UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMATICA

TESIS DE GRADO

RECONOCIMIENTO DE GESTOS PARA LA INTERACCIÓN POR COMPUTADOR, CON REALIDAD AUMENTADA.

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA EN SISTEMAS INFORMÁTICOS

POSTULANTE: MARISOL QUISPE ADUVIRI

TUTOR METODOLÓGICO : M.Sc. JORGE H. TERAN POMIER

ASESOR: LIC. ROBERTO VARGAS BLACUTT

La Paz – Bolivia

2013

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

Dedicatoria:

A mis padres Regina y Andrés,
quienes me brindaron apoyo
incondicional y comprensión en
el transcurso de mi formación
como profesional.

AGRADECIMIENTOS

Debo dar las gracias a mi revisor del proyecto, Lic. Roberto Vargas Blacutt y a mi tutor M.Sc. Jorge H. Teran Pomier por apoyarme y guiarme en el desarrollo del proyecto, y haber hecho posible la finalización del mismo.

Quisiera expresar también mi gratitud al Ing. Luis Soto Dorado por abrirme las puertas de su Empresa DACORP S.R.L., para la realización de la Tesis, colaborándome con todas las ideas innovadoras para la elaboración y culminación del proyecto y a todos los licenciados de la carrera de informática quienes hicieron posible mi formación como profesional.

También agradecer a todos mis amigos(as) de la carrera por su constante apoyo, comprensión y sobre todo por la amistad incondicional que me brindaron siempre e hicieron posible que no me rindiera ante situaciones difíciles.

A mis compañeros de la empresa DACORP S.R.L. y a Juan Carlos Huarachi, quienes siempre estuvieron prestos a ayudar.

Y finalmente agradecer a mi padre y a mi madre quienes me formaron como persona e hicieron posible culminar con una meta mas en mi vida, a mis hermanos quienes me brindaron momentos de alegria.

.....de todo corazón, muchisimas gracias!!

RESUMEN

El uso de gesto para comunicarse con aplicaciones esta a la orden del día, especialmente tras la aparición de los dispositivos con pantalla táctiles, aunque esta características no es la única este tipo de dispositivos con pantallas táctiles, aunque esta característica no es única de este tipo e dispositivos, sino que están apareciendo en el mercado nuevos terminales que permiten una iteración persona-maquina sin ningún tipo de contacto, ya sea a través del análisis de imágenes o utilizando algún otro tipo de sensores.

La comunicación a través de gestos, además de tratar de ofrecer una manera más sencilla e intuitiva de comunicarse con otros sistemas o aplicaciones, también aporta importantes avances a personas que poseen algún tipo de problema de movilidad, de manera que para ellas, realizar tareas que antes, llevarlas a cabo utilizando únicamente periféricos como teclado o ratón eran muy costoso e incluso imposibles, con estos nuevos métodos de comunicación, muchas aplicaciones podrán llegar a ser accesibles para un mayor número de usuarios.

A modo de contextualización se presenta una revisión del estado del arte en el ámbito de las metodologías y herramientas de reconocimiento gestual. El objetivo general planteado corresponde al diseño, implementación y validación de una plataforma de software para el reconocimiento de la mano en tiempo real a través de una pantalla virtual, y esta a su vez tiene eventos de interacción, dando un principio hacia la realidad aumentada, utilizando métodos ya establecidos y validados, como lo son el algoritmo Haar para detección de gestos.

Para la ejecución de pruebas, del clasificador se impuso la mano sobre la pantalla y en 3 segundos lo reconoció, también es importante la resolución que se esté manejando de las cámaras.

Después del estudio del clasificador y analizar los resultados podemos determinar que hemos realizado estudio exhaustivo sobre la clasificación de manos y la certeza de poder generar un entrenamiento guiado que nos ha llevado a un clasificador de manos basado en los características de Haar y a un clasificador ADABoost. Una vez dicho esto, hemos valorado el resultado que hemos obtenido con nuestro clasificador. En cuanto a eficacia, hemos obtenido una media de 87,4% de porcentaje de acierto, lo que nos permitiría sobradamente detectar objetos en un entorno controlado.

Las conclusiones obtenidas sobre las librerías que se estudio para el reconocimiento gestual, se pudo determinar a través de resultados obtenidos la más recomendable es OpenCV, siendo este que un potente procesador de imágenes, que también tiene otras funcionalidades de detección de rostro, detección de formas, detección de movimiento, a comparación de otras librerías limitadas como AForge.Net.

PALABRAS CLAVE

Reconocimiento gestual, Opencv, Aforge.Net, librerías, movimiento, clasificador, pantalla virtual, imágenes, detección, realidad aumentada.

CONTENIDO

INDICE DE FIGURAS

CONTENIDO **Pág.**

INDICE DE TABLAS

CONTENIDO **Pág.**

INTRODUCCIÓN

1. Introducción

El avance de la tecnología, particularmente el desarrollo de nuevos dispositivos y aplicaciones ha impulsado el interés en explorar nuevas modalidades de interacción, que permitan al usuario aprovechar al máximo las capacidades y funcionalidades de estas nuevas tecnologías. En este sentido, existe un creciente interés, en especial, por las interfaces de usuario naturales, que son capaces de interpretar expresiones naturales del ser humano como los gestos y las palabras, dependiendo poco así de la manipulación de dispositivos hardware por parte del usuario

Las manos nos permite mover objetos que se encuentran a nuestro alcance, es importante tratar el lenguaje corporal de los movimientos; nuestra realidad física es entendida a través de la vista, el oído, el olfato, el tacto y el gusto. La realidad aumentada viene a potenciar esos cinco sentidos del mundo real complementándose con la del digital agrupando tecnologías que en tiempo real permiten la superposición de imágenes, o información generados virtualmente, sin poder tocarlos pero si verlos sobre imágenes del mundo real, de esta manera se crea un entorno en el que la información y los objetos virtuales se fusionan con los objetos reales ofreciendo una experiencia, para que el usuario que puede llegar a pensar que forma parte de su realidad cotidiana olvidando incluso la tecnología que le da soporte.

El proyecto se enfoca en la idea de profundizar la busqueda de una tecnica o herramienta que detecte la palma de la mano, relacionando el campo de la vision computacional dando inicios al desarrollo de una interfaz de RA, primeramente generando una pantalla virtual que hace que este inmerso el usuario, realizando el reconocimiento gestual de la mano proyectándose a través del dispositivo de cámara, procesando imágenes en tiempo real relacionando eventos en diferentes lugares de la pantalla .

1.2. Antecedentes

Desde tiempos remotos el hombre siempre mostro su imaginación tratando de salir de lo cotidiano, ahora con el desarrollo de la tecnología de software, ordenadores, sistemas operativos con mayor capacidad de desarrollo, el procesamiento de imagen y de vídeo en tiempo real son cada vez mayores y van apareciendo nuevas posibilidades en sistemas gráficos digitales, consiguiendo efectos más realistas en el uso de los sistemas de RA, asimismo se han extendido a muchos sectores, cada vez es más evidente para los grupos de investigación, en el ámbito docente como en el ámbito industrial, de la potencia que posee la RA para transmitir de forma visual una información digital que potencie o sustituya parcialmente el entorno percibido y esto permite reducir la sensación de artificialidad de los elementos virtuales añadidos.

1.3. Descripción del Problema

Analizando los diferentes algoritmos que ya existen para el reconocimiento gestual y mediante el uso de técnicas de Visión por Computador y reconocimiento de formas, gestos, rostros, incorporando las técnicas de procesamiento de imágenes, se intenta que el ordenador identifique la mano de un usuario y sea capaz de mostrar una respuesta, el computador reaccionará a la mano del usuario y éste visualizará en tiempo real tanto su entorno como la respuesta generada, para eso se ha desarrollado una aplicación con una librería de funciones que reconocen la mano del usuario a través de la camara web.

1.4 Objetivos

1.4.1. Objetivo General

Realizar el estudio de las librerías y herramientas que existen para el desarrollo de reconocimiento gestual que detecte la mano para la manipulación objetos sobre una pantalla virtual proyectada desde el monitor de la pc, que permita interactuar con el usuario en cualquier posición en tiempo real.

1.4.1. Objetivo Específicos

a) Ver la funcionalidad que ofrece la librería OpenCV y AForge.NET ,para la detección de gestos, elegir las mas eficiente en terminos de usabilidad, eficiencia y portabilidad.

b) Identificar los gestos llevados a cabo por el usuario a través de una cámara video

c) Proponer la implicación de reconocimiento gestual, visión computacional conllevan a la aplicación de desarrollo de realidad aumentada.

d) Implementación de la librería de detección de movimiento, hará uso de una cámara o algún dispositivo similar que permita capturar video, este dispositivo capturar diferentes frames durante el movimiento del usuario y tras aplicar un algoritmo se identificara el gesto que el usuario ha llevado a cabo.

e) Desarrollar e implementar una interfaz multimedia virtual, inmersiva, basado en reconocimiento gestual,.

f) Crear un programa que aplique lo aprendido con un clasificador por defecto.

1.5. Hipótesis

Mediante las técnicas de la visión computacional será posible implementar una interfaz inmersivo, multimedia y virtual basado en el reconocimiento de gestos sin la necesidad de tener entradas perifericas"? formando un entorno de Realidad Aumentada.

1.6. Justificaciones

1.6.1. Tecnológica

En la actualidad el desarrollo de la realidad aumentada con reconocimiento de gestos, pantallas virtuales resulta complejo y novedoso en varios ámbitos de la tecnología y vida cotidiana.

1.6.2. Económica

El estudio del reconocimiento gestual esta inmerso a la visión computacional, junto a la realidad aumentada es el inicio para poder incursionar en el ámbito del ocio y el marketing, presumiblemente se va extendiendo a otras áreas a medida que la tecnología avance, como ser a la manufactura, mantenimiento automovilístico, aeronáutico, el entrenamiento de habilidades y destreza; a este grupo se suma los sectores de turismo, educación y la salud. Evitando La RA nos facilita de sobremanera para la visualización de todo y sin que se tenga que tener los objetos en físico, es por eso que se implementa sobre una librería abierta para no depender de empresas que restringen el código, obteniendo un costo minimo y software necesario con solo una camara web.

1.7. Herramientas utilizadas

1.7.1. Microsoft Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C #, Visual J #, ASP.NET y Visual Basic .NET, la programación orientada a objetos conjuntamente con el sistema de desarrollo SDK2 de Windows, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

1.7.2. Audiere.

En el desarrollo se uso la libreria Opencv, esta no cuenta con el desarrollo para los temas de audio, y para esto se hizo uso de la libreria audiere.

Audiere es una API de audio de alto nivel. puede reproducir Ogg Vorbis , MP3, FLAC , sin comprimir WAV, AIFF, MOD, S3M, XM, y archivos. Para la salida

de audio, Audiere soporta DirectSound o WinMM en Windows, Linux y OSS en Cygwin, y SGI IRIX en AL.

Audiere es open source, bajo la licencia LGPL . Audiere es portátil e independiente.

1.8. Alcances y Limites del Trabajo

1.8.1. Alcances

En el presente trabajo de investigación realizara el estudio, análisis e implementación de una aplicación interactiva por visión computacional, con introduccion a la realidad aumentada, permitiendo el reconocimiento gestual y el desarrollo de una pantalla virtual, creando interactividad con la pc sin necesidad de entradas perifericas como el mause, generando una solución autónoma a partir de la librería o herramienta que se elija como la conveniente y optima para el desarrollo de reconocimiento gestual.

1.8.2. Limites

EL desarrollo de la interfaz solo sera para pc, el mismo tipo de desarrollos se puede realizar para telefonos celulares con el sisitema operativo de android. Endfocandose en la deteccion de la mano humana.

MARCO TEÓRICO

2.1. Visión Computacional

Visión Computacional o visión artificial por computador es un campo de la Inteligencia artificial que trata de alguna forma de emular esta capacidad en las computadoras, mediante la interpretación de las imágenes adquiridas, por ejemplo con una camara, se puedan reconocer los diversos objetos en el ambiente y su posicion en el espacio para que pueda llegar a comprender una escena. La visión artificial incluye muchas técnicas, algunas de ellas son el procesamiento de imágenes y la extracción de patrones; otras más significativas tratan de conseguir una descripción geométrica útil: forma, perímetro, área para realizar modelos geométricos y con otra se realiza un análisis de color.

Este conjunto de técnicas incluye pre procesamiento, segmentación, descripción, clasificación e interpretación de resultados. Las imágenes recogidas para su interpretación son imágenes digitales. Una imagen digital es una representación discreta, con valores en puntos regularmente muestreados y es una representación cuantificada, donde cada valor es un número entero. Las imágenes constituyen la materia prima de una ciencia que se esfuerza en emular las capacidades perceptivas del hombre.

2.2. Reconocimiento de Gestos

La palabra “gesto” se relaciona con movimientos que se hacen con los músculos de la cara para expresar algo. Sin embargo, este concepto también se relaciona con cualquier movimiento que se realice con cualquier parte del cuerpo. Este proyecto se centra en gestos realizados principalmente con las manos.

El reconocimiento de gestos ha sido tema de interés por muchas personas y empresas principalmente de videojuegos u orientados a la elaboración de dispositivos, se busca cada vez simplificar más la comunicación e interacción entre el humano y las máquinas buscando manipularlas o recibir algún tipo de respuesta en particular por medio de señales más naturales y/o significativas como lo es moviendo una mano, la cabeza, el mismo cuerpo.

2.3. Realidad Aumentada

La realidad aumentada (RA) es el termino que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos combinan con elementos virtuales es una tecnología que trata de incluir información generada por un computador sobre un mundo real, centrándose en proporcionar un entorno virtual para el usuario y alternar en tiempo y escenario real.

La RA integra señales captadas del mundo real (video y audio) con señales generadas por computadores (objetos gráficos tridimensionales) inclusión de modelos virtuales gráficos 2D y 3D en el campo de visión del usuario; haciendo la visualización del procesamiento de imágenes captadas por la cámara de video, estos son analizados por procesos de visión para extraer propiedades geométricas del entorno y de los objetos, los gráficos por computadora toma la síntesis de objetos tridimensionales y sus transformaciones, mientras que gracias a la teoría de interfaces gráficas ha sido posible la construcción de nuevas metáforas dentro de estos mundos mixtos.

La concatenación de los procesos que implica la RA, resulta en un sistema con las siguientes características, las cuales la definen:

- Combina objetos reales y virtuales en nuevos ambientes integrados.
- Las señales y su reconstrucción se ejecutan en tiempo real.
- Las aplicaciones son interactivas.
- Los objetos reales y virtuales son registrados y alineados geométricamente entre ellos y dentro del espacio, para darles coherencia espacial

Existen dos definiciones comúnmente aceptables de la RA en la actualidad.

- (Kishino. P. & MIlgram F. , 1994) definen un conjunto llamado “Milgram-Virtuality Continuum” (ver fig. 2.1) que describe un entorno real y un entorno

virtual puro. Entre medio hay RA (mas cerca del mundo real) y virtualidad aumentada (mas proxima a la virtualidad pura).

Figura 2.1:Continúo de Milgram
Fuente: Revista de la Tecnología Audiovisual

- (Azuma, 1997) en la defunicion de Ronald A. identifica a tres características fundamentales para la RA, las cuales son:
  - Combinación de objetos virtuales y reales en un ambiente real.
  - Interactividad tiempo real.
  - Objetos virtuales coherentes con el mundo real.

La Realidad Aumentada (RA) está emergiendo como una tecnología con grandes posibilidades de aplicación en diferentes ámbitos. Constantemente se plantea la generación de nuevas aplicaciones de RA en muchos dominios de aplicación, tales como educación, entretenimiento, investigación, industria, arte, etc.

La evolución tecnológica es tan drástica que ahora existe una nueva manera de complementar el contenido publicitario de una forma más fácil, efectiva y, sobre todo, mucho más impresionante: la realidad aumentada, consiste en aumentar la realidad del entorno físico por medio de la interacción con quien lo esté utilizando. Por el momento, en nuestro país se desarrollo aplicaciones y sistemas de realidad aumentada a través de marcadores, tales empresas como ENTEL(Empresa Nacional de Telecomunicaciones), que en sus publicidades te muestran un marcador, el cual te inmersa a un mundo con objetos inanimados, así como esa publicidad varias pequeñas, medianas y grandes

empresas comenzaron con esa tecnología, tratando de generar un impacto al consumidor, con el tiempo sera vista con mayor frecuencia en nuestro país esta estrategia publicitaria tan singular como sorprendente, en la fig. 2.2 se muestra una visión directa o indirecta, del entorno físico real, cuyos componentes se mezclan con elementos digitales para crear una situación mixta en tiempo reala partir del reconocimiento gestual.

Figura 2.2: Venta de Zapatos con Realidad Aumentada
Fuente: Virtual Shoe Fitting - Goertz Casefilm

2.4. Estudio de la Situación Actual

2.4.1. Evaluación de productos existentes para Reconocimiento de Gestos.

Existe una amplia variedad de librerías de código abierto, productos y/o sistemas que permiten interactuar mediante el uso de gestos. Seguidamente se van a describir algunos de los mas populares o curiosos.

2.4.2. Wiigee

Es una biblioteca de reconocimiento de gestos basada en java, de código abierto para gestos, fue desarrollada por Benjamin Poppinga, un estudiante de la Universidad de Oldenburg (Alemania). Wiigee basa en reconocimiento de gestos usando el algoritmo de cadenas de Markov, con las características: definir sus propios gestos arbitrarios, reconoce gestos con gran precisión, basado en una arquitectura orientada a eventos con los que usted será capaz de integrar el gesto de entrada tan fácil como el ratón de entrada común. Solamente basados y desarrollados específicamente para el regulador del telecontrol de Nintendo Wii.

2.4.3. OpenCV

The Open Source Computer Vision Library (OpenCV), nació el 13 de junio de 2000, bajo Intel Corporación, especialmente diseñado para el tratamiento, captura y visualización de imágenes en áreas como interacción hombre-máquina, robótica, monitorización, biométrica, segmentación y reconocimiento de objetos y seguridad y que proporciona bibliotecas de tipos de datos estáticos y dinámicos. Es una librería optimizada para ser usado bajo procesadores Intel, de acceso libre y además multiplataforma, compatible con Mac OS, Linux y Windows.

La biblioteca OpenCV es una API la cual tiene implementada más de 500 que abarcan muchas aéreas de la visión y esta diseñada especialmente para el tratado de imagenes en tiempo real, funciones escritas en lenguaje C++

Concretamente, el conjunto de funciones suministradas por la librería OpenCV pueden agruparse en los siguientes bloques:

- Estructuras y operaciones básicas.
- Procesamiento y análisis de imágenes: filtros, momentos, histogramas
- Análisis estructural: geometría, procesamiento del contorno, etc.

- Análisis del movimiento y seguimiento de objetos: plantillas de movimiento, seguidores, flujo óptico, etc.
- Reconocimiento de objetos: objetos propios (eigen objects), modelos HMM, etc.
- Calibración de la cámara: morphing, geometría epipolar, estimación de la pose.
- Reconstrucción tridimensional (funcionalidad experimental): detección de objetos, seguimiento de objetos tridimensionales, etc.
- Interfaces gráficos de usuarios y adquisición de video.

OpenCV se puede dividir en 5 librerías bien diferenciadas como estructura fundamental para su uso.

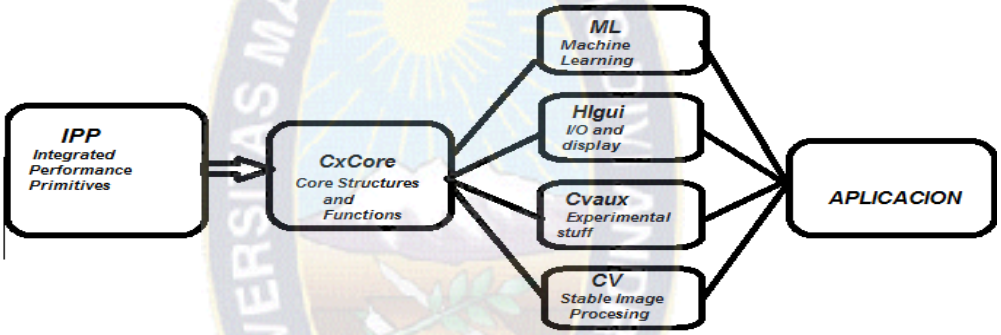

Figura 2.3: Representación de Opencv
Fuente: Elaboración Propia

Sobre el análisis estructural, se puede encontrar función cv, que es para procesamiento de contornos y computación geométrica. También existen funciones para tratar aspectos de análisis de movimiento y seguimiento de objetos, procesamiento de flujo óptico, reconocimiento de patrones y estimación de la pose de objetos, así como funciones para reconstrucción en 3D (calibración de la webcam, reconocimiento de gestos, etc).

