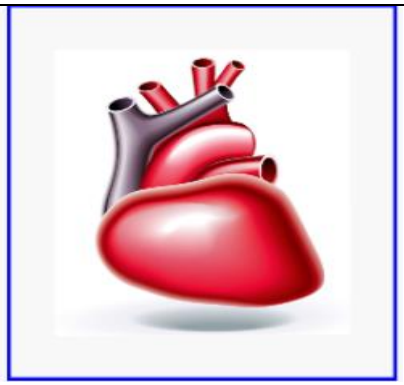


Foto 3: REALIDAD AUMENTADA EJEMPLO 1

Foto 4: REALIDAD AUMENTADA EJEMPLO 2

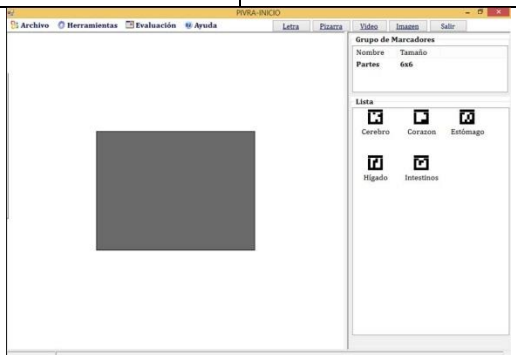


Foto 5: FORMULARIO PRINCIPAL

ÓRGANOS INTERNOS DEL CUERPO HUMANO:

EDUCACIÓN INTERACTIVA USANDO PIVRA

Foto 6: MODELO 3D DEL CEREBRO

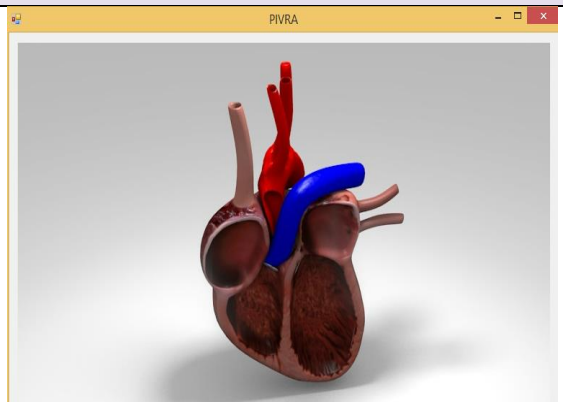


Foto 7: MODELO 3D DEL CORAZÓN

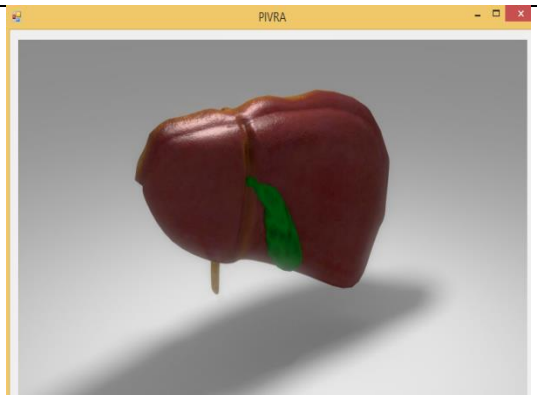


Foto 8: MODELO 3D DEL HÍGADO

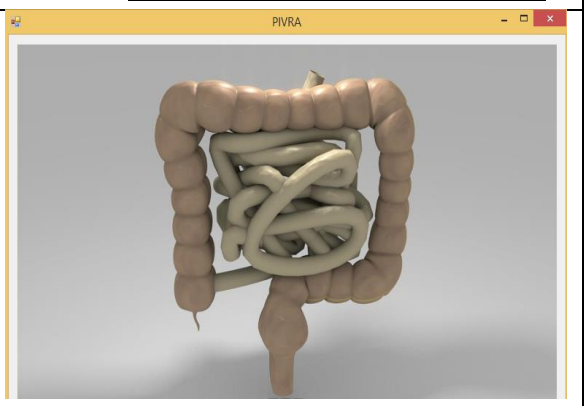
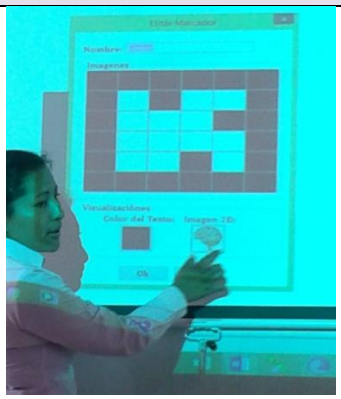
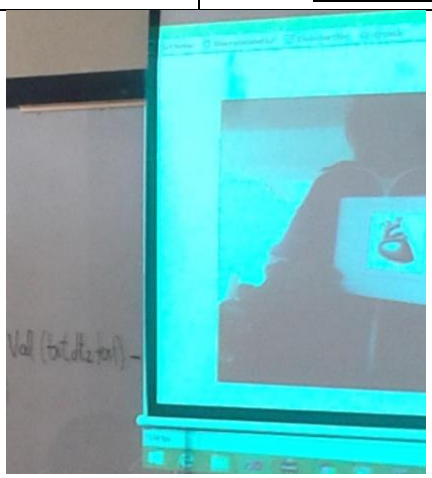


Foto 9: MODELO 3D DEL INTESTINO

Foto 10: VENTANA ACERCA DE

CAPACITACIÓN USO SISTEMA PIVRA:

EDUCACIÓN INTERACTIVA USANDO PIVRA	
 **Foto 11:** CREACIÓN MARCADORES	 **Foto 12:** USO DE PIVRA
 **Foto 13:** CLASE CON PIVRA	 **Foto 14:** CAPACITACIÓN PROFESORES-ALUMNOS
 **Foto 15:** MANIPULACION PIVRA	