- CvAux algoritmos mas experimentales, para visión estéreo, seguimiento 3D.
- HighGUI permite la escritura/lectura de imágenes en numerosos formatos (BMP, JPEG, TIFF, PxM, Sun Raster, etc.) y la captura de stream de video de capturadotas y cámaras/capturadoras con controlador VFW/WDM. También

permite la creación de ventanas para visualizar imágenes en ellas. Estas ventanas recuerdan su contenido

- Cv es el que 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.
- CxCore dispone de estructuras básica de datos dinámicos y operadores para estas estructuras.
- La mayoría recibe por parámetro un vector máscara para especificar sobre qué elementos de la estructura se aplica el operador. Podremos encontrar operadores aritméticos (Add, Mult, Sub, AbsDiff…), lógicos (AND, OR, NOT, XOR), de comparación, etc., así como funciones para aplicar cálculos estadísticos (SUM, MAX, MIN, AVG…) y operadores de conversión de tipo o escala de colores.

2.4.4. EmguCV

Es una plataforma, un wrapper sobre OpenCV, que permite utilizar la libreria sobre .Net es rapida y eficiente de codigo abierto, Permitir OpenCV funciones que deben desembolsar. idiomas compatibles NET como C #, VB, VC + +, etc IronPython La envoltura puede ser compilado en Mono y se ejecuta en Windows, Linux, Mac OS X, iPhone, iPad y dispositivos

2.4.5. Kinect

Es un producto que trabaja con un interrelaciona con la camama través de Windows 7 y Windows 8, únicamente capturando los movimientos de los usuarios e interactuar con la consola si la necesidad de tener contacto fisico con un controlador. Aunque Kinect estaba pensada para utilizar con la videoconsola Xbox, seis mese después fue lanzado su SDK, que permite hacer desarrollo de aplicaciones desde la PC.

Kinect cuenta con una cámara RGB, un sensor de profundidad, un micrófono multirray y un procesador independiente, el controlador de Kinect permite identificar a las personas situadas ante la cámara, localizando sus extremidades y articulaciones, de manera que reconocerá cada uno de los movimientos realizados, tanto de la persona completa como de una de sus extremidades. El sistema puede llegar a reconocer hasta veinte articulaciones por persona, con un máximo de seis personas.

Una de las grandes ventajas que ofrece kinect es el correcto funcionamiento independiente de la iluminación que existía en el ambiente. Su funcionamiento es perfecto tanto con baja luz como con un foco de luz directa sobre el objetivo, aunque la potencia de Kinect no esta únicamente en la imagen, si no que tambien ofrece un gran abanico de posibilidades ante el reconocimiento de audio, aplicando filtros que permiten eliminar de forma eficiente ruido de fondo, incluso distinguir una voz concreta.

2.4.6. Air Gesture (Galaxy)

Air Gesture reconoce el movimiento de tu mano sobre el telefono, con lo cual podemos pasar de una imagen a otra, recorrer pistas de audio o incluso aceptar y rechazar llamadas o hacer scroll sin tocar el equipo, tan solo desplazando la mano por encima de él. A esto, Samsung ha denominado, la evolución del concepto táctil. es una función totalmente nueva del Galaxy S4 que hace uso de varios sensores infrarrojos del dispositivo para detectar el movimiento de nuestras manos.

2.4.7. AForge.NET

Aforge.NET esta diseñada para trabajos relacionada con visión artificial, que permite captura y procesamiento de imágenes, procesamiento de video, computación de redes neuronales, etc. El Aforge.NET es un marco de C# disenado para desarrolladores e investigadores en los campos de Visión por computador e Inteligencia Artificial, procesamiento de imágenes, redes neuronales, algoritmos genéticos, aprendizaje automático robótica, etc.

2.4.8. Openarch

Se trata de un sistema que busca convertir cualquier tipo de superficie en pantalla dactil.

El proyecto es de código abierto y pretende acercarnos a la domotica del futuro, debido a que cada vez disponemos de un mayor número de dispositivos o sensores en nuestros hogares, muchas veces interconectados entre si, progresando hacia la interconexión continua y permanente.

Openarch, ofrece un prototipo de vivienda inteligente que incorpora una capa digital que conecta la casa y ciertos elementos a internet. Consta de un sistema de sensores y proyectores que permiten interactuar tanto a través de gestos como de voz, mostrando información en cualquier superficie, tanto paredes, techos o suelos. Estas proyecciones se activa con la presencia, permitiendo controlar todo a través de movimientos de manos, la interfaz de complementa con un sistema RFID integradas en los objetos de la pacas permitiendo obtener información de los mismos a través de un dispositivo de lectura que incorpore la tecnología NFC.

De este modo la vivienda se transforma en un espacio de interacción continua e intercambio de datos, permitiendo compartir información entre varios hogares, o concretar con otra persona de una manera mas real, visualizando a la persona a tamaño natural sobre una pared. Se trata de un proyecto muy amplio que además proporciona una interfaz de comunicación con y desde cualquier zona de la casa, permite entre otras cosas adaptar el entorno domestico a cada momento a través del uso de paneles móviles, o ofreciendo acceso a datos en todo momento en tiempo real, para controlar determinados elementos desde lugar, tanto dentro de casa como fuera, a traves de un dispositivo móvil o cualquier pc.

(http://www.openarch.cc/ -2013)"Openarch es una plataforma basada en la filosofia del software y el hardware libre. Un playground habitado donde poner a prueba tecnologías relacionadas con la vivienda inteligente."

2.4.9. Leap Motion

Se trata de un pequeno dispositivo que se conecta a nuestro PC, detectando los movimientos realizados por nuestras manos y dedos, permitiendo utilizar el ordenador como si se tratara de una pantalla dactil, pero sin llegar a tocar la pantalla. Aunque ya se sabe que la empresa HP incorporara este dispositivos en los proximos ordenadores del portátil HP Envy 17.

gracias a sus sensore, dos camaras y tres LEDs infrarojos, permite realizar un seguimiento continuo de las manos y dedos sobre tres ejes, con un alcanse maximo de un metro y una precision espacial de 0,01 milimetros.

Aunque es un dispositivo apto para trabajar con cualquier aplicacion existente como si se tratara de cualquier raton actual, es posible crear aplicaciones que aprovechen de una manera mas completa las posibilidades del mismo, ya que estamos acostumbrados a trabajar con un unico dispositivos apuntando, como es el raton, o incluso dos zonas de contacto, si utilizamos nuestros dedos en patalla tactiles, sin embargo Leap Motion es capaz de detectar nuestros diez dedos interactuando cada uno de ellos de manera diferente sobre la aplicacion. En caso de necesitar ser mas precisos podemos utilizar un elemento que sirva como apuntador como puede ser un lapiz.

2.5. Clasificador AdaBoost

Esta etapa dentro del algoritmo de detección se encarga de asignar un conjunto de características dado a una clase con la que se encuentra una mayor similitud, de acuerdo a un modelo inducido durante el entrenamiento.

Boosting: es un método de clasificación que combina varios clasificadores básicos para formar un único clasificador más complejo y preciso. La idea se basa en la afirmación de que varios clasificadores sencillos, cada uno de ellos con una precisión ligeramente superior a una clasificación aleatoria, pueden combinarse para formar un clasificador de mayor precisión, siempre y cuando se disponga de un número suficiente de muestras de

entrenamiento. La aplicación de clasificadores en cascada ha permitido obtener buenos resultados.

Para aplicar la técnica de boosting primero se debe establecer un algoritmo de aprendizaje sencillo (clasificador débil o base), que será llamado repetidas veces para crear diversos clasificadores base. Para el entrenamiento de los clasificadores bases se emplean, en cada iteración, un subconjunto diferente de muestras de entrenamiento y una distribución de pesos diferente sobre las muestras de entrenamiento Finalmente, estos clasificadores base se combinan en un único clasificador que se espera sea mucho más preciso que cualquiera de los clasificadores base por separado.

En función de los clasificadores bases que se utilicen, las distribuciones que se empleen para entrenarlos y el modo de combinarlos, podrán crearse distintas clases del algoritmo genérico de boosting. El algoritmo de boosting empleado por Viola y Jones en su trabajo es conocido como AdaBoos.

2.5.1. Algoritmo de Detección de Gestos basado en filtros Haar y clasificadores en cascada.

El algoritmo utilizado para la detección de gestos, tanto en la etapa de entrenamiento como en la etapa de reconocimiento en video, se basa en la extracción de características con filtros de ‘base Haar’ y clasificadores en modo cascada. Este método, llamado "Clasificadores Haar", fue propuesto por Paul Viola y Michael Jones, es un clasificador basado en árboles de decisión con entrenamiento supervisado.

Este clasificador funciona a partir de un entrenamiento previo realizado con muchas manos como objetos; llamados objetos positivos y objetos negativos, para el entrenamiento. Para buscar el objeto en la totalidad de la imagen, se mueve la ventana de búsqueda a lo largo de la imagen y así revisa cada sector. El clasificador ha sido diseñado para que pueda ser fácilmente “redimensionado” en orden de ser capaz de encontrar objetos de interés a diferentes tamaños, lo cual es más eficiente que

redimensionar la imagen por sí misma. Por lo tanto, para encontrar un objeto de tamaño desconocido en la imagen, la búsqueda se realiza varias veces a diferentes escalas.

La denominación “cascade” o cascada es el nombre resultante, que consiste en varios clasificadores simples que son aplicados subsecuentemente a una región de interés hasta que en alguna etapa el candidato es rechazado o todas las etapas son aceptadas.

Los clasificadores simples son árboles de decisión con al menos 2 ramas. Las características Haar son la entrada de dichos clasificadores.

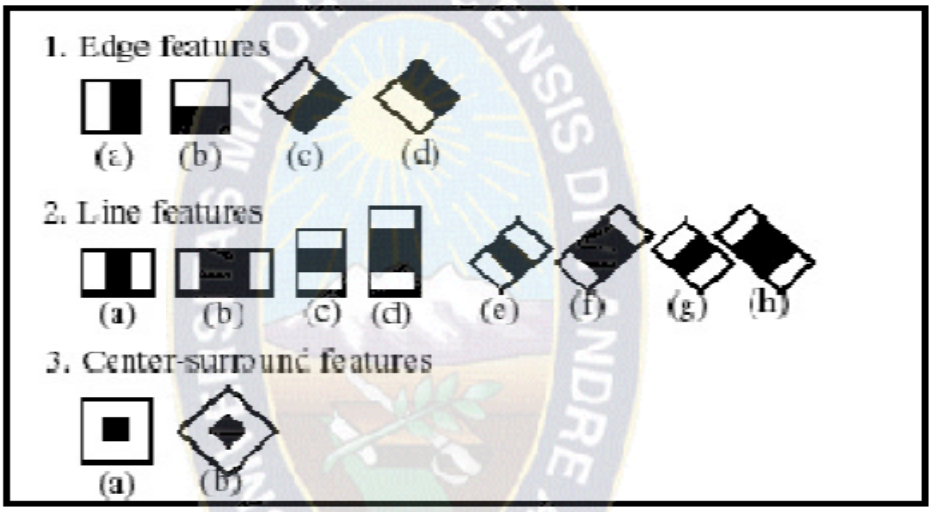

Figura 2.4: Características de Haar
Fuente: http://www.sebest.com.ar/?q=node/78

Claramente se muestra en la figura 2.4, un ejemplo de las características Haar. Las usadas en un clasificador en particular dependen de su forma (1a, 2b, etc.) posición en la región de interés y la escala (no es la misma escala usada en la etapa de detección). Por ejemplo, en el caso de la característica (2c) la respuesta es calculada como la diferencia entre la suma de los píxeles de la imagen bajo el rectángulo cubriendo la característica completa (incluyendo las franjas blancas y negras) y la suma de los píxeles de la imagen bajo la franja negra multiplicada por 3 en orden de compensar la diferencia de tamaños

entre áreas. La suma de los valores de los píxeles sobre las regiones rectangulares es calculada rápidamente usando imágenes integrales.

Extracción de Características

Las características de cada objeto se extraen al aplicar ciertas funciones que permitan la representación y descripción de los objetos de interés de la imagen (patrones). La extracción de características es un paso en el reconocimiento de patrones en el cuál las medidas u observaciones son procesadas para encontrar atributos que puedan ser usados para asignar los objetos a determinada clase. En la metodología seguida, la extracción de características es realizada aplicando a la imagen filtros con bases Haar. Estos filtros pueden ser calculados eficientemente sobre la imagen integral, son selectivos en la orientación espacial y frecuencia, y permiten ser modificados en escala y orientación. Los filtros con bases Haar, realizan una codificación de diferencia de intensidades en la imagen, generando características de contornos, puntos y líneas, mediante la captura de contraste entre regiones.

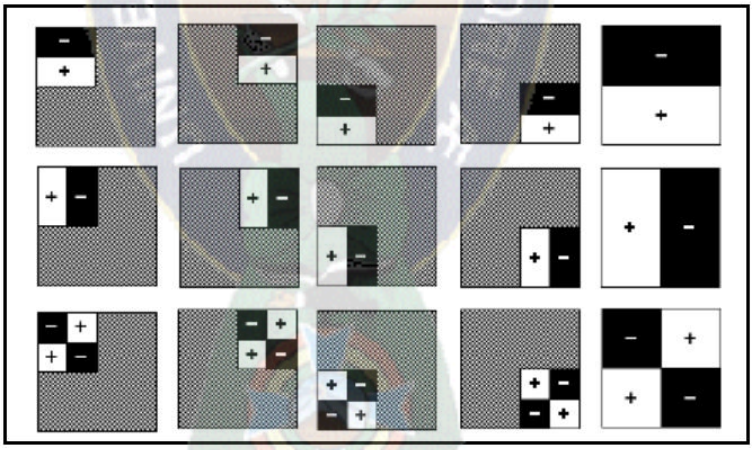

Figura 2.5: Filtros Haar rotados, trasladados y con cambios de Escala.

Clasificación

Esta etapa dentro del algoritmo de detección se encarga de asignar un conjunto de características dado a una clase con la que se encuentra una mayor similitud, de acuerdo

a un modelo inducido durante el entrenamiento. Boosting es un método de clasificación que combina varios clasificadores básicos para formar un único clasificador más complejo y preciso. La idea se basa en la afirmación de que varios clasificadores sencillos, cada uno de ellos con una precisión ligeramente superior a una clasificación aleatoria, pueden combinarse para formar un clasificador de mayor precisión, siempre y cuando se disponga de un número suficiente de muestras de entrenamiento.

En la figura 2.6, se muestra un esquema de un clasificador en cascada.

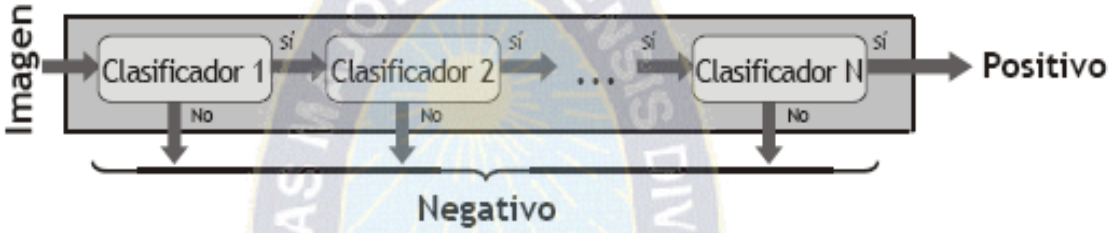

Figura 2.6: Clasificador en cascada.
Fuente: http://www.sebest.com.ar/?q=node/78

Para aplicar la técnica de AdaBoost se emplea un algoritmo de aprendizaje sencillo para encontrar reglas de clasificación muy precisas combinando con las de baja presicion, es decir una combinación lineal de clasificadores, que será llamado repetidas veces para crear diversos clasificadores base. Para el entrenamiento de los clasificadores base se emplea, en cada iteración, un subconjunto diferente de muestras de entrenamiento y una distribución de pesos diferente sobre las muestras de entrenamiento. Finalmente, estos clasificadores base se combinan en un único clasificador que se espera sea mucho más preciso que cualquiera de los clasificadores base por separado. En función de los clasificadores base que se utilicen, las distribuciones que se empleen para entrenarlos y el modo de combinarlos, podrán crearse distintas clases del algoritmo genérico de boosting.

2.5.2. Metodología de Haar Cascade

La metodología se divide en tres etapas como se muestra en la figura 2.7 en la primera se realiza una transformación de la imagen generando una nueva llamada imagen integral; en la segunda etapa se realiza la extracción de características usando filtros con base Haar; y por último se usa boosting para la construcción de clasificadores en cascada.

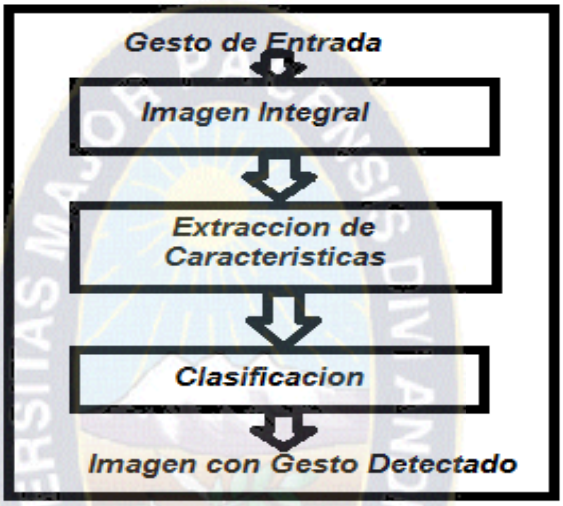

Figura 2.7: Etapas metodología Haar
Fuente: Elaboración Propia

La principal ventaja de una característica Haar como en la mayoría de otras características es la velocidad de cálculo que puede tener, debido a la utilización de imágenes integrales, donde esta función puede calcular en tiempo constante.

2.5.3. Función Haar

En el nivel más bajo, la cascada de Haar se compone de detectores de características que son regiones rectangulares de la imagen que se resumen, multiplicado por un coeficiente, y suman. Características Haar siempre tienen dos o tres de tales regiones. Las regiones exactas para resumir se especifican en el archivo XML en

cascada Haar dando las coordenadas superior izquierda del rectángulo, el ancho del rectángulo y altura, y el coeficiente, en ese orden. Por ejemplo:

```
<feature>
  <rects>
     <_> 3 7 14 4 -1. </ _>
     <_> 3 9 14 2 2. </ _>
   </ rects>
   <tilted> 0 </ inclinada>
</ Feature>
```

Especifica dos rectángulos: uno a partir de (3,7) y que se extiende 14 píxeles de anchura y 4 de altura, que se multiplica por -1, y el otro comienza a (3,9) y se extiende 14 píxeles de ancho y 2 de altura, y se multiplica por 2. Se verá que los dos rectángulos se superponen y forman un detector de borde horizontal como se muestra a continuación:

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1
2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1	2-1 = 1

Tabla 2.1 : Calculo de puntos del archivo XML
Fuente:Elaboracion Propia

La optimización de clave en el cálculo de estas sumas es rectangulares para formar una suma acumulada de la imagen de arriba a abajo y de izquierda a derecha.

Es decir, dada una imagen que de entrada (i, j) se forma la imagen

$$S (k , l) = \text{Sum} (I (i , j), i <= k , j <= l)$$

Podemos entonces formar la suma del rectángulo de partida en (x, y) y se extiende horizontalmente pixeles w y h pixeles verticalmente por computación

$$S(x + w, y + h) - S(x - 1, y + h) - S(x + w, y - 1) + S(x - 1, y - 1) \quad (1)$$

Es necesario tener en cuenta cuando se calcula rango de datos S (k, l). La imagen que de entrada (i, j) es un valor de 8 bits con signo. La suma acumulada de una imagen de píxeles 2Wx2H requerirá W + H 8 bits para la representación precisa y 2XW 2 xH 8 bits para el cálculo preciso de las operaciones en la imagen en su conjunto, como promedio. Esto significa que el tipo de datos limita el tamaño de la imagen.

Una vez que los rectángulos son sumadas deben ser multiplicados por los coeficientes. En OpenCV Haar cascada utiliza valores de punto flotante como coeficientes.

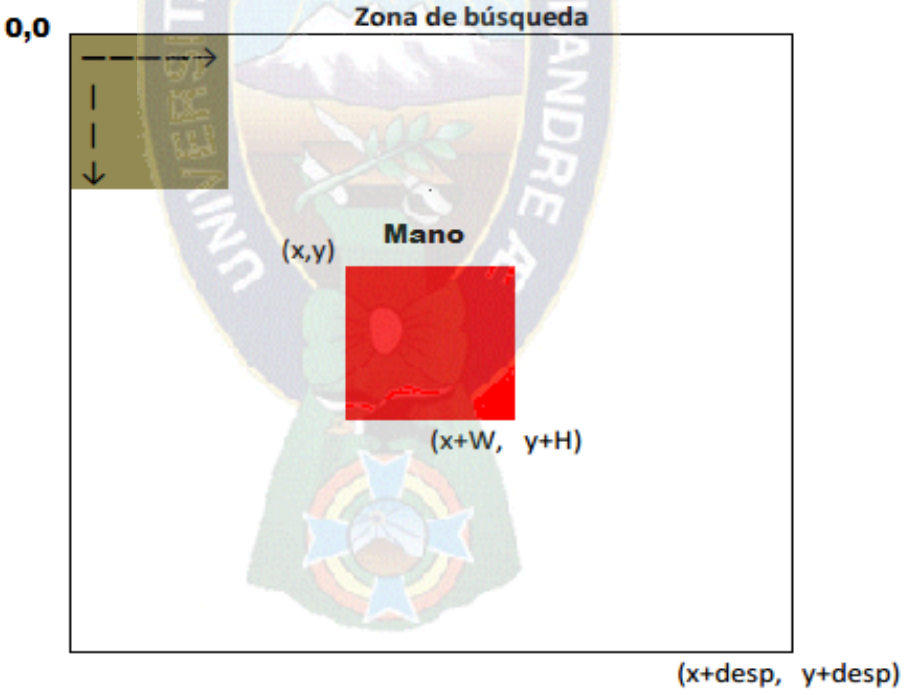

Figura 2.8: Representacion de una imgen seleccionada en un Marco.
Fuente: Elaboracion Propia

Una complicación final en el cálculo de la función de Haar proviene de una condición de contorno. Cuando un rectángulo está en el lado izquierdo o derecho de la parte superior de la sub-imagen (es decir, su coordenada x ó y es 0), la ecuación (1) se refiere a un valor de la suma acumulativa fuera de la sub-imagen, lo que lleva a una referencia de matriz no válido.

2.6. Conceptos generales

Debido a que este proyecto se enmarca en el campo de la visión computacional, es necesario introducir algunos de los elementos básicos de la representación de imágenes.

2.6.1. Clasificación de imágenes digitales

Dependiendo del rango de los valores que pueda tomar cada píxel podemos distinguir los siguientes tipos de imágenes:

Imágenes en blanco y negro: el rango está formado por los valores negro o blanco [0 1] únicamente. En la figura 2.9 se muestra una imagen de este tipo.

Figura 2.9: Imagen en blanco y negro

Imágenes 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].

2.6.2. Imagen Estática

Es una representación de un objeto, reales o ficticios, en un instante en el tiempo. Este tipo de imágenes se clasifican en dos categorías: imágenes vectoriales e imágenes de mapa bit o bitmaps Imágenes vectoriales.

Las imágenes vectoriales están compuestas por entidades geométricas simples: segmentos y polígonos básicamente. Cada una de estas entidades está definida matemáticamente por un grupo de parámetros (coordenadas inicial y final, grosor y color del contorno, color del relleno, etc.) Por compleja que pueda parecer una imagen, puede reducirse a una colección de entidades geométricas simples.

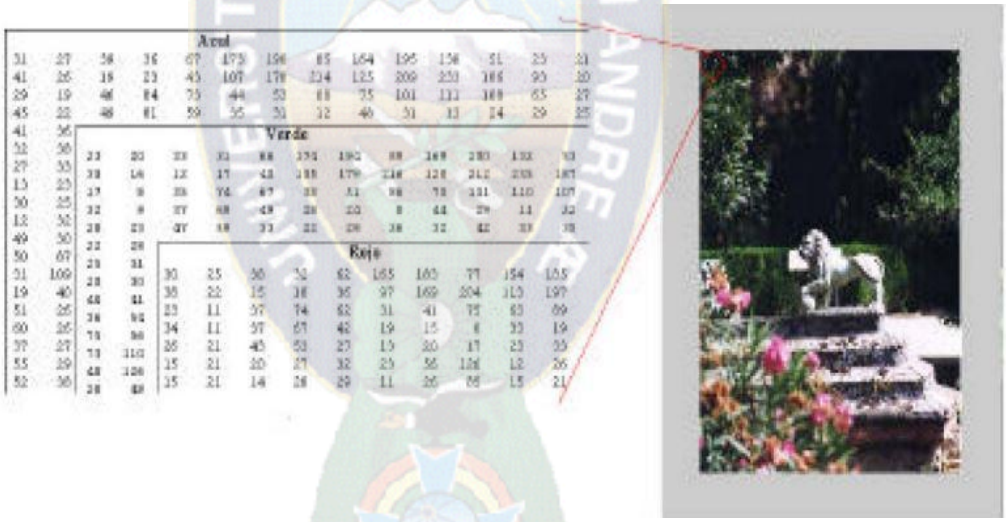

Figura 2.10: Imagen en color

2.6.3. Imágenes Vectoriales

Las imágenes vectoriales están compuestas por entidades geométricas simples: segmentos y polígonos básicamente (de hecho, una curva se reduce a una sucesión de segmentos). Cada una de estas entidades está definida matemáticamente por un grupo de

parámetros (coordenadas inicial y final, grosor y color del contorno, color del relleno, etc.) Por compleja que pueda parecer una imagen, puede reducirse a una colección de entidades geométricas simples.

2.6.4. Imágenes Bitmaps

Las imágenes en mapa de bits se las suele definir por su altura y anchura (en píxeles) y por su profundidad de color (en bits por píxel), creando una gran cantidad de cuadritos rellenos de un color uniforme, dando una sensación visual de integración en la retina, por la luminosidad entre pixeles vecinos. Las imágenes bitmaps no permiten el cambio de escala.

2.6.5. Imagen Dinámica

La imagen dinámica o en movimiento es en realidad un conjunto de imágenes estáticas denominadas cuadros de video que mostrados en secuencia rápida dan la idea de movimiento continuo.

El problema de la visión dinámica es que se compone de una secuencia de frames o fotogramas interrelacionados, donde cada fotograma representa la escena en un determinado instante de tiempo t.

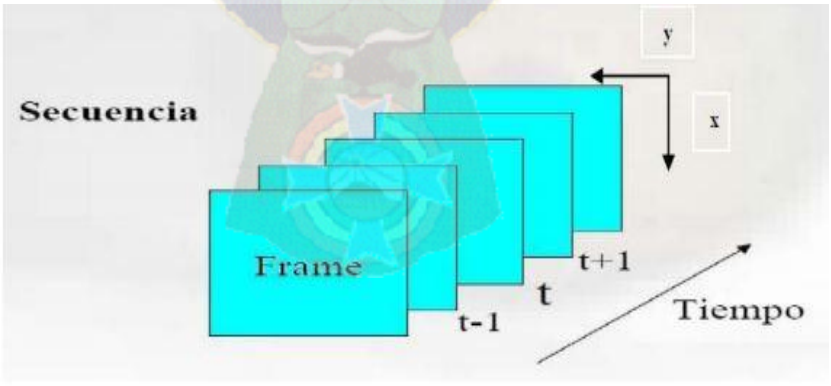

Figura 2.11: Fotogramas de una secuencia de imágenes a través del tiempo.

Debemos entender un sistema de visión dinámica como un conjunto de coordenadas espaciales (x,y) dentro de cada fotograma y además añadir un tercer parámetro como sería el tiempo t, permitiéndonos su combinación la descripción de la entrada de un sistema de visión dinámica como una función F(x,y,t).

La entrada en juego del parámetro t es fácil de entender si se tiene en cuenta los posibles cambios que pueden darse en la escena, ya sea debido al movimiento de la escena en sí misma, de la cámara o de ambos. En el trabajo que se trata, ese movimiento de cámara no va a producirse, debido a que las secuencias de imágenes serán captadas mediante una cámara estática.

Para procesar información dinámica se podrán aplicar técnicas estáticas a cada fotograma por separado o analizar una secuencia de forma continúa.

2.6.6. La Resolución De Una Imagen

La resolución de una imagen es la cantidad de píxeles que la componen. Suele medirse en píxeles por pulgada (ppi) o píxeles por centímetro (pcm). Cuanto mayor es la resolución de una imagen más calidad tendrá su presentación pero, desgraciadamente, más espacio ocupará en el disco el archivo gráfico que la contiene.

2.6.7. Transformada de la Imagen

Son operaciones que convierten imágenes desde una representación hacia otra diferente con el objetivo de hacer más evidente algún tipo de información existente de la imagen. Generalmente luego de una transformación sigue un proceso de umbralización para extraer las características más relevantes de la nueva representación.

2.6.8. Segmentación de Imágenes

Es un proceso que permite extraer información de una imagen; consiste en dividir a dicha imagen en diferentes regiones o áreas de acuerdo a un criterio que esta dado en

función de lo que se busca en la imagen, con esto se trata de separar las regiones de interés para posteriormente someterlas a un análisis o simplemente presentarlas.

2.6.9. Espacio de color: Modelo RGB

RGB es uno de los modelos más utilizados por los sistemas informáticos, con el que es posible representar un color mediante una mezcla de los tres colores de luz primario y reproducir los colores en el monitor. Está basado en la síntesis aditiva de las intensidades de luz relativas al rojo, al verde y al azul para conseguir los distintos colores; por lo que los mismos valores RGB pueden mostrar colores notablemente diferentes incluyendo el negro y el blanco.

El nombre del modelo RGB viene de las iniciales, en inglés, de esos tres colores: Red, Green y Blue.

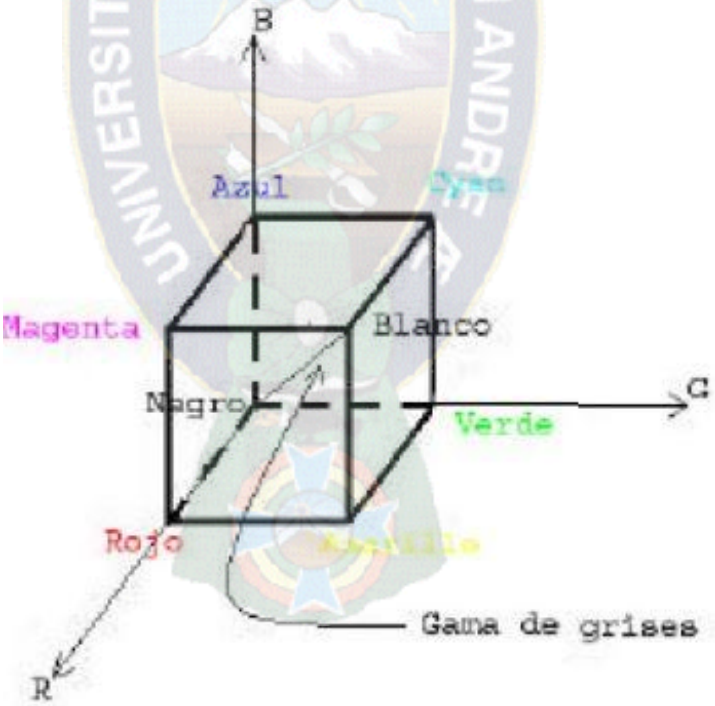

Figura 2.12: Representación gráfica del modelo de color RGB

La representación gráfica del modelo RGB (ver figura 2.12) se realiza mediante un cubo unitario con los ejes R, G y B. 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.

Las imágenes con modelo RGB utilizan tres planos de imágenes independientes, uno para cada color primario. Cuando estas tres imágenes son proyectadas a un monitor RGB, la pantalla de fósforo produce una imagen de color compuesto. Para el desarrollo del proyecto se usara una cámara de color para adquirir imágenes digitales, utilizando el formato RGB.

2.7. Procesamiento de Imágenes

Cuando se está procesando una imagen en el dominio espacial los pixeles son manipulados en el plano de la imagen, y la generación de un nuevo será una función del valor de cada píxel o del grupo de vecindad.

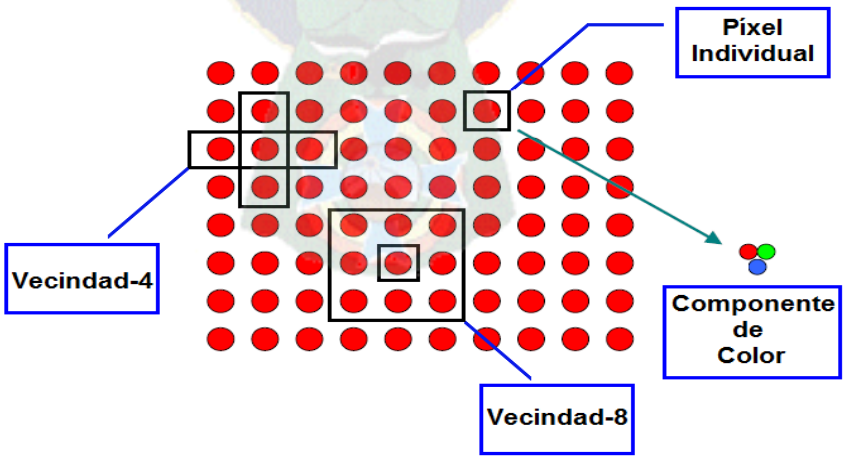

Figura 2.13: imagen de dominio espacial los pixeles

2.7.1. Operaciones Individuales

Se genera una nueva imagen de salida *q* a partir del procesamiento de cada píxel de una imagen de entrada *p*, descripción grafica ver (fig.2.14)

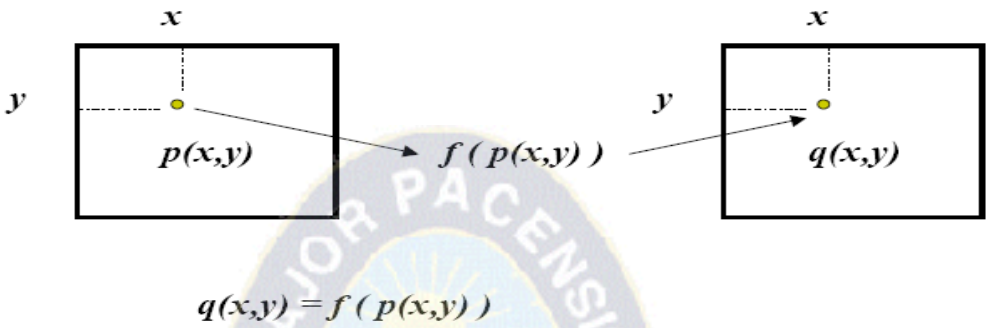

Figura 2.14:Funciones de operaciones

2.7.2. Operador Identidad

Se crea una imagen de salida *q*, idéntica a la entrada *p*.

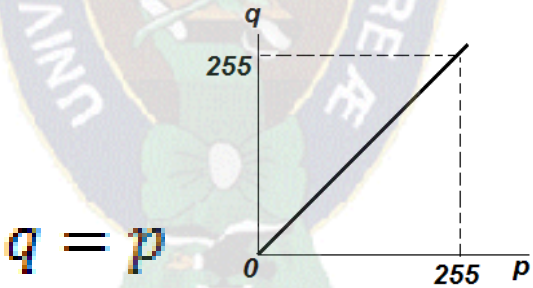

Figura 2.15:Operador Identidad

2.7.3. Operador Umbral

Crea una imagen de salida *q* con resolución en amplitud 1bpp (binarizada), a partir de un umbral.

$$q = \begin{cases} 0 & para\ p \leq p1 \\ 255 & para\ p > p1 \end{cases}$$

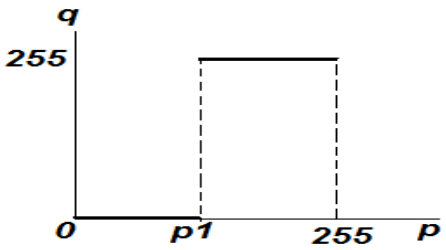

Figura 2.16: Operador Umbral

2.7.4. Operador de umbral en escala de grises

Crea una imagen de salida *q* donde permanecen los pixeles con niveles de gris en el intervalo definido por p1 y p2.

$$q = \begin{cases} 255 & para\ p \leq p1\ o\ p \geq p2 \\ p & para\ p1 < p < p2 \end{cases}$$

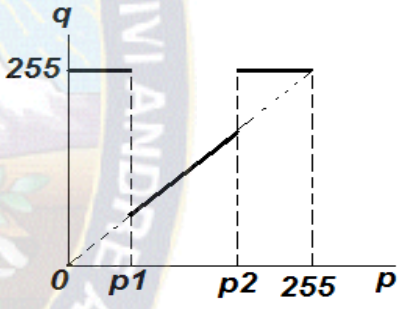

Figura 2.17: Operador escala de grises

2.8. Operaciones Aritméticas entre Imágenes

En muchas aplicaciones como el procesamiento de video en tiempo real, se involucran varias imágenes o frames para el procesamiento de una imagen resultado, esta imagen resultado se obtiene mediante operaciones lógicas, aritméticas ó geométricas de 2 o más imágenes, para que puedan efectuarse pixel a pixel y se conocen mejor como operaciones didácticas, las cuales operan en los valores correspondientes de los pixeles. Entre estas se encuentran la suma, la resta, la multiplicación y división de imágenes

Se debe tener un especial cuidado cuanto se realizan estas operaciones, porque en muchísimos casos se van a generar imágenes con un tamaño diferente y con niveles de intensidad fuera del rango de resolución en amplitud de la imagen.

2.8.1.Suma de Imágenes

La operación suma es utilizada frecuentemente por programas de diseño gráfico en el montaje y retoque de imágenes.

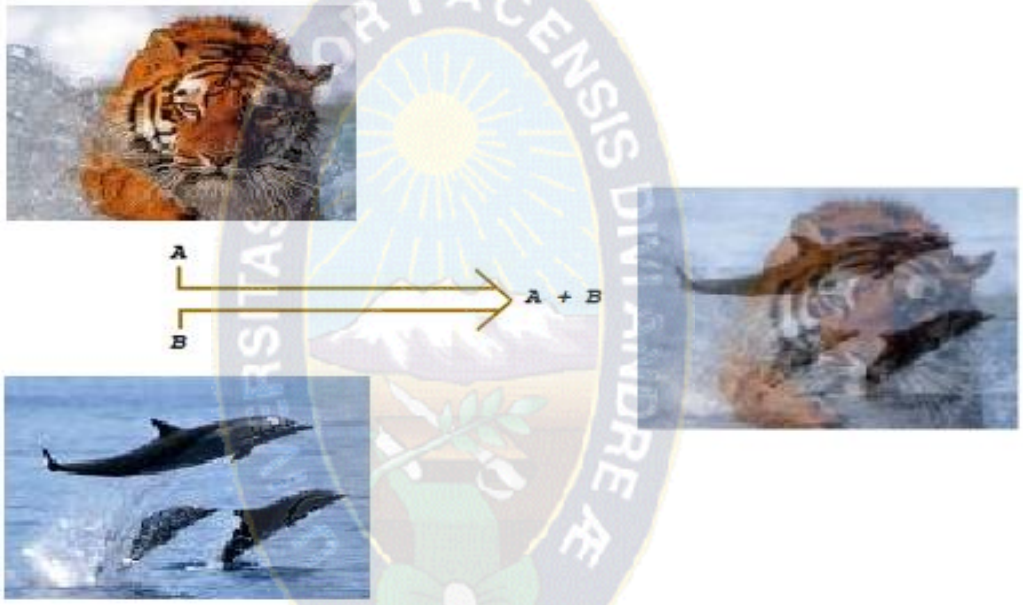

Figura 2.18: Suma de imágenes
Fuente: Elaboración propia

Al realizar la operación suma es posible que se genere un posible overflow de los niveles de intensidad o saturación esto debido a que los niveles de intensidad de cada imagen pueden variar de donde n es igual a la resolución en amplitud de la imagen y al realizar la suma se puede obtener un valor máximo de intensidad. Para contrarrestar esta situación se puede realizar una normalización o una limitación de intensidades.

Normalización: $\frac{(A+B)}{2}$

Limitación: $g = \begin{cases} x+y, & x+y < 2^n - 1 \\ 2^n - 1, & x+y \geq 2^n - 1 \end{cases}$

2.8.2. Resta de imágenes

La operación resta es la solución más simple para la detección de movimiento. Cuando se realiza la resta entre 2 imágenes de un mismo escenario tomadas consecutivamente, se obtendra una imagen resultado que permite visualizar los elementos que han cambiado de posición.

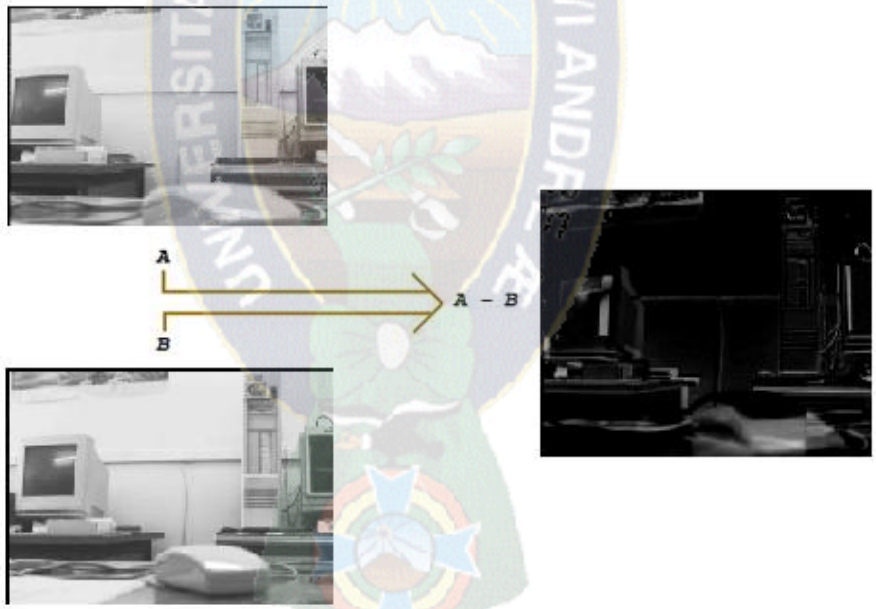

Figura 2.19: Resta de Imágenes

Fuente: Elaboración propia

2.8.3. Multiplicación de imágenes

La operación multiplicación se puede realizar mediante la formula $g = k * x * y$, donde $k = 1/(2^n - 1)$, esto para evitar el overflow de amplitud y por consiguiente la saturación de la imagen.

Figura 2.20: Multiplicación de Imágenes
Fuente: Elaboración propia

2.9. Imagen Integral

Esta imagen permite extraer de forma rápida características a diferentes escalas ya que no se trabaja directamente con los valores de intensidad si no con una imagen acumulativa que se construye a partir de operaciones básicas.

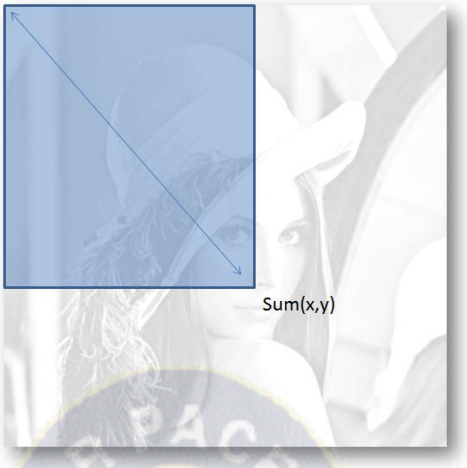

Figura 2.21: Imagen Integral.

La imagen integral (figura 2.21), en la localización x, y, contiene la suma de los píxeles de la parte superior izquierda de la imagen y se puede calcular como se indica a continuación:

$$II(x,y) = \sum_{x' \le x, y' \le y} Im(x', y')$$

Donde II(x,y,) es la imagen integral e Im(x,y) es la imagen original.

CONSTRUCCIÓN DEL MODELO DE RECONOCIMIENTO GESTUAL

3.1. Introducción

EL diseño del modelo comienza con la descripción de investigación sobre las librerías y herramientas existentes que pudieran facilitar en todo lo posible la tarea de reconocimiento de gestos. Se ha realizado una búsqueda generalizada, considerando cualquier plataforma, valorando las ventajas e inconvenientes de cada una de ellas, para poder realizar una visión computacional de manera más rápida y efectiva. La siguiente tabla muestra sus descripciones de las herramientas que se puede manejar para el reconocimiento gestual en diferentes plataformas:

Del analisis de cada una de las herramientas que exite, elaboradad en la tabla 3.1 de comparaciones y la descripcion obtenida por el analisis de cada una de las librerias , podríamos destacar dos: OpenCV y AForge.NET, ya que son las mas completas que existen, existen otras de igual manera de manera efectiva que es el kinect, a razón de que implica un costo aparate de la PC, no se estudiara la detección de gestos con el kinect.

3.2.Programa TEST

Para probar la deteccion de manos usando la libreria OpenCV haremos un programa de prueba que llamaremos openpalmaa. Este programa recibiria como entrada una imagen y devolvera la misma imagen con la mano detectada marcada en recuadro**.**

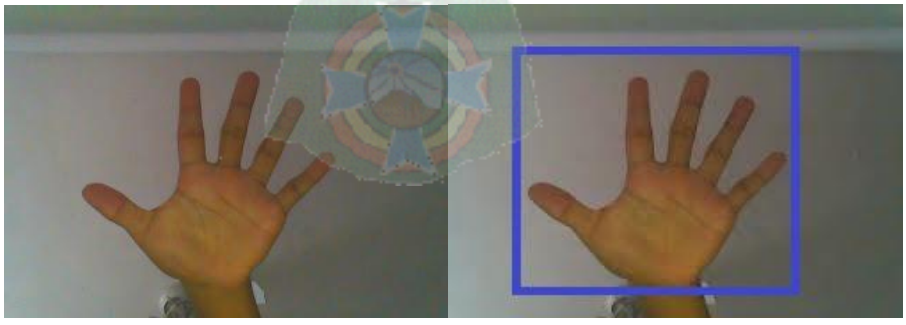

Figura 3.1: Ejemplo de reconocimiento de la mano
Fuente: Elaboracion Propia.

Librerías	Lenguaje	Descripción	Ventajas	Desventajas
OpenCV	C/C ++, .NET	Opencv permite el procesamiento y análisis de imágenes.	Open Sourse, Rápida y eficiente. Disponible para Windows, Mac, Linux.	Nivel de dificultad alto y escasa documentación
EmguCV	.NET	EmguCV se trata de un wraper sobre OpenCV, que permite utilizar la libreria.	Hace procesamiento de imágenes y cosas relacionadas con visión artificial.	Dependiente de OpenCV.
OpenArch	C# o C++	Asociada a la domotica, para crear un prototipo real de vivienda inteligente.	basado en sistemas de hardware y software open source,	Nivel de dificultad alto, no hay documentación para el desarrollo.
AForge.NET	.NET	Facil uso	codigo libre, plataforma de desarrollo requerido (Visual Studio) es robusta y de libre acceso.	No es muy eficiente, solo para el SO Windows. No está del todo centrado en la visión por Ordenador
Kinect	C# , VB	El único coste material que implica este dispositivo es la cámara Kinect y el cable para poder conectarla por usb a un ordenador.	Movimientos precisos.	Esta basado en Wiimote.
Air Gesture		Renoce movimientos de la mano sobre el telefono sin llegar a tacar la pantalla		Solo para el SO Android.
Leap Motion	C++, C #, Java, Python.	Leap Motion, un pequeño gadget que te permite controlar la computadora con gestos y movimientos de las manos.		OStware y hadware ya diceñado para el mercado.
VisionLab VCL	C++ / delphi	Permite capturar y grabar video, además de detectar objetos.	Permite crear aplicaciones fácil y rápidamente.	Comercial (30 días de prueba)

Tabla 3.1: Tabla de comparaciones

Fuente: Elaboracion Propia

3.2.1. Diseño del clasificador TEST

En su versión actual, la librería OpenCV dispone de recursos para la clasificación de rostros, empleando distintos algoritmos ya desarrollado en los que estan en la libreria de opencv, como no se tiene incorporado un clasifer que haga la deteccion de la mano la desarrollaremos basandonos en lo mismo que se aplico para la deteccion de rostros, usando el algoritmo de Viola-jones. Primero el entrenamiento, en la que a una serie de filtros en cascada se les pasan unos patrones positivos (que coinciden con el objeto buscado) y negativos (no tienen el objeto) de forma que el sistema aprenda y se forme un modelo de las características de Haar del objeto a detectar. Éste clasificador se ha conseguido por medio de un aprendizaje guiado como mostraremos el desarrollo más adelante.

3.2.2. Explicación Entrenamiento

El aprendizaje o entrenamiento consiste en introducir una muestra significativa de imágenes positivas y negativas para que el algoritmo sea capaz de diferenciar lo que realmente queremos, se necesitar alrededor de 3000 imagenes para poder identificar bien qué es una mano. Para obtener las imágenes que necesitaremos para el aprendizaje usaremos algunas bases de datos públicas que subministran por medio de subscripción o simplemente de manera libre algunas conjuntos de imágenes. Dentro de estos organismos se encuentran en su mayoría universidades con laboratorios de investigación en Visión por Computador que han tenido que desarrollar una base de datos de caras para sus investigaciones.

Para entrenar un clasificador que funcione en primer lugar se debe obtener imágenes positivas que indiquen donde se encuentra el objeto que se quiere obtener, en nuestro caso introduciremos palmas de manos frontales indicando donde se encuentra la palma dentro de la imagen con un formato adecuado, además también necesitaremos imágenes negativas, es decir, cualquier cosa que no sea la imagen de una palma de mano,

permitiendo descartar más fácilmente las subimágenes que puedan parecerse a una mano.

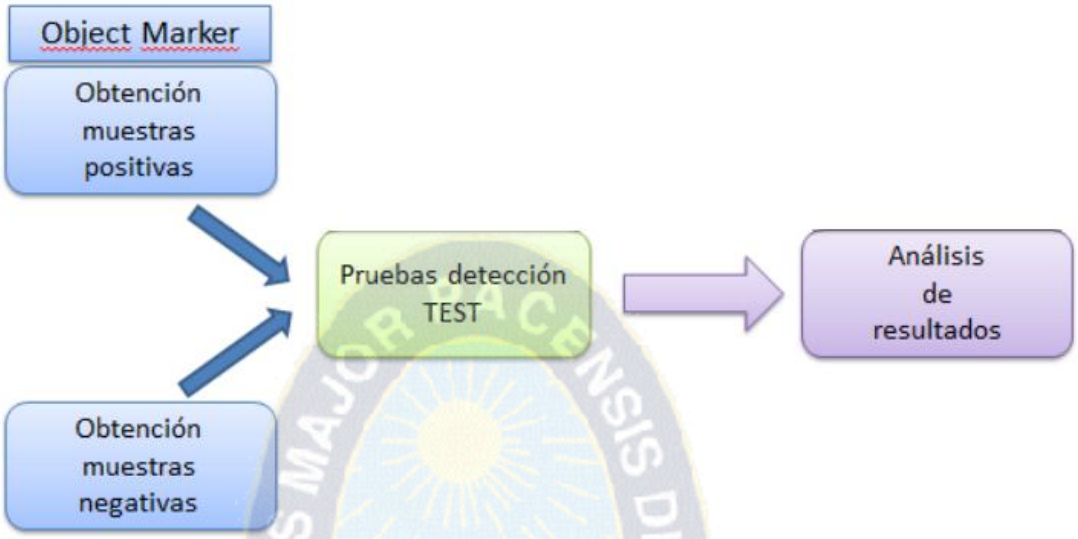

Figura 3.2: Metodologia Haartraining.
Fuente: Elaboracion Propia.

Como el algoritmo de Viola-Jones utiliza un árbol de decisión, si podemos en los primeros pasos descartar el mayor número posible de subimágenes conseguiremos una identificación más rápida y más efectiva, y además reduciremos los falsos positivos(imagenes que no tienen el objeto que se quiere identificar). Una vez hecho esto deberemos definir los parámetros configurables que permite la función haar-training de OpenCV en función del tipo de clasificador que queremos obtener.

```
./haartraining
-data <dir_name>
-vec <vec_file_name>
-bg <background_file_name>
[-npos <number_of_positive_samples = 2000>]
[-nneg <number_of_negative_samples = 2000>]
```

```
[-nstages <number_of_stages = 14>]
[-nsplits <number_of_splits = 1>]
[-mem <memory_in_MB = 200>]
[-sym (default)] [-nonsym]
[-minhitrate <min_hit_rate = 0.995000>]
[-maxfalsealarm <max_false_alarm_rate = 0.500000>]
[-weighttrimming <weight_trimming = 0.950000>]
[-eqw]
[-mode <BASIC (default) | CORE | ALL>]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
[-bt <DAB | RAB | LB | GAB (default)>]
[-err <misclass (default) | gini | entropy>]
[-maxtreesplits <max_number_of_splits_in_tree_cascade = 0>]
[-minpos <min_number_of_positive_samples_per_cluster = 500>]
```

Figura 3.3: Funcion modificable del haar-training de OpenCV, para el clasificador.
Fuente: Elaboracion Propia

Dentro del entrenamiento se define la tasa de fallo que se puede permitir y la tasa de acierto. Una tasa de fallo demasiado baja no permitiría avanzar hacia soluciones locales válidas, es por ello que una tasa adecuada puede ser del 0.5, ya que descartaríamos la mitad de las posibles subimágenes en cada uno de los pasos.

En cuanto a la tasa de acierto pues podemos permitir tasas de acierto bajas para que la clasificación se haga más rápida, en cambio si queremos que nuestro análisis se haga de manera exhaustiva deberemos utilizar tasas de acierto elevadas, cercanas al 100%, esto implica que a mayor datos recorridos mayo acierto y un poco mas lento.

También debemos tener en cuenta el parámetro -sym (default)] [-nonsym]. En caso de que nuestro objeto no fuera simétrico horizontalmente podríamos usar -nonsym que

obtiene un procesado más costoso. Si utilizamos la opción por defecto -sym la clasificación se realizará sólo sobre una de las mitades horizontales de nuestro objeto, de manera que ahorramos la mitad del procesado.

Además tenemos que considerar -ALL para activar el uso de todas las características de Haar. Si no se utiliza este parámetro el entrenamiento sólo utiliza las características de haar verticales y nos conviene que se usen las horizontales, verticales y las diagonales para identificar nuestros objetos. Esto provoca un aprendizaje más lento pero también más elaborado.

3.2.3.Valoracion Final del Clasificador

Despues del estudio del clasificador y ver los resultados en la figura 3.1, podemos determinar que hemos realizado estudio sobre la clasificacion de manos y la certeza de poder generar un entrenamiento guaido que nos ha llevado a un clasificador de manos badaso en las caracteristicas de Haar y ADABoost.

Antes de usar nuestro clasificador se sugiere hacer una iteracion mas, para la reduccion de falsos positivos y hacerlo mas robusto.

3.3. Modelación

El proceso para mostrar la interacción del usuario a través de la mano con el computador se basa en el procesamiento de una secuencia de imágenes (frames) obtenidas a partir de la cámara de video. Este proceso se divide en cuatro etapas principales: detección inicial de la posición de la mano, obtención de las características descriptivas de la misma, su seguimiento, e interpretación de la información obtenida en las etapas anteriores

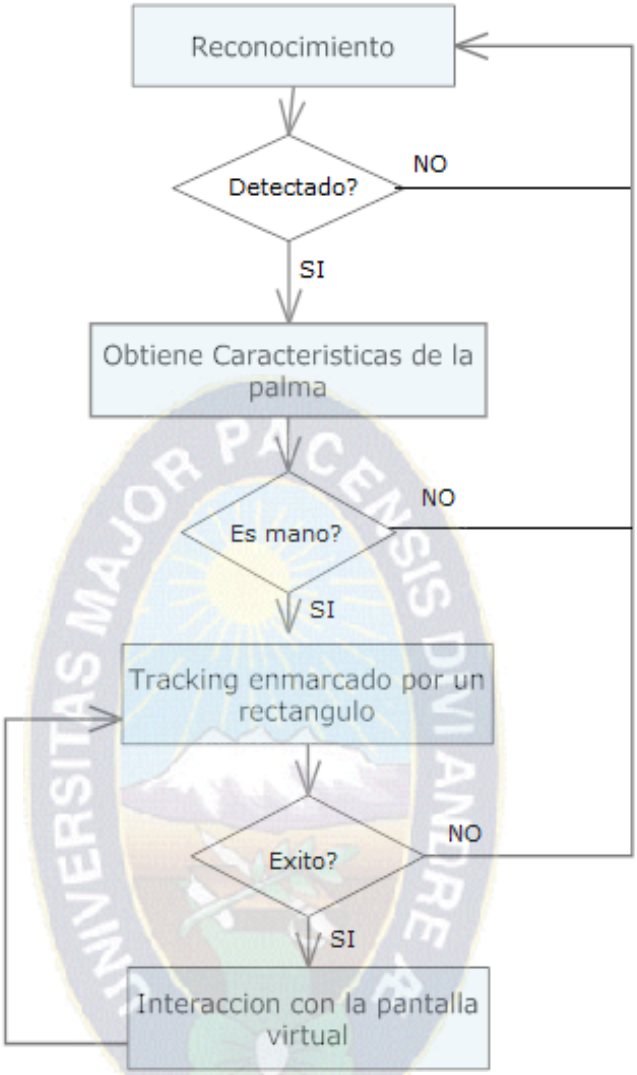

Figura 3.4 : Diagrama de Flujo

Fuente: Elaboración Propia

3.4. Descripción de la Librería AForge.NET

El autor de esta librería es Andrew Kirillov nos permite realizar toda clase de operacione s, desde un sencillo cambio de color a blanco y negro, es un marco de código abierto de C # diseñado para desarrollado e investigación en los campos de Visión por Computador e Inteligencia Articial, procesamiento de imágenes.

AForge.NET no se limita solamente al tratamiento de imágenes, contiene un conjunto de librerías y programas.

Características de AForge.Net
• AForge.Imaging - biblioteca destinada a rutinas de procesamiento de imágenes y filtros; • AForge.Vision - Biblioteca para el desarrollo de visión computacional, utilizando la deteccion de movimiento y el procesamiento de secuencias de fotogramas de video. • AForge.Video - conjunto de librerías para el procesamiento de vídeo. • AForge.Neuro -Incluye interfaces, redes neuronales biblioteca de computación. • AForge.Genetic - Contiene rutinas comunes del campo de la porgramacion genetica. • AForge.Fuzzy - fuzzy biblioteca de cálculos fuzzy y operacion de conjuntos difusos. • AForge.Robotics - Apoyo de algunos kits de robótica. • AForge.MachineLearning - Biblioteca que incluye diferentes algoritmos de aprendizaje automatico.

Tabla 3.2: Descripcion de blibliotecas de Aforge.net

Fuente: Elaboracion Propia.

En la actualidad el marco de Aforge consta con 3 partes importantes de algoritmos de detección de movimiento:

- El procesamiento de imágenes
- Las redes neuronales
- Algoritmos Evolution

Para la detección de movimiento en los fotogramas de video, los algoritmos que utiliza esta biblioteca son:

- Diferencia de frames.
- Modelación de fondo Simple.
- Diferencia de frames personalizados

La libreria de AForge.Net realiza la deteccion de movimiento capturando a traves de una entrada de video, y se identificara los movimiento de los gestos llevado a cabo por el usuario.

Al manejar el algoritmo de Diferencia de frames, maneja las adicion y resta de los fotogramas, para el reconocimiento de movimiento la diferencia de frames personalisados implica mayor tiempo de proceso al hacer uso de cada frame haciendo comparaciones de frames. Solo hace el reconocimento de movimiento al hacer comparaciones pero no hace reconocmiento gestual no hace por que solo maneja imaganes pixel a pixel.

3.4.1.Codigo para la deteccion de moviento.

El codigo compilacion para la deteccion de movimiento, realizado con la libreiria AForge.Net se puede ver en el anexo b. La clase procesa imagen especificada del cuerpo humano , tratando de reconocer las manos procesando solo (8 bpp indexado) imágenes binarias , donde el color blanco, está representado por valor de 255 y negro es 0, el valor establece la proporción mínima del ancho de la mano directamente a su altura. Si la mano en un procesamiento de imágenes tiene proporción mayor que este valor, el de la mano se considera recto . El valor mínimo posible es 0 el valor predeterminado se establece en 1,3 Maximal levantó proporción mano. El valor establece la proporción máxima de espacio libre por encima de la mano levantada Si la proporción de espacio libre por encima de la mano de la imagen es inferior a este valor , la mano se supone que Plantearse diagonal . de lo contrario, se supone que debe ser elevado en diagonal valor mínimo posible es 0 . El algoritmo de procesamiento primero de todo intenta localizar objeto grande , que es presumiblemente la mano del cuerpo, cuando el objeto no se mueve por un tiempo, los algoritmos intenta reconocer su gesto. En el caso de

reconocimiento exitoso , la clase dispara un evento de que realizo un movimiento hacia la izquierda o derecha.

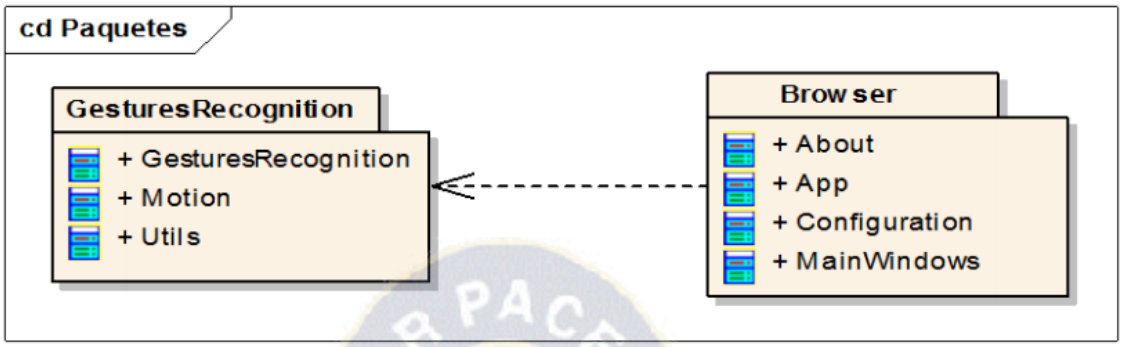

Figura 3.5: Diagrama de Paquetes de la libreeria AForge.Net
Fuente: Elaboracion Propia.

Una de las caracteristicas que puede interferir en el correcto funcionamiento de la libreria, es la calidad de la imagen utilizada para el reconocimiento de moviento, la investigacion del algortimo de deteccion de movimiento realiza un recorrido de la imagen pixel a pixel, por lo que a mayor calidad, mayor sera el numero de pixeles, y por lo tanto el tiempo necesario para analizar la imagen sera mas alto, al igual en una calidad de la imagen baja pordria llevarnos a una perdida de informacion sobre el movimiento. Es un marco excelente para el reconocimiento de objetos, pero no es preciso para el reconocimiento de gestos.

3.5. Descripcion de la Librería OpenCV

Mientras que OpenCV implementa una gran variedad de herramientas para la interpretación de la imagen, es principalmente una librería que implementa varios algoritmos para las técnicas de la calibración de la camara, detección de rasgos, para rastrear, análisis de la formas geométricas, análisis del movimiento, reconstrucción 3D, segmentación de objetos y reconocimiento de gestos, rostros, etc. OpenCV usa la estructura IplImage para crear y manejar imágenes, manejando una gran cantidad de campos, entre ellos por ejemplo el width es la anchura del Iplimage, height es la altura,

depth es la profundidad en bits y nChannels el número de canales, viene con una interface gráfica llamada highGUI, esta interfaz gráfica es muy importante para la visualización de imágenes.

3.5.1. Codigo para deteccion Gestual

Describe este apartado el modo de trabajo con OpenCV para la adquisición de imágenes en vivo desde una cámara, la introduccion del clasificador para el reconocimento, una pantalla virtual de interaccion de eventos. Lo típico introductorio de OpenCV pasa por cargar y mostrar por pantalla lo que recoge de nuestra webcam.

Es importante destacar en la implementación que el uso de la funcióncvWaitKeyes muy importante que se ejecute en el bucle principal, ya que es la única forma del módulo HighGUI(las funciones de alto nivel de OpenCV) para devolución y manejo de eventos. Por ello, debe ser invocado de forma periódica para un proceso de eventos normal. El codigo se puede obtener en el anexo C.

3.6. Frames por segundo

Tras analizar los resultados que se veía en la detección de imágenes y movimiento, es el numero de frames por segundo de video capturado, que también a sido motivo de estudio, por defecto un dispositivo de video utiliza por lo general 24 fps en condiciones normales, sin embargo este valor es demasiado alto para analizar cada una de las imágenes e identificar el movimiento, lo que producirá un aumento del tiempo de respuesta. Además se ocurre lo contrario que un numero de frames muy bajo, puede dar lugar a no percibir el movimiento, es decir si el movimiento se realizar de forma rápida, es posible que se llegue a capturar dos o incluso un solo frame durante el proceso de detección de movimiento.

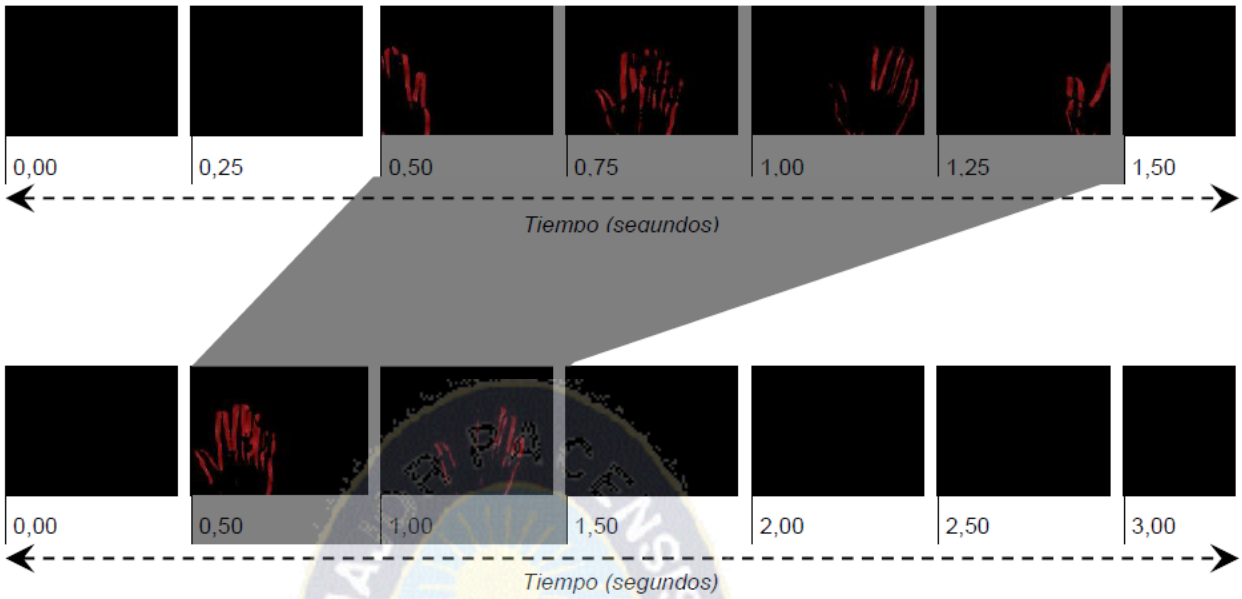

Figura 3.6: Ejemplo de movimiento ante diferentes valores de frame.

Representación de comparación perdida de información al aumentar o disminuir el numero de frames capturados, una de las opciones en OpenCV para el manejo de frames es:

video.set(CV_CAP_PROP_FPS, fps);

Mientras que en la libreria de AForge.Net depende mucho de la calidad de la imagen que se quiera detectar movimiento.

Hay que tener en cuenta además, que el dispositivo no podrá proporcionar el valor exacto, dependerá de sus capacidades, rendimiento del sistema, etc.

PROTOTIPO PRUEBAS Y RESULTADO

4.1. Introduccion

Como se ha descrito, previo a comenzar el desarrollo de las pruebas de reconocimiento gestual, para la iteraccion por computador, con realidad aumentada, se ha realizado un estudio de las librerias, que a detectar cual es la mas adecuada para el desarrollo, era necesario identificar que nos permite realizar cada libreria y que características podríamos llegar a conseguir durante el desarrollo.

4.2. Planificación.

Es por esa clasificación que se hizo para trabajar con la libreria Opencv, siendo muy extensa, proporcionándonos multitud de funciones para el manejo y manipulacion de imagenes, vision artificial, muchos datos de alto nivel como juegos, arboles, graficos, matrices , etc y se expondrá en forma detallada el método para la identificación gestual, tratandose básicamente de la detección de la mano. Uno de los mayores riesgos a considerar en el desarrollo de aplicación es el tiempo.

Ya que implica el estudio de nuevas herramientas que anteriormente no había utilizado como ser la librería que se eligió, y mejorar los conocimientos en el lenguaje de programación C++.

4.3. Plataforma de desarrollo

Para el desarrollo se utilizó el lenguaje C++, considerando las siguientes características básicas requeridas para el objetivo planteado:

• Necesidad de contar con un lenguaje potente para procesamiento matemático y gráfico.

• Orientado a objetos con el fin de dar modularidad a las distintas etapas del procesamiento de imágenes y visualización de resultados.

Para determinar el IDE (ambiente de desarrollo) más adecuado, se analizaron varias alternativas principalmente del ámbito código abierto, descartando el uso de productos dependientes del pago de licencia, principalmente para no limitar la eventual continuidad del desarrollo futuro sobre el presente trabajo, eligiendo Visual Studio, básicamente debido a que:

• Cuenta con una evolución de versiones continuas que permite suponer un nivel de madurez

• Contiene nativamente en compilador y librerías implementadas en Windows

Para el manejo y procesamiento de imágenes se determinó el uso de la biblioteca de código abierto para desarrollo de visión artificial OpenCV. Para la interacción de sonido se uso la librería Audiere.

4.4. Desarrollo Insercion del la mano al entorno con OpenCV

Implícitamente para probar la detección de manos usando la librería OpenCV, se hizo un clasificador que llamamos openpalma.xml, este recibirá como entrada una imagen y devolverá la misma imagen con las manos detectadas marcadas en recuadros, en cualquier punto de la pantalla.

El desarrollo se componen por un conjunto de paquetes como son:

cv::CascadeClassifier openpalma;

Para llevar a cabo una clasificación de imágenes buscando por el método de Viola-Jones algún patron dentro de una imagen necesitamos un clasificador en cascada. Dentro de opencv podemos declarar un clasificador cv::CascadeClassifier y después utilizarlo con algún propósito.

face_cascade.load(face_cascade_name)

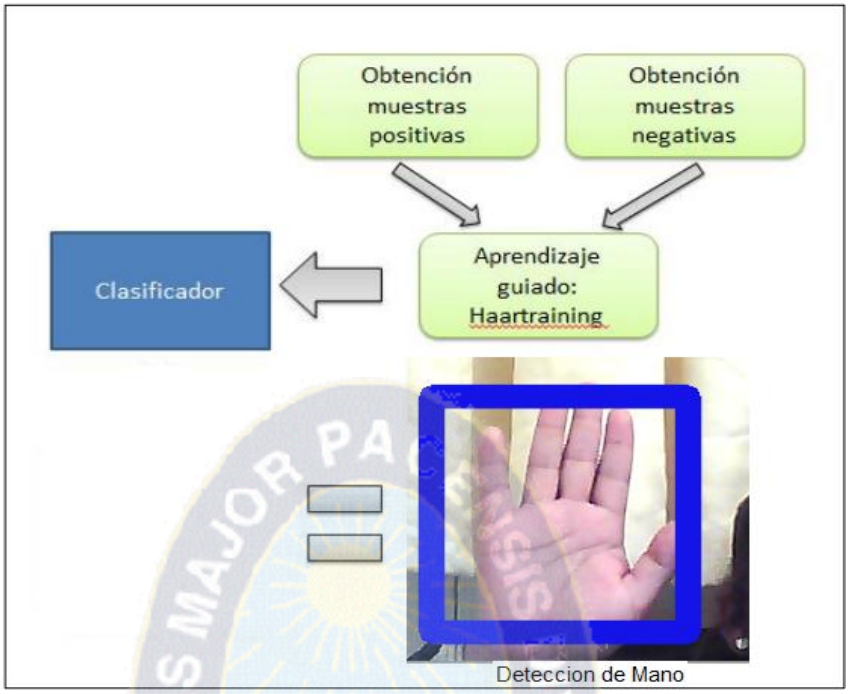

Figura 4.1:Detección de mano.
Fuente: Elaboración Propia.

Los clasificadores en cascada están definidos en un formato concreto dentro de OpenCV, este formato está incluido dentro de ficheros xml que determinan cuantos pasos tiene el clasificador y que características de Haar utiliza. OpenCV incorpora dentro de la librería algunos operadores y para utilizarlos debemos cargar el clasificador dentro del objeto cv::CascadeClassifier indicando la ruta del fichero en el parámetro name.

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

La función detectMultiScale lleva a cabo la detección de objetos dentro de una imagen. Los objetos encontrados son devueltos en un vector de Rectángulos que indican la localización y el tamaño.

Parámetros:

• image – Matriz de tipo CV_8U que contiene una imagen en la que se detectan objetos.

• objects – Vector de rectángulos que pasado por referencia retorna un objeto detectado por cada rectángulo.

• flags – Este parámetro no se utiliza en los nuevos clasificadores en Cascada y tiene el mismo significado que en la función cvHaarDetectObjects. La detección de manos está optimizada por CV_HAAR_SCALE_IMAGE más que por el método por defecto, 0 CV_HAAR_DO_CANNY_PRUNING

• MinSize – El espacio mínimo posible de objeto. Objetos más pequeños que eso serán ignorados.

Por último necesitamos pintar los resultados de los objetos obtenidos por detectMultiScale:

void rectangle(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0) dibuja un rectángulo con el tamaño de línea y tamaño indicado en la imagen introducida por parámetros.

Parámetros:

- img – Imagen donde queremos pintar.
- pt1 – Uno de los vértices del rectángulo.
- pt2 – El punto opuesto a pt1 formando una de las diagonales del rectángulo.
- color – Color del rectángulo o brillo en imagen en escala de grises.
- shift – Número de bits fraccionarios en las coordenadas del punto, Ejem. cvRectangle(images, pt1, pt2, CV_RGB(20,20,232), 15, 15, 0.5).

Figura 4.2: Reconocimiento de la Mano

Fuente: Elaboracion Propia

4.5. Módulo de Inserción de Video

Para procesar una secuencia de vídeo, tenemos que ser capaces de leer cada uno de sus cuadros. OpenCV ha puesto en marcha una herramientas fácil de usar, marco para llevar a cabo la extracción de fotograma de los archivos de vídeo, o incluso de una cámara USB, el único problema que enfrentamos es que necesitamos algún tipo de bucle para leer cada fotograma de la secuencia por ejemplo.

cv::VideoCapture()	**cvCreateFileCapture()**	**cvCaptureFromCAM()**

#include “highgui.h”
int main()
{
cv::VideoCapture capture("../bike.avi");
if (!capture.isOpened())
return 1;
double rate=
	#include “highgui.h”
	int main(int argc, char** argv) {
	cvNamedWindow(“Example”, CV_WINDOW_AUTOSIZE);
	CvCapture* capture =
		#include <highgui.h>
		int main() {
		CvCapture* capture = cvCaptureFromCAM(CV_CAP_ANY);
		if(!capture) {
		fprintf(stderr, "ERROR: no se puede abrir el dispositivo.\n"

capture.get(CV_CAP_PROP_FPS);
bool stop(false);
cv::Mat frame; // current video frame
cv::namedWindow("Extracted Frame");
int delay= 1000/rate;
while (!stop) {
if (!capture.read(frame))
break;
cv::imshow("Extracted Frame",frame);
if (cv::waitKey(delay)>=0)
stop= true;
}
capture.release();
}
	cvCreateFileCapture(argv[1]);
	IplImage* frame;
	while(1) {
	frame = cvQueryFrame(capture);
	if(!frame) break;
	cvShowImage(“Example”, frame);
	char c = cvWaitKey(33);
	if(c == 27) break;
	}
	cvReleaseCapture(&capture);
	cvDestroyWindow(“Example2”);
	}
		); getchar(); return -1;}
		IplImage* img=cvQueryFrame(capture);
		if(!img) {
		fprintf(stderr,"Error: no hay imagen de la cámara.\n");
		getchar(); return -1;}
		while(1) {
		img = cvQueryFrame(capture);
		if(!img) {
		fprintf(stderr, "ERROR: no hay imagen de la cámara.\n"); getchar(); break; }
		cvShowImage("mywindow", img);
		if((cvWaitKey(10) & 255) == 27) break;
		}
		cvReleaseCapture(&capture);
		cvDestroyWindow("mywindow");
		return 0;}

cv::VideoCapture() es un metodo no muy eficiente al momento de extraer los fotogramas del video, al momento de extraer haciendo reconocimiento de la mano se vuelve mas lento y el trabajo FPS va disminuyendo, pero es el unico que ofrece trabajar al mismo tiempo con Realidad Aumentada, vision computacional.

CvCapture* capture = cvCaptureFromCAM(), es una buena opcion para trabajar con reconocimiento gestual, pero al momento de tratar de manejar los frames, iterar con otras de imágenes o secuencias de video, deja de funcionar.

Al momento de proyectar la cámara, se sobrepone el primer fotograma de los videos, descomponiendo secuencialmente el video, con san_cap.isOpened() se extrae los frames que se proyecta para integrar de diferentes maneras a la proyección de la pantalla, y con

el reconocimiento de la palma se puede reproducir o poner stop a la reproducción de los videos, como se puede ver en la figura 4.3.

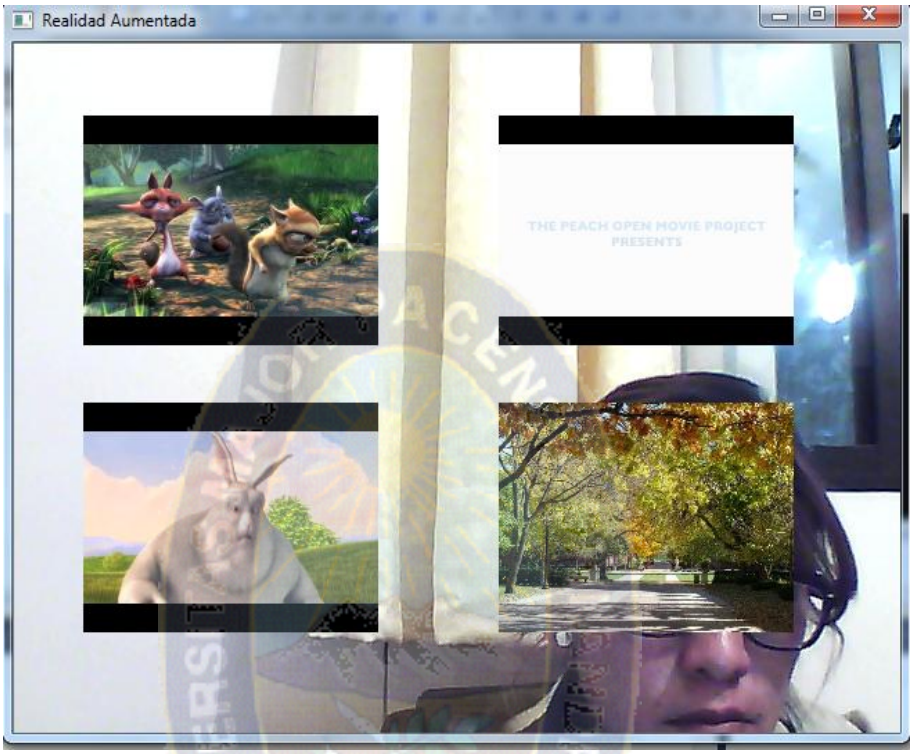

Figura 4.3: Pantalla Iteractiva.
Fuente: Elaboración Propia

4.6. Módulo de Inserción de Sonido

Después de trabajar con la libreria Opencv para incursionar en la realidad aumentada con reconocimiento gestual , se implementa la inmersión de una pantalla virtual de interacción con el usuario creando una interfaz que genere audio en función de la información que se obtenga a través de los fotogramas que se genera en la cámara, asemejando a la realidad aumentada. Lo primero es por incluir una librería de audio adecuada. Nuestra elección ha sido Audiere, por ser una librería libre, multiplataforma, gratuita y versatil. La característica que quizás da más personalidad a Audiere, es su orientación a la generación de audio 3D. Se busca generar un sonido o tema elegido por el usuario en función de la media de movimiento que se percibe. Trabajando con una

vector de sonidos que permite la reproducción del sonido que se eligio respectivamente con la mano, como se observa en la figura 4.4, es un menú de opción de que sonido se quiere escuchar.

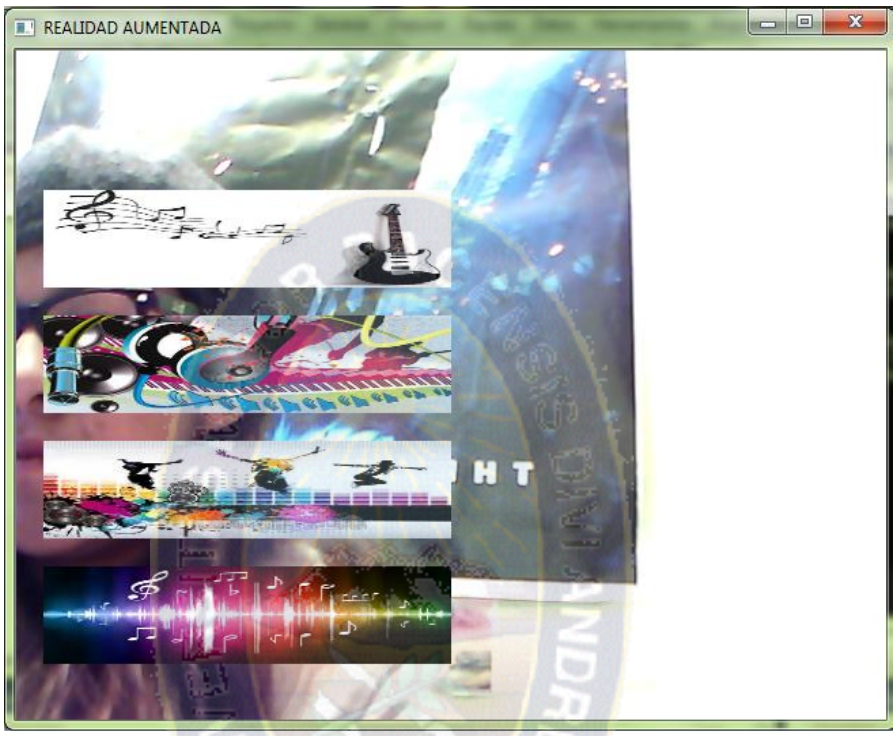

Figura 4.4: Menu de Opciones de sonidos
Fuente: Elaboración Propia

4.7. Resultados Presentación y Evaluacion de Pruebas

En este apartado se comentaran las pruebas realizadas con las distintas librerías, el objetivo de estas pruebas es verificar que la libreria funciona de forma correcta y optima, identificando correctamente los movimientos detectados realizados por el el usuario. Al tratarse de librerias que es imposible ejecutarse por si solas, por que es necesario implementar una aplicacion que inicie la deteccion de reconocimiento de la mano, y deteccion de movimientoe introduccion a la realidad aumentada.

Se analizan los resultados obtenidos en la evaluacion de las librerias tanto en la face de clasificacion de imagenes para generar el clasificador, la deteccion de movimiento con Aforge.NET y el reconocimiento gestual con OpenCV.

El proceso para realizar las pruebas consistira en los siguientes casos:

Se hizo pruebas con respecto a la capturacion de los fotogramas que capturan atraves de la camara cada libreria. Comenzemos el experimento con AForge.NET.

Con Aforge.Net se hizo pruebas de un tiempo de 15 minutos a razon de 30 fps(frames por segundo), se realizo movimientos y no hubo un reconocimiento de movimiento certero, no captando los movimientos que estaba realizando el usuario u objeto, esto se debe a la velocidad de fps.

Haciendo pruuebas con OpenCV, en un tiempo de 15 minutos, con una ventana que se muestra a 9 frames por segundo reconoce la mano de manera factible, para el proceso de reconocimiento es importante el entrenamiento que se tuvo con el clasificador, es tamien importante mencionar que si el clasificador tiene mas iteraccionde de entrenamiento , este sera mas optimo al momento de reconocer la mano.

Momvimiento Realizado	Aforge.NET	OpenCV
	Reconocimiento de movimiento hacia la Izquierda	Reconocimiento de movimiento de la mano en cualquier punto hacia la izquierda, derecha, arriba, abajo
	Reconocimiento de movimiento hacia la Derecha	

Tabla 4.1: Comparacion de Movimiento AForge.NET vs. OpenCV
Fuente: Elaboracion Propia

4.7.1 Reconocimiento con ruido

Para el analisis de la libreria AForge.NET, en un tiempo de 15 minutos se obtuvo el mismo resultado que si estuviera sin ruido, no es tan importante el ruido , si el entorno tiene ruido pues este no se mueve no reconoce ningun objeto en movimiento, esto se debe a que trabaja pixel a pixel.

Con OpenCV se hizo en un tiempo de 15 minutos, con un procesamiento de fotogramas por segundo igual a 9, con ruido se pudo verificar claramente que para el reconocimiento gestual se precisa estar sin ruido por que este puede confundir al clasificador. En el caso de OpenCV es muy importante el tema de ruido por que el clasificador muchas veces confunde imágenes o ruidos que estén detrás de la pantalla en el que se puede reconocer a traves de vacios ruidoso crear imagenes aleatorias parecidas a la palma de la mano.

4.7.2. Comparacion de entrada de imagenes

Como se menciono en los anteriores experimentos en un tiempo estimado de 15 minutos, en arforge se recibe de entrada, la mano o una parte del cuerpo, no hace precicion del reconocimiento de la mano, esta libreria trabaja mas con el algoritmo de reconocimiento de movimiento, entonces cuando el objeto , la mano, un cuerpo genera movimiento, este responde si fue un movimieinto hacia el lado derecho, hacia el lado izquierdo o hacia arriba.

El mismo tiempo de comparacion se tomo para OpenCV, recibiendo de entrada todo el cuerpo humano, pero en este caso la aplicacion solo hace reconocimeinto de la mano, y la enmarca en un rectangulo, lo favorable de opencv es que es al mismo tiempo de generar la camara web se puede incursiona sobre la pantalla el evento que no sotros quisiseramos que nos muestre, dando una incursion a la realidad aumentada, siendo este una pantalla virtual de interaccion de eventos.

Precisión de tiempo de respuesta	Aforge.NET	OpenCV
	Analizando resultados se identifica las muestras en la que se ha detectado movimiento, aumentado considerablemente el tiempo de ellas, dependiendo de la calidad de la imagen con un tiempo de respuesta muy cerca a 30 segundos	Los resultados que devuelve Opencv es mas Optimo, respecto al reconocimiento de la mano, el tiempo de es de 0.2 segundos, considerablemente con mayor velocidad.

Tabla 4.2: Comparacion precision tiempo AForge.NET vs. OpenCV
Fuente:Elaboracion Propia

La clase procesa imagen especificada del cuerpo humano con determinados eventos, para eso primero se construyo el haarclasifer para la detección de la mano, reconociendo la mano, la librería OpenCV está totalmente integrada con el framework aunque OpenCV es una librería desarrollada mayoritariamente en C, para que sea mucho más eficiente, para el procesamiento de imágenes y es una de las que se está usando para la realidad aumentada ya que es de código abierto, opencv tiene incluida ya algunas funciones haar para el reconocimiento de la mano por ejemplo, donde ya tiene varios algoritmos introducidos y desarrollados para el reconocimiento de distintas formas, pero si se quiere hacer un especifico reconocedor, entonces se puede hacer un clasificador como lo hicimos en este caso, no tiene restricciones siendo la mejor para el desarrollo de reconocimiento gestual, realidad virtual y realidad aumentada.

5. Conclusiones y Líneas de trabajo Futuras

5.1 Conclusiones

Con la realización de esta tesis se a discernido las ventajas que puede ofrecer las distintas librerias o herramientas para el reconocimento gestual, entre ellas se puede destacar Opencv por adquirir una visión integradora del conocimiento. El ámbito de la visión por computador ha versado en torno a la detección de la mano, un problema a un complejo. En esta tesis hemos implementado un modo de hacerlo, con unos resultados que podemos considerar positivos aunque mejorables. Aplicando ya un algoritmo elaborado de clasificadores pero aún así la detección de las manos nunca sería completamente fiable, debido a la gran complejidad del problema, en el que sólo se pueden producir progresos si se avanza paso a paso.

Precisamente, con este proyecto se han asentado los cimientos para, en un futuro cercano, incluir el módulo de incursión a la realidad aumentada junto a la visión artificial, también nos hemos adentrado en el campo de la calidad, valorando esta incursión de manera muy positiva.

En definitiva, este estudio ha sido ante todo integrador, pero no es más que una pequeña porción de un trabajo mucho más amplio, puesto que la detección de gestos es un campo todavía por mejorar; supone un gran reto pero también presenta grandes oportunidades desarrollando una interfaz, que hace que sea inmersa a la visión computacional donde únicamente la palma de la mano del usuario, esa mano capturada es el punto de unión entre la realidad y lo virtual, como la percibimos con los objetos virtuales creados para aumentar tal realidad.

El sistema de interfaz permite la entrada de objetos, al tomarse de un cámara y operado por el sistema de reconocimiento de gestos en tiempo real, los resultados de los cuales se

utilizan para manipular una pantalla virtual, puesta en la ventana del monitor respectivamente.

Se tuvo cuidado para que las asignaciones de los gestos a las acciones eran tan intuitiva posible, es obvio que el uso de este sistema se ve facilitada por la facilidad con la que los usuarios pueden interactuar naturalmente a través de gestos. Sin embargo, el poder del sistema aumentará enormemente como sus capacidades de reconocimiento son una mayor y más complejo junto a los entornos interactivos se que introducen y más formación de los usuarios de se proporcionan gestos.

5.2. Cumplimiento de Objetivos

El objetivo general "Realizar el estudio de las librerías y herramientas que existen para el desarrollo de reconocimiento gestual que detecte la mano para la manipulación objetos sobre una pantalla virtual proyectada desde el monitor de la pc, que permita interactuar con el usuario en cualquier posición en tiempo real" planteada en el capitulo uno se cumplio con el estudio exautivo, a las diferentes herramientas que se tiene para el reconocimiento gestual, creando al mismo una interfaz que hace la manipulacion de objetos con solo la deteccion de la palma descritas en el capitulo tres.

En cuanto a los objetivos especificos planteado en la tesis de investigacion, a continuacion se describe el grado de cada uno del cumplimiento de cada uno de ellos:

a) Se hiso pruebas para verificarer la funcionalidad que ofrece la librería OpenCV y AForge.NET ,para la detección de gestos, eligiendo la mas eficiente en terminos de usabilidad, eficiencia y portabilidad. Haciendo pruebas con los dos se eligio a OpenCV.

b) Al momento de activar la camara web se identifica claramente marcado por un recuadro los gestos llevados a cabo por el usuario.

c) Se desarrollo una aplicacion que hace reconocmiento gestual, visión computacional que conllevan a la introduccion de desarrollo de realidad aumentada.

d) Se implemento de la librería de detección de gestos OpenCV haciendo el uso de una cámara permitiendo la capturar video, este dispositivo capturar diferentes frames durante el movimiento del usuario, tras aplicar un algoritmo se identifico el gesto que el usuario ha llevado a cabo.

e) Se desarrollo e implemento una interfaz multimedia virtual, inmersiva, basado en reconocimiento gestual.

5.3.Estado de la Hipotesis

La hipotesis del trabajo de investigacion postula lo siguiente:

"Mediante las técnicas de la visión computacional será posible implementar una interfaz inmersivo, multimedia y virtual basado en el reconocimiento de gestos sin la necesidad de tener entradas perifericas formando un entorno de Realidad Aumentada."

Para demostrar la hipotesis, se recurre a los resultados obtenidos mediante la implementacion de la aplicacion, tras el estudio de las librerías y herramientas que puedan existir para la visión computacional, e incursionando a la realidad aumentada se pudo hacer un clasificador realizando estudio exhaustivo sobre la clasificación de manos y la certeza de poder generar un entrenamiento guiado que nos ha llevado a un clasificador de manos basado en los características de Haar. La mejor librería de código abierto se puede describir es opencv, ya que es la más robusta en el procesamiento de imágenes y entrenamiento de gestos, realizado por un entrenamiento de varias iteraciones.

Es un claro ejemplo que no está difícil incursionar en la realidad aumentada a partir de una librería de código abierto como lo es opencv, esta nos ofrece muchas ventajas en el proceso de imágenes es un desafío intentando simular el evento del mouse con el

reconocimiento de gesto, llegando a un punto donde se juega con la profundidad de la pantalla solo a través de movimientos de la mano, se tiene que tener en cuenta que toda la información.

5.4. Línea de Trabajos Futuros

Se utilizaron muchos enfoques heurísticos para esta línea de desarrollo, se pueden buscar mejoras en los resultados del sistema desarrollado, utilizando algoritmos que logren una mayor performance de reconocimiento gestual, que sea una solución autónoma.

U optar por plataformas que ofrecen el reconocimiento gestual, pero estas ya es con licencia de paga.

5.4.1. Mejora en el clasificador

Como hemos visto, el clasificador de manos tiene porcentaje de acierto bastante alto, pero cambio arrojaba demasiados falsos positivos. Una posible mejora seria realizar una iteración. Con un nuevo procesado de las imágenes donde introduciremos los falsos positivos como muestras negativas para que el clasificador las descarte. Aunque el índice de detección no mejore recibiríamos unos resultados más limpios, y no necesitaríamos hacer preprocesados en el resultado para asegurarnos que los que estamos recibiendo es correcto.

REFERENCIAS BIBLIOGRÁFICAS

(Poggio T, Brunelli R, 1993), Poggio T, Brunelli R, (1993)vol. 15 , Face recognition: features versus templates. IEEE Trans. Recuperado de: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=254061&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel1%2F34%2F6467%2F00254061

(Herrero T. & Villena J., 2010), HerreroTamara & Villena Julio. (Marzo, 2010) Sistema Automatico de Deteccion y Etiquetado de caras en Imágenes, Recuperado de: http://e-archivo.uc3m.es/bitstream/handle/10016/11170/PFC%20Tamara%20Herrero%20Vez.pdf;jsessionid=C0BD5FBAB0B63835CFAE25CD09189A0A?sequence=1

(KOKH, 2002) Koch, N., Kraus, A., Hennicker, R. (2002). "The Authoring Process of the UML-based Web Engineering Approach".

(KOKR, 2003) Koch, N., Kraus, A. (2003). "The Expressive Power of UML-based Web Engineering".

(LARM, 1999) Larman 1999, UML y Patrones. Introducción al análisis de diseño orientada a objetos. Página 506. 1ra Edición, Prentice_Hall Hispanoamérica S. A. México.

(Lopez H., 2010), Hector Lopez Pombo (2010). Análisis y Desarrollo de Sistemas de Realidad Aumentada, Recuperado de: http://eprints.ucm.es/11425/1/memoria_final_03_09_10.pdf

(OpenCV 2013), (mayo 2013) The OpenCV Tutorials Release 4.4 Recuperado de: http:// **opencv**org / downloads.html

(López J., 2010), Lopez Bolos Jose (Noviembre 2010) Desarrollo de un sistema de Realidad Aumentada que Incluya Reconocimiento de Gestos. Recuperado de:

http://riunet.upv.es/bitstream/handle/10251/10353/Desarrollo%20de%20un%20sistema%20RA.pdf

(Agustí M. , 2012), Agustí Melchor, Manuel (2010), Introducción al empleo de técnicas de Visión por Computador mediante OpenCV. Disponible en: http://riunet.upv.es/bitstream/handle/10251/12690/introOpenCV.pdf?sequence=1

(Gomez G. & Sucar L. E.) Giovani Gomez & L. Enrique Sucar, Vision Computacional. Recuperado de: ccc.inaoep.mx/~esucar/Libros/**vision**-sucar-gomez.pdf

(Shlomer T. & Poppinga B. & Henze &Boll S. , 2013),Thomas Schlomer, Benjamin Poppinga & Niels Henze & Susanne Boll, (2013) Gesture Recognition with a Wii Controller. Recuperado de:http://www.benjaminpoppinga.de/downloads/gesture_recognition_with_a_wii_controller-schloemer_poppinga_henze_boll.pdf

(Bradsky G. & Kaebler A., 2008) Gary Bradski & Adrian Kaehler (Octubre 2008), Open Computer Vision Library, O'Reilly Press. Recuperado de: http://sourceforge.net/projects/opencvlibrary/

(OpenCV, 2013) OpenCV, (Septiembre 2013) OpenCV. Recuperado de: http://opencv.org/

(Viola P, & Michael J., 2001), Paul Viola & Michael J. Jones (2001)Rapid Object Detection using a Boosted Cascade of Simple features. Conference On Computer Vision And Pattern Recognition. Recuperado de: http://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

(Tamayo Z., 2012), Tamayo Zuluagua J. G. (2012), Reconocimiento De Figuras Geométricas A Través De Una Webcam Con Opencv. Recuperado de: http://bibliotecadigital.usbcali.edu.co/jspui/bitstream/10819/1041/1/Reconocimiento_Figuras_Geometricas_Tamayo_2012.pdf

(HighGUI OpenCV, 2013), HighGUI Reference Manual Recuperado de: http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_highgui.htm

(Aranda A. & Lopez J.& Yin L. ,2011-2012), Arranz Aranda, Francisco & Liu Yin, Qi &López Cámara, Jose M. Interacción persona-computador basada en el reconocimiento visual de manos.

(Laganière R, 2011), Robert Laganière, (mayo,2011) OpenCV 2 Computer Vision Application Programming Cookbook, Recuperado de: zenithlib.googlecode.com

(Agustí Melchor Manuel, 2011), Primitivas gráficas: dibujando con OpenCV, Recuperado de: http://riunet.upv.es/bitstream/handle/10251/12692/primitivasGraficas.pdf?sequence=1&isAllowed=y

(Herbas L. & Villaarreal J.L. , 2007), Herbas L. & Villaarreal J.L.(junio, 2007) LA tecnología Aumentada: Una tecnología en espera de usuarios, Recuperado de: http://www.revista.unam.mx/vol.8/num6/art48/jun_art48.pdf

(UNLP-TEA, 2013), Realidad Aumentada, recuperado de:http://unlp-tea.wikispaces.com/Realidad+aumentada

(Andersen M., Jensen T., Lisouski P., Mortensen A., Hansen M., Gregersen T., Ahrendt P., 2013) Kinect depth sensor evaluation for computer vision applications(febrero, 2013), Recuperado de: http://eng.au.dk/fileadmin/DJF/ENG/PDF-filer/Tekniske_rapporter/Technical_Report_ECE-TR-6-samlet.pdf

(superposición en OpenCV, 2013) blog jepson (2012), recuperado de: http://jepsonsblog.blogspot.com/2012/10/overlay-transparent-image-in-opencv.html

(Airgesture, 2013) Funciones de Airgesture en el Samsumg Galaxy S4 Recuperdo de: http://www.tuexperto.com/2013/03/20/asi-funciona-air-view-y-air-gesture-en-el-samsung-galaxy-s4/

(Pastor J., 2013) Javier Pastor (2013) Leap Motion disponible Recuperado de: http://www.xataka.com/tag/leap-motion

(Openarch, 2013) Openarch Prototipo de vida inteligente, Recuperado de: http://ecosistemaurbano.org/castellano/openarch-prototipo-real-de-vivienda-inteligente/

(OpenCV, 2013) Introducción a OpenCV, Recuperado de: http://web-sisop.disca.upv.es/~imd/cursosAnteriors/2k3-2k4/copiaTreballs/serdelal/trabajoIMD.xml

(Leap Mootion, 2013) Leap Motion https://www.leapmotion.com/

(Leap Motion, 2013) Desarrollo de videojuegos para leap motion Recuperado de http://www.xklibur.com/archives/2013/07/22/mi-experiencia-en-el-desarroll-de-videojuegos-para-leap-motion/

https://code.google.com/p/aforge/issues/list

(AForge.NET, 2013) AForge.NET Recuperado de http://www.aforgenet.com/aforge/framework/

(AdaBoost, 2013) Bibliografia de AdaBoost Recuperado de http://en.wikipedia.org/wiki/AdaBoost

(Emgu CV, 2012) Tutorial de EmguCV. Recuperado de: http://www.didehbonyan.com/rz/Portals/0/pdf/Emgu%20CV%20Tutorial%20Skander.pdf

(Andersen M.& Jensen T. & Lisousji P. & MOrtensen K.&Hansen K.& Gregersen T., 2012)M.R. Andersen & T. Jensen& P. Lisouski& A.K. Mortensen& M.K. Hansen& T. Gregersen & P. Ahrendt (Febrero, 2012)Department of Engineering – Electrical and Computer Engineering, Aarhus University, Kinect Depth Sensor Evaluation For Computer Vision Applications Recuperado de: http://eng.au.dk/fileadmin/DJF/ENG/PDF-filer/Tekniske_rapporter/Technical_Report_ECE-TR-6-samlet.pdf

ANEXO A

TIPOS DE DATOS UTILIZADOS

La librería OpenCV dispone de múltiples tipos de datos específicos para desenvolverse con facilidad en cada área de la visión por computador. En este Apartado únicamente abordaremos los utilizados por nuestro programa.

CvPoint
Define las coordenadas de un punto. Posee dos campos, el valor entero de la coordenada X y el de Y.

CvScalar
La estructura CvScalar es un contenedor de hasta 4 valores de tipo double. Útil para almacenar el valor de un píxel en los diferentes canales.

IplImage
Formato de imagen de OpenCV. Las imágenes están compuestas por una cabecera y una zona de datos. La estructura está formada por campos relativos a la imagen, de los cuales sólo los más importantes describimos a continuación:

- Width y height: Ancho y alto de la imagen.
- Depth: Tipo de valor que puede tomar cada píxel. Las opciones disponibles son:

IPL_DEPTH_8U Unsigned 8-bit integer (8u)
IPL_DEPTH_8S Signed 8-bit integer (8s)
IPL_DEPTH_16S Signed 16-bit integer (16s)
IPL_DEPTH_32S Signed 32-bit integer (32s)
IPL_DEPTH_32F 32-bit floating-point single-precision (32f)
IPL_DEPTH_64F 64-bit floating-point double-precision (64f)

- NChanels: Número de planos que tiene la imagen, 1 para imágenes monocromo y 3 para imágenes a color.
- Origin: Origen de los ejes de coordenadas de la imagen.
- ImageData: Puntero a la primera final de la imagen.

CvCapture
La estructura de captura de vídeo. No posee interface pública y es usada solamente como parámetro para las operaciones de captura de vídeo.

CvRect
Define la posición y tamaño de un rectángulo. Posee cuatro campos, dos para las Posiciones X e Y de la esquina superior izquierda del rectángulo, y otros dos con el Ancho y largo de éste. La definición del constructor es la siguiente:
CvRect cvRect (int x, int y, int width, int height);

CvSize
Estructura para definir el tamaño de una imagen. Posee dos campos:

- Width: Ancho de la imagen.
- Height: Alto de la imagen.

CvSeq
Una secuencia es un array de tamaño variable de diferentes tipos de elementos almacenado en un almacén de memoria. La secuencia es dividida en partes y cada una se almacena en un bloque de memoria. Cada uno, apunta al siguiente y al anterior.

CvMat
Estructura que define a las matrices. Puede contener números reales de precisión simple o doble. Hay funciones que permiten hacer operaciones aritméticas sencillas y otras más complejas que pueden calcular los auto valores o la descomposición en valores singulares.

FUNCIONES UTILIZADAS

En este apartado se definirán algunas de las funciones utilizadas y se explicarán brevemente los parámetros de cada una:

cvCreateImage
Inicializa una imagen, crea la cabecera y reserva memoria. Devuelve un puntero a la cabecera de la imagen creada.
IplImage* cvCreateImage(CvSize size, int depth, int channel);
• Size: tamaño de la imagen.
• Depth: tipo de datos para los píxeles
• Channels: número de canales

cvReleaseImageHeader
Libera únicamente la cabecera de una imagen.
void cvReleaseImageHeader(IplImage** image);
• Image: doble puntero a la cabecera de la imagen.

cvReleaseImage
Libera la cabecera y memoria de una imagen.
void cvReleaseImage(IplImage** image);
• Image: doble puntero a la cabecera de la imagen.

cvSet2D
Cambia el valor de un elemento de un array.
CvScalar cvSet2D (const CvArr* arr, int idx0, int idx1);
• Arr: array de origen.
• Idx0, idx1: índices del elemento.

cvResetImageROI
Elimina el área de interés creado para incluir la imagen completa y libera la estructura ROI.
void cvResetImageROI (IplImage* image);

cvShowImage
Dibuja una imagen en la ventana especificada.
void cvShowImage(const char* name, const CvArr* image);
• Name: nombre de la ventana.
• Image: puntero a la imagen que mostrar.

cvExtractSURF
Extrae las características SURF de una imagen.
void cvExtractSURF(const CvArr* image, const CvArr* mask,
CvSeq** keypoints, CvSeq** descriptors, **CvMemStorage*** storage,
CvSURFParams params);
• Image: la imagen de entrada. Debe ser de 8 bits monocroma.
• Mask: máscara de 8 bits opcional.
• Keypoints: parámetro de salida, doble puntero a la secuencia de puntos clave.
• Descriptors: parámetro de salida opcional, doble puntero a la secuencia de descriptores.
• Storage: almacén de memoria donde los puntos clave y descriptores serán guardados.
• Params: varios parámetros del algoritmo definidos por la estructura CvSURFParams.

cvDestroyWindow
Destruye una ventana.
void cvDestroyWindow(const char* name);
• Name: nombre de la ventana.

cvWaitKey
Espera la pulsación de una tecla.
int cvWaitKey (int delay=0);
• Delay: retraso en milisegundos.

cvGet2D
Devuelve el valor específico de un elemento de un array.
CvScalar cvGet2D (const CvArr* arr, int idx0, int idx1);
• Arr: array de origen.
• Idx0, idx1: índices del elemento.

cvSetImageROI

Establece la región de interés de una imagen, delimitada por el rectángulo dado.

void cvSetImageROI (IplImage* image, CvRect rect);

• Image: puntero a la imagen.

• Rect: rectángulo que determina la región de interés.

cvCreateMat

Crea la cabecera y almacena los datos de una matriz.

CvMat* cvCreateMat (int rows, int cols, int type);

• Rows: Número de filas de la matriz.

• Cols: Número de columnas de la matriz.

• Type: Tipo de los elementos de la matriz. Definido como CV_<bit depth><S|U|F><number of channels>.

cvWarpPerspective

Apca la transformación en perspectiva a una imagen.

void cvWarpPerspective(const CvArr* src, CvArr* dst, const

CvMat* mapMatrix, int flags, CvScalar fillval);

• Src: imagen de origen.

• Dst: imagen de destino.

• MapMatrix: matriz de transformación 3x3.

• Flags: métodos de interpolación.

• Fillval: valor que se usa para rellenar los valores exteriores.

cvQueryFrame

Devuelve un puntero a una captura de un frame de la cámara.

IplImage* cvQueryFrame (CvCapture* capture);

• Capture: estructura de captura de vídeo.

ANEXO B

Codigo desarrollado con AForge.NET, para el Reconocimiento de Movimiento

```
namespace AForge.GestureRecognition
{
   using System;
   using System.Drawing;
   using System.Drawing.Imaging;
   using System.Runtime.InteropServices;

   using AForge.Imaging;
   using AForge.Imaging.Filters;

  public delegate void GestureDetectionEventHandler( object sender, Gesture gesture );

   public class GesturesRecognizerFromVideo
   {
      private int width;
      private int height;
      private int frameSize;
     / / Fondo y fotogramas anteriores
      private Bitmap backgroundFrame = null;
      private Bitmap previousFrame = null;
      private GrayscaleBT709 grayscaleFilter = new GrayscaleBT709( );
      private Difference differenceFilter = new Difference( );
      private Threshold thresholdFilter = new Threshold( 25 );
      private Erosion erosionFilter = new Erosion( );
      private Opening openingFilter = new Opening( );
      private MoveTowards moveTowardsFilter = new MoveTowards( );
      private BlobsFiltering blobsFilter = new BlobsFiltering( 50, 50, 1000, 1000 );
      private BlobCounter blobCounter = new BlobCounter( );
      private GestureRecognizer gestureRecognizer = new GestureRecognizer( );  / / Reconocedor de
movimiento de imagen fija
      private bool notDetected = true;  // Indicador de detección de gesto
      private Gesture previousGesture = new Gesture( );  / / Gesto detectado en un fotograma de vídeo
anterior
      private int framesWithoutMotion = 0;
      private int framesWithoutGestureChange = 0;
      private int minBodyWidth = 50;
      private int minBodyHeight = 50;
      private double motionLimit = 0.005;
      private int minFramesWithoutMotion = 3;
      private int minFramesWithoutGestureChange = 1;
      public GestureRecognizer GestureRecognizer
      {
         get { return gestureRecognizer; }
      }
      public int MinBodyWidth
```

```
        {
            get { return minBodyWidth; }
            set
            {
                minBodyWidth = value;
                blobsFilter.MinWidth = minBodyWidth;
            }
        }
        public int MinBodyHeight
        {
            get { return minBodyHeight; }
            set
            {
                minBodyHeight = value;
                blobsFilter.MinHeight = minBodyHeight;
            }
        }
        public double MotionLimit
        {
            get { return motionLimit; }
            set { motionLimit = Math.Max( 0, Math.Min( 1, value ) ); }
        }
        public int MinFramesWithoutMotion
        {
            get { return minFramesWithoutMotion; }
            set { minFramesWithoutMotion = value; }
        }
        public int MinFramesWithoutGestureChange
        {
            get { return minFramesWithoutGestureChange; }
            set { minFramesWithoutGestureChange = value; }
        }
        public event GestureDetectionEventHandler GestureDetected;
        public GesturesRecognizerFromVideo( )
        {
            blobsFilter.MinWidth  = minBodyWidth;
            blobsFilter.MinHeight = minBodyHeight;
        }
        public void Reset( )
        {
            if ( backgroundFrame != null )
            {
                backgroundFrame.Dispose( );
                backgroundFrame = null;
            }
            / / Eliminar fotograma anterior
            if ( previousFrame != null )
            {
                previousFrame.Dispose( );
                previousFrame = null;
            }
/ / Restablecer contadores
```

```
            framesWithoutMotion = 0;
            framesWithoutGestureChange = 0;
            notDetected = true;
        }
          / / Proceso nuevo marco de la fuente de vídeo y tratar de reconocer el gesto.
        public void ProcessFrame( ref Bitmap image )
        {
            if ( backgroundFrame == null )  / / Comprobar marco de fondo.
            { / / Guardar la dimensión de la imagen
                width     = image.Width;
                height    = image.Height;
                frameSize = width * height;
                backgroundFrame = grayscaleFilter.Apply( image );  / / Se aplica el filtro de escala de grises
                return;
            }
            if ( ( image.Width != width ) || ( image.Height != height ) )
                return;
            Bitmap currentFrame = grayscaleFilter.Apply( image );
            differenceFilter.OverlayImage = backgroundFrame;
            Bitmap motionObjectsImage = differenceFilter.Apply( currentFrame );
            BitmapData motionObjectsData = motionObjectsImage.LockBits(
                new Rectangle( 0, 0, width, height ),
                ImageLockMode.ReadWrite, PixelFormat.Format8bppIndexed );
            thresholdFilter.ApplyInPlace( motionObjectsData );
            openingFilter.ApplyInPlace( motionObjectsData );
            blobCounter.ProcessImage( motionObjectsData );
            Blob[] blobs = blobCounter.GetObjectInformation( );
            int maxSize = 0;
            Blob maxObject = new Blob( 0, new Rectangle( 0, 0, 0, 0 ) );
            if ( blobs != null )
            {
                foreach ( Blob blob in blobs )
                {
                    int blobSize = blob.Rectangle.Width * blob.Rectangle.Height;

                    if ( blobSize > maxSize )
                    {
                        maxSize = blobSize;
                        maxObject = blob;
                    }
                }
            }
/ / Si tenemos sólo los objetos pequeños, entonces vamos a adoptar a los cambios en la escena
            if ( ( maxObject.Rectangle.Width < 20 ) || ( maxObject.Rectangle.Height < 20 ) )
            {
              / / Mover fondo hacia fotograma actual
                moveTowardsFilter.OverlayImage = currentFrame;
                moveTowardsFilter.ApplyInPlace( backgroundFrame );
            }

            if ( ( maxObject.Rectangle.Width >= minBodyWidth ) && ( maxObject.Rectangle.Height >=
minBodyHeight ) && ( previousFrame != null ) )
```

```
        {
            differenceFilter.OverlayImage = previousFrame;

            Bitmap betweenFramesMotion = differenceFilter.Apply( currentFrame );
            BitmapData betweenFramesMotionData = betweenFramesMotion.LockBits(
                new Rectangle( 0, 0, width, height ),
                ImageLockMode.ReadWrite, PixelFormat.Format8bppIndexed );
            thresholdFilter.ApplyInPlace( betweenFramesMotionData );
            openingFilter.ApplyInPlace( betweenFramesMotionData ); / / Solicitar la apertura del filtro
para eliminar el ruido
          / / Calcular la cantidad de píxeles que cambian
            VerticalIntensityStatistics vis = new VerticalIntensityStatistics( betweenFramesMotionData );
            int[] histogram = vis.Gray.Values;
            int   changedPixels = 0;
            for ( int i = 0, n = histogram.Length; i < n; i++ )
            {
                changedPixels += histogram[i] / 255;
            }
            betweenFramesMotion.UnlockBits( betweenFramesMotionData );
            betweenFramesMotion.Dispose( );
            if ( (double) changedPixels / frameSize <= motionLimit )
            {
                framesWithoutMotion++;
            }
            else
            {
                framesWithoutMotion = 0;
                framesWithoutGestureChange = 0;
                notDetected = true;
            }

            if ( framesWithoutMotion >= minFramesWithoutMotion )
            {
                if ( notDetected )
                {
                    blobCounter.ExtractBlobsImage( motionObjectsData, maxObject );
                    Gesture gesture = gestureRecognizer.Recognize( maxObject.Image, true ); / / Reconocer
el gesto de la imagen
                    maxObject.Image.Dispose( );
                    / / Comprobar si los gestos ha cambiado desde el marco anterior
                    if (
                        ( gesture.LeftHand == previousGesture.LeftHand ) &&
                        ( gesture.RightHand == previousGesture.RightHand )
                        )
                    {
                        framesWithoutGestureChange++;
                    }
                    else
                    {
                        framesWithoutGestureChange = 0;
                    }
                    / / Comprobar si gesto no fue cambiando durante un tiempo
```

```
                    if ( framesWithoutGestureChange >= minFramesWithoutGestureChange )
                    {
                        if ( GestureDetected != null )
                        {
                            GestureDetected( this, gesture );
                        }
                        notDetected = false;
                    }

                    previousGesture = gesture;
                }
            }
        }
        else
        {
            framesWithoutMotion = 0;
            framesWithoutGestureChange = 0;
            notDetected = true;
        }
        motionObjectsImage.UnlockBits( motionObjectsData );
        motionObjectsImage.Dispose( );
        if ( previousFrame != null )
        {
            previousFrame.Dispose( );
        }
        previousFrame = currentFrame;
    }
  }
}
namespace AForge.GestureRecognition
{
  using System;
  using System.Drawing;
  using System.Drawing.Imaging;

  using AForge;
  using AForge.Imaging;
  using AForge.Imaging.Filters;
  using AForge.Math;

  public class GestureRecognizer
  {
      private double torsoCoefficient = 0.3;
      private double handsMinProportion = 0.3;
      private double minStraightHandProportion = 1.33;
      private double maxRaisedUpHandProportion = 0.35;
     public double TorsoCoefficient
      {
          get { return torsoCoefficient; }
          set { torsoCoefficient = Math.Max( 0, Math.Min( 1, value ) ); }
      }
      public double HandsMinProportion
```

```
{
   get { return handsMinProportion; }
   set { handsMinProportion = Math.Max( 0, value ); }
}
public double MinStraightHandProportion
{
   get { return minStraightHandProportion; }
   set { minStraightHandProportion = Math.Max( 0, value ); }
}
public double MaxRaisedUpHandProportion
{
   get { return maxRaisedUpHandProportion; }
   set { maxRaisedUpHandProportion = Math.Max( 0, value ); }
}

public GestureRecognizer( ) { }
public Gesture Recognize( Bitmap image, bool bodyImageOnly )
{
   if ( image.PixelFormat != PixelFormat.Format8bppIndexed )
   {
      throw new ArgumentException( "Source image can be binary (8 bpp indexed) only" );
   }

   BitmapData imageData = image.LockBits(
      new Rectangle( 0, 0, image.Width, image.Height ),
      ImageLockMode.ReadOnly, PixelFormat.Format8bppIndexed );
  Gesture gesture = Recognize( imageData, bodyImageOnly );
   image.UnlockBits( imageData );

   return gesture;
}

public Gesture Recognize( BitmapData imageData, bool bodyImageOnly )
{
   if ( imageData.PixelFormat != PixelFormat.Format8bppIndexed )
   {
      throw new ArgumentException( "Imagen puede ser binario (8 bpp)" );
   }

   Gesture gesture = new Gesture( HandPosition.NotRaised, HandPosition.NotRaised );
   Bitmap bodyImage = null;
   BitmapData bodyImageData = null;

   if ( bodyImageOnly == false )
   {
       Shrink shrinkFilter = new Shrink( );
      bodyImage = shrinkFilter.Apply( imageData );
       bodyImageData = bodyImage.LockBits(
         new Rectangle( 0, 0, bodyImage.Width, bodyImage.Height ),
         ImageLockMode.ReadOnly, PixelFormat.Format8bppIndexed );
   }
   else
```

```
        {
            bodyImageData = imageData;
        }
        int bodyWidth = bodyImageData.Width;
        int bodyHeight = bodyImageData.Height;
        HorizontalIntensityStatistics his = new HorizontalIntensityStatistics( bodyImageData );
        int[] hisValues = (int[]) his.Gray.Values.Clone( );
        double torsoLimit = torsoCoefficient * bodyHeight;
        for ( int i = 0; i < bodyWidth; i++ )
        {
            hisValues[i] = ( (double) hisValues[i] / 255 > torsoLimit ) ? 1 : 0;
        }
       int leftHand = 0;
        while ( ( hisValues[leftHand] == 0 ) && ( leftHand < bodyWidth ) )
            leftHand++;
        int rightHand = bodyWidth - 1;
        while ( ( hisValues[rightHand] == 0 ) && ( rightHand > 0 ) )
            rightHand--;
        rightHand = bodyWidth - ( rightHand + 1 );
        int torsoWidth = bodyWidth - leftHand - rightHand;
        if ( ( (double) leftHand / torsoWidth ) >= handsMinProportion )
        {
            Crop cropFilter = new Crop( new Rectangle( 0, 0, leftHand, bodyHeight ) );
            Bitmap leftHandImage = cropFilter.Apply( bodyImageData );
            gesture.LeftHand = GetHandPosition( leftHandImage );
        }
        if ( ( (double) rightHand / torsoWidth ) >= handsMinProportion )
        {
            Crop cropFilter = new Crop( new Rectangle( bodyWidth - rightHand, 0, rightHand,
bodyHeight ) );
            Bitmap rightHandImage = cropFilter.Apply( bodyImageData );
            gesture.RightHand = GetHandPosition( rightHandImage );
        }
        if ( !bodyImageOnly )
        {
            bodyImage.UnlockBits( bodyImageData );
            bodyImage.Dispose( );
        }
        return gesture;
    }
    private HandPosition GetHandPosition( Bitmap handImage )
    {
        HandPosition handPosition = HandPosition.NotRaised;
        VerticalIntensityStatistics stat = new VerticalIntensityStatistics( handImage );
        Histogram histogram = stat.Gray;
        FilterLowValues( histogram );
        FilterNoisyPeaks( histogram );
        if ( ( (double) handImage.Width / ( histogram.Max - histogram.Min + 1 ) ) >
minStraightHandProportion )
        {
            handPosition = HandPosition.RaisedStraigh;
        }
```

```
            else
            {
               if ( ( (double) histogram.Min / ( histogram.Max - histogram.Min + 1 ) ) <
maxRaisedUpHandProportion )
               {
                  handPosition = HandPosition.RaisedDiagonallyUp;
               }
               else
               {
                  handPosition = HandPosition.RaisedDiagonallyDown;
               }
            }
            return handPosition;
         }
         private void FilterLowValues( Histogram histogram )
         {
            int[] values = histogram.Values;
            int   globalMax = 0;
            for ( int k = 0; k < values.Length; k++ )
            {
               if ( values[k] > globalMax )
               {
                  globalMax = values[k];
               }
            }
            int filterLevel = (int) ( (double) globalMax / 10 );
            for ( int k = 0; k < values.Length; k++ )
            {
               if ( values[k] <= filterLevel )
               {
                  values[k] = 0;
               }
            }
            histogram.Update( );
         }
         private void FilterNoisyPeaks( Histogram histogram )
         {
            int[] values = histogram.Values;
            int   globalMax = 0;
            for ( int k = 0; k < values.Length; k++ )
            {
               if ( values[k] > globalMax )
               {
                  globalMax = values[k];
               }
            }
            int i = 0;
            while ( i < values.Length )
            {
               while ( ( i < values.Length ) && ( values[i] == 0 ) )
               {
                  i++;
```

```
            }
            int localMax = 0;
            while ( ( i < values.Length ) && ( values[i] != 0 ) )
            {
               if ( values[i] > localMax )
               {
                  localMax = values[i];
               }
               i++;
            }
            if ( localMax < globalMax )
            {
               int j = i - 1;

               while ( ( j >= 0 ) && ( values[j] != 0 ) )
               {
                  values[j] = 0;
                  j--;
               }
            }
         }
         histogram.Update( );
      }
   }
}
```

ANEXO C

Codigo desarrollado con OpenCVpara el Reconocimiento Gestual con Realidad Aumentad

```
#include "stdafx.h"
# include <opencv/cv.h>
#include <opencv/highgui.h>
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <assert.h>
# include <math.h>
# include <float.h>
# include <limits.h>
# include <time.h>
# include <ctype.h>
#include <iostream>
#include "audiere.h"
   using namespace audiere;
          const char * cascade_name = "openpalma.xml";        //reconocimiento de la palma
          int  vmax = 256;
          using namespace std;
          using namespace cv;
          bool actPaint = false;
          bool actVideo = false;
          bool actVideo1 = false;
          bool actVideo2 = false;
          bool actVideo3 = false;
          bool actVideo4 = false;
          bool actSonido1 = false;
          bool actSonido2 = false;
          bool actSonido3 = false;
          bool actSonido4 = false;
          bool actStop = false;
          bool ResetPantalla = false;
          bool pantalla=true;
          int RecorPalX[100000];
          int RecorPalY[100000];
          int recorridoPalma[2][100000];
          int numeroPuntos = -1;
          bool estaLleno = false;
          int red = 20;//20 definiendo colores del rectangulo
   int blue = 25;
          int green = 20;
          bool rojo = false;
          bool amarillo= false;
          bool azul= false;
```

```
        int sw = false;
        bool rojo1=false, amar=false, azu=false;
        // Adiciona el punto medio del cuadrado y lo almacena en un vector
        OutputStreamPtr audio[10] =
{NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL};
        int temps = 30;
        bool music=false;
        bool ActvMusic;
int teclesDown[10] = {0,0,0,0,0,0,0,0,0,0};/*Array que indica el temps que cada tecla ha estat
presionada*/
void CargarMusica()
{
   IplImage* tecla = cvLoadImage("M:/Haar/sonido/sonido.jpg
        string filename = "M:/Haar/sonido/";
   AudioDevicePtr device(OpenDevice());
        /*Es carreguen tots els audios*/
   for(int i=0; i<5 ; i++)
   {
      std::string s;
      std::stringstream out;
      out << i+1;
      string aux = filename +out.str() + ".mp3";
      OutputStreamPtr sound(OpenSound(device, aux.c_str(), false));
      audio[i] = sound;
   }
}       bool Puntodentrodeuncuadrado(int x1, int y1, int ancho, int alto, int xf,int yf)//
        {
                if( x1<xf && xf<(ancho+x1) && yf>y1 && yf<(alto+y1))
                        return true;
                return false;
        }
        void adicionarPuntoDibujo(int x , int y, int ancho, int alto)
        {
                printf("Pintando puntos : %d, x: %d, y: %d, ancho: %d, alto %d", numeroPuntos, x,
y, ancho, alto);
                numeroPuntos ++;
                if(numeroPuntos == 100000)
                {
                        numeroPuntos = 0;
                        estaLleno = true;
                }
                recorridoPalma[0][numeroPuntos] = x + (ancho/2);
                recorridoPalma[1][numeroPuntos] = y+ (alto/2);
                printf("Resutlatod punto: x: %d, y: %d", recorridoPalma[0][numeroPuntos],
recorridoPalma[1][numeroPuntos]);
}
        void pintarPuntoDibujo(IplImage* images, int x , int y, int ancho, int alto)
        {
                CvPoint pt1, pt2;
                rojo =cuadradoestadentrodelcuadrado(350, 0, 100, 100, x, y, ancho, alto);
                amarillo =cuadradoestadentrodelcuadrado(450, 0, 100, 100, x, y, ancho, alto);
                azul =cuadradoestadentrodelcuadrado(550, 0, 100, 100, x, y, ancho, alto);
```

```
                if(rojo )
        {               rojo1=rojo;
                          numeroPuntos=0;
                    rojo=false;
                }
                if(amarillo )
        {               bool amar=amarillo;
                          numeroPuntos=0;
                    amarillo=false;
                }
                if(azul )
        {               bool azu = azul;
                          numeroPuntos=0;
                    azul=false;
                }
                int tope = numeroPuntos;
                for(int i=0; i<tope; i++)
                {
                        pt1.x = recorridoPalma[0][i];
                        pt1.y = recorridoPalma[1][i];
                        pt2.x = pt1.x + 1;
                        pt2.y = pt1.y + 1;
                        //cvPoint()////aqui hay que cambiar
                        if(rojo1)
                        cvRectangle( images, pt1, pt2, CV_RGB(250,8,8), 15, 15, 0.5);//pintando
los colores
                        if(amar)
                        cvRectangle( images, pt1, pt2, CV_RGB(8,250,242), 15, 15,
0.5);//pintando los colores
                        if(azu)
                        cvRectangle( images, pt1, pt2, CV_RGB(250,16,8), 15, 15, 0.5);
                }
        }
        void cvOverlayImageOverImage(IplImage* src, IplImage* overlay, CvPoint location,
CvScalar S, CvScalar D)
        {
                int x,y,i;

                for(x=0;x < overlay->width;x++)
                {
                        if(x+location.x>=src->width) continue;
                        for(y=0;y < overlay->height;y++)
                        {
                                if(y+location.y>=src->height) continue;
                                CvScalar source = cvGet2D(src, y+location.y, x+location.x);
                                CvScalar over = cvGet2D(overlay, y, x);
                                CvScalar merged;
                                if(over.val[0] == 255 && over.val[1] == 255 && over.val[2] ==
255 && over.val[3] == 255)
                                {               for(i=0;i<4;i++)
                                                merged.val[i] = (source.val[i]);
                                }
```

```
                                        else
                                        {
                                                for(i=0;i<4;i++)
                                                        merged.val[i] = (over.val[i]);
                                        }
                                        cvSet2D(src, y+location.y, x+location.x, merged);
                                }                       }
        }
IplImage* skipNFrames(CvCapture* capture, int n)
        {
                for(int i = 0; i < n; ++i)      {
                        if(cvQueryFrame(capture) == NULL)           {
                                return NULL;
                        }
                }
                return cvQueryFrame(capture);
        }
        void overlayImage(const cv::Mat &background, const cv::Mat &foreground, cv::Mat
&output, cv::Point2i location)
        {
                                        background.copyTo(output);
                for(int y = std::max(location.y , 0); y < background.rows; ++y)
                {
                int fY = y - location.y;
                if(fY >= foreground.rows)
                        break;
                for(int x = std::max(location.x, 0); x < background.cols; ++x)
                {
                        int fX = x - location.x;
                        if(fX >= foreground.cols)
                        break;
                        double opacity = ((double)foreground.data[fY * foreground.step + fX *
foreground.channels() + 3]) / 255.;  // determine the opacity of the foregrond pixel, using its fourth
(alpha) channel.
                        opacity = 1.0;
                        for(int c = 0; opacity > 0 && c < output.channels(); ++c)
                        {
                        unsigned char foregroundPx = foreground.data[fY * foreground.step + fX *
foreground.channels() + c];
                        unsigned char backgroundPx = background.data[y * background.step + x *
background.channels() + c];
                        output.data[y*output.step + output.channels()*x + c] = backgroundPx * (1.-
opacity) + foregroundPx * opacity;
                        }
                }
                }
        }
        void overlayImage(const cv::Mat &background, const cv::Mat &foreground, cv::Mat
&output, cv::Point2i location, int ancho, int alto)
        {
                cv::Mat resizedImage;
           cv::resize(foreground,resizedImage, cv::Size(ancho, alto));
```

```
		overlayImage(background, resizedImage, output, location);
	}
	void PlayMusic(int xp, int yp,int anchoPalma, int altPalma)
	{
		bool sonido1 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionMusic[0], posicionMusic[1], posicionMusic[2], posicionMusic[3]) ||
cuadradoestadentrodelcuadrado(posicionMusic[0], posicionMusic[1], posicionMusic[2],
posicionMusic[3], xp, yp, anchoPalma,altPalma);
		bool sonido2 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionPaint[0], posicionPaint[1], posicionPaint[2], posicionPaint[3]) ||
cuadradoestadentrodelcuadrado(posicionPaint[0], posicionPaint[1], posicionPaint[2], posicionPaint[3],
xp, yp, anchoPalma,altPalma);
		bool sonido3 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionVideo[0], posicionVideo[1], posicionVideo[2], posicionVideo[3]) ||
cuadradoestadentrodelcuadrado(posicionVideo[0], posicionVideo[1], posicionVideo[2],
posicionVideo[3], xp, yp, anchoPalma,altPalma);
		bool sonido4 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionVideo[0], posicionVideo[1], posicionVideo[2], posicionVideo[3]) ||
cuadradoestadentrodelcuadrado(posicionVideo[0], posicionVideo[1], posicionVideo[2],
posicionVideo[3], xp, yp, anchoPalma,altPalma);
		bool stop = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma, 0, 0, 110,
110) || cuadradoestadentrodelcuadrado(0, 0, 110, 110, xp, yp, anchoPalma,altPalma);

		if (actSonido1)
		{	printf(" reproduciendo sonido 1");
			if(sonido1)
				actSonido1 = false;//si esta activado lo desactiva si hay colicion
entre la mano y el video
			else
				actSonido1 = true;//si esta desactivado lo ACTIVA si hay colicion
entre la mano y el video
			return;
		}
		if (sonido2)
		{	printf(" reporduciendo sonido 2");
			if(actSonido2)
				actSonido2 = false;
			else
				actSonido2 = true;
			return;
		}

		if (sonido3)
		{	printf(" Reproduccion de sonido 3");
			if(actSonido3)
				actSonido3 = false;
			else
				actSonido3 = true;
			return;
		}
		if (sonido4)
		{	printf(" reproduccion de sonido 4");
```

```
                        if(actSonido4)
                                actSonido4 = false;
                        else
                                actSonido4 = true;
                        return;
                }

                if (stop)
                {
                        if(actStop)
                                actStop = false;
                        else
                                actStop = true;
                        return;
                }
        }
        void ActivarPantalla(int xp, int yp,int anchoPalma, int altPalma)//, int w, int h
        {
                bool PantallaMusic = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionMusic[0], posicionMusic[1], posicionMusic[2], posicionMusic[3]) ||
cuadradoestadentrodelcuadrado(posicionMusic[0], posicionMusic[1], posicionMusic[2],
posicionMusic[3], xp, yp, anchoPalma,altPalma);
                bool PantallaPaint = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionPaint[0], posicionPaint[1], posicionPaint[2], posicionPaint[3]) ||
cuadradoestadentrodelcuadrado(posicionPaint[0], posicionPaint[1], posicionPaint[2], posicionPaint[3],
xp, yp, anchoPalma,altPalma);
                bool PantallaVideo = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionVideo[0], posicionVideo[1], posicionVideo[2], posicionVideo[3]) ||
cuadradoestadentrodelcuadrado(posicionVideo[0], posicionVideo[1], posicionVideo[2],
posicionVideo[3], xp, yp, anchoPalma,altPalma);
                if (PantallaMusic)
                {       printf(" trabajando en la pantalla de reproduccion de videos");
                        if(actMusic)
                                actMusic = false;//si esta activado lo desactiva si hay colicion
entre la mano y el video
                        else
                                actMusic = true;//si esta desactivado lo ACTIVA si hay colicion
entre la mano y el video
                        return;
                }
                if (PantallaPaint)
                {       printf(" Trabajando en la pantalla de dibujo de paint");
                        if(actPaint)
                                actPaint = false;
                        else
                                actPaint = true;
                        return;
                }
                if (PantallaVideo)
                {       printf(" trabajando en la pantalla de reproduccion de video");
                        if(actVideo)
                                actVideo = false;
```

```
                        else
                                actVideo = true;
                        return;
                }
        }
        void ActivarEvento(int xp, int yp,int anchoPalma, int altPalma)//, int w, int h
        {
                bool Video1 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionV1[0], posicionV1[1], posicionV1[2], posicionV1[3]) ||
cuadradoestadentrodelcuadrado(posicionV1[0], posicionV1[1], posicionV1[2], posicionV1[3], xp, yp,
anchoPalma,altPalma);
                bool Video2 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionV2[0], posicionV2[1], posicionV2[2], posicionV2[3]) ||
cuadradoestadentrodelcuadrado(posicionV2[0], posicionV2[1], posicionV2[2], posicionV2[3], xp, yp,
anchoPalma,altPalma);
                bool Video3 = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma,
posicionV3[0], posicionV3[1], posicionV3[2], posicionV3[3]) ||
cuadradoestadentrodelcuadrado(posicionV3[0], posicionV3[1], posicionV3[2], posicionV3[3], xp, yp,
anchoPalma,altPalma);
                bool stop = cuadradoestadentrodelcuadrado(xp, yp, anchoPalma,altPalma, 0, 0, 110,
110) || cuadradoestadentrodelcuadrado(0, 0, 110, 110, xp, yp, anchoPalma,altPalma);
                if (Video1)
                {       printf(" video 1 activado");
                        if(actVideo1)
                                actVideo1 = false;//si esta activado lo desactiva si hay colicion
entre la mano y el video
                        else
                                actVideo1 = true;//si esta desactivado lo ACTIVA si hay colicion
entre la mano y el video
                        return;
                }
                if (Video2)
                {       printf(" video 2 activado");
                        if(actVideo2)
                                actVideo2 = false;
                        else
                                actVideo2 = true;
                        return;
                }
                if (Video3)
                {       printf(" video 3 activado");
                        if(actVideo3)
                                actVideo3 = false;
                        else
                                actVideo3 = true;
                        return;
                }
                if (stop)
                {
                        if(actStop)
                                actStop = false;
                        else
```

```
                                        actStop = true;
                                return;
                        }
                }
                IplImage* detect_and_draw (const cv::Mat &resultado)
                {       IplImage* images;
                        static CvMemStorage* storage = 0;// Crear la memoria de cálculos
                        static CvHaarClassifierCascade* cascade = 0;// Crear un nuevo clasificador Haar
                        int scale = 1;//1  // Establece la escala con la que el rectángulo se dibuja con
                        CvPoint pt1, pt2; // Cree dos puntos para representar las posiciones de mano
                        int i; // Variable Looping
                        cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 ); // Cargar el
HaarClassifierCascade                                   //( cascade_name, 0, 0, 0 )
                                if( !cascade )  // Comprobar si la cascada se ha cargado correctamente.
Informe Else y error y salir
                        {
                                fprintf( stderr, "ERROR: Could not load classifier cascade\n" );
                                                }
                        storage = cvCreateMemStorage(0); // Asignar el almacenamiento de la memoria
                        cvCreateTrackbar( "Vmax", "result", &vmax, 256, 0 );
                        cvClearMemStorage( storage );  // Borrar el almacenamiento de la memoria que se
utiliza antes de
                        if( cascade )  // Comprueba si se ha cargado la cascada, para encontrar las manos. Si
es así, entonces:
                        { // No puede haber más de una mano en una imagen. Por lo tanto crear una
secuencia cultivable de las manos.// Detectar los objetos y almacenarlos en la secuencia
                                        images=cvCloneImage(&(IplImage)resultado);
                                        CvSeq* hands = cvHaarDetectObjects( images, cascade, storage,
1.1, 2,CV_HAAR_MAGIC_VAL,   cvSize(48, 48) );
                                        for( i = 0; i < (hands ? hands->total : 0); i++ )// Recorrer el número de
manos encontró.
                                        {
                                                CvRect* r = (CvRect*)cvGetSeqElem( hands, i );// Crear un
nuevo rectángulo para dibujar la mano
                                          fprintf( stderr ,"punto X!");         // Encuentra las dimensiones de la
mano, y la escala si es necesario
                                                pt1.x = r->x*scale;
                                                pt1.x = r->x; //r->
                                                //RecorPalX[i]=r->x; recorre los x de la pantalla de x
                                                printf("(%d,%d)\n", pt1.x);
                                                pt2.x = (r->x+r->width)*scale;
                                                fprintf( stderr ,"puntos YYY!! ");
                                                pt1.y = r->y*scale;
                                                pt1.y = r->y;
                                                //RecorPalY[i]=r->y; //poniendo los puntos en un vector
                                                printf("(%d,%d)\n", pt1.y);
                                                pt2.y = (r->y+r->height)*scale;
                                                int alto= r->height;
                                                cvRectangle( images, pt1, pt2, CV_RGB(20,20,232), 15, 15, 0.5);
        // Dibuja el rectángulo en la imagen de entrada
                                                ActivarEvento (r->x,r->y,r->width,r->height); //activa los
fotogramas de los videos
```

```
                                        if(pantalla)
                                        {
                                        ActivarPantalla (r->x,r->y,r->width,r->height); //activa los
fotogramas de los videos
                                        }

                                        PlayMusic(r->x, r->y, r->width, r->height);
                                        if(ResetPantalla){
                                        adicionarPuntoDibujo(r->x, r->y, r->width, r->height);
                                        pintarPuntoDibujo(images, r->x, r->y, r->width, r->height);
                                        }
                                }
                        }
                        return (images);
        }
        int main ()
        {       IplImage* palma = NULL;
                IplImage* frame = NULL;
                IplImage* frameImag = NULL;
                int waitKeyValue = 10;
                Mat frameV1,frameV2,frameV3,frameV4,frameImg;
                Mat overlay1, overlay2;
                Mat overlayIm;
                Mat src1, camera, dst, dstV,dstS,dstP;
                Mat resultado,resultado2;
                CvCapture* capture = cvCaptureFromAVI("01.avi");
                CvCapture* capture1 = cvCaptureFromAVI("02.avi");
                CvCapture* capture2 = cvCaptureFromAVI("03.avi");
                Mat video = imread("video.jpg");
                Mat musica = imread("musi.jpg");
                Mat Img1 = imread("1.jpg");
                Mat Img2 = imread("2.jpg");
                Mat Img3 = imread("3.jpg");
                Mat Img4 = imread("4.jpg");
                Mat off = imread("of.jpg");
                Mat Imagen = imread("img.jpg");
                frame1 = skipNFrames(capture,1);
           frameV1 = cvarrToMat(frame1);
                frame2 = skipNFrames(capture1,1);
                frameV2 = cvarrToMat(frame2);
                frame3 = skipNFrames(capture2,1);
                frameV3 = cvarrToMat(frame3);
                CargarMusica();
                cv::Mat san;
                cv::VideoCapture san_cap(0);
                if (san_cap.isOpened()) {
                while (1)
                        {       san_cap.read(san);
                                palma=detect_and_draw (san);
                                camera= cvarrToMat(palma);
                                printf("prueba 3");
                                overlayImage(camera, off, dst, Point(0,0), 110,110);
```

```
                                        if(PantallaPrincipal){
                                        printf("prueba 5");
                                        actStop=false;
                                        overlayImage(dst, musica, dst,
Point(posicionMusic[0],posicionMusic[1]), posicionMusic[2], posicionMusic[3]);
                                        overlayImage(dst, paint, dst,
Point(posicionPaint[0],posicionPaint[1]), posicionPaint[2], posicionPaint[3]);
                                        overlayImage(dst, video, dst,
Point(posicionVideo[0],posicionVideo[1]), posicionVideo[2], posicionVideo[3]);
                                        if(actVideo==true || actMusic==true|| actPaint==true ){
                                                PantallaPrincipal=false;
                                        }
                                        cv::imshow("REALIDAD AUMENTADA", dst);
                                        }
                                        else{

                                if(actVideo)
                                {
                                        actMusic=false;
                                        actPaint=false;

                                        overlayImage(dst, frameV1, dst,
Point(posicionV1[0],posicionV1[1]), posicionV1[2], posicionV1[3]);
                                        if(actVideo1)
                                        {       frame1 = skipNFrames(capture, 8);
                                                frameV1 = cvarrToMat(frame1);
                                        }
                                        overlayImage(dst, frameV2, dst,
Point(posicionV2[0],posicionV2[1]), posicionV2[2], posicionV2[3]);
                                        if(actVideo2){
                                                frame2 = skipNFrames(capture1, 8);
                                                frameV2 = cvarrToMat(frame2);
                                        }

                                        overlayImage(dst, frameV3, dstV,
Point(posicionV3[0],posicionV3[1]), posicionV3[2], posicionV3[3]);
                                        if(actVideo3){
                                                frame3 = skipNFrames(capture3, 8);
                                                frameV3 = cvarrToMat(frame3);
                                        }
                                        if(actStop)                             {
                                        PantallaPrincipal=true;
                                        actVideo=false;
                                }
                                cv::imshow("REALIDAD AUMENTADA", dstV);
                        }
                                        }
                                if(actMusic)    {
                                        actVideo=false;
                                        actPaint=false;
                                        overlayImage(dst, Img1, dst,
Point(posicionImg1[0],posicionImg1[1]), posicionImg1[2], posicionImg1[3]);
```

```
                                        overlayImage(dst, Img2, dst,
Point(posicionImg2[0],posicionImg2[1]), posicionImg2[2], posicionImg2[3]);
                                        overlayImage(dst, Img3, dst,
Point(posicionImg3[0],posicionImg3[1]), posicionImg3[2], posicionImg3[3]);
                                        overlayImage(dst, Img4, dstS,
Point(posicionImg4[0],posicionImg4[1]), posicionImg4[2], posicionImg4[3]);
                                        if(actSonido1)     {
                                        audio[0]->play();
                                        audio[1]->stop();
                                        audio[2]->stop();
                                        audio[3]->stop();
                        }
                                if(actSonido2)
                                {
                                        audio[0]->stop();
                                        audio[1]->play();
                                        audio[2]->stop();
                                        audio[3]->stop();
                        }
                                if(actSonido3)
                                {
                                        audio[0]->stop();
                                        audio[1]->stop();
                                        audio[2]->play();
                                        audio[3]->stop();
                        }
                                if(actSonido4)
                                {
                                        audio[0]->stop();
                                        audio[1]->stop();
                                        audio[2]->stop();
                                        audio[3]->play();
                        }
                                if(actStop)         {
                                        PantallaPrincipal=true;
                                        actMusic=false;}
                                cv::imshow("REALIDAD AUMENTADA", dstS);
                                }
                                        waitKey(200);
                                        int key = cv::waitKey(waitKeyValue);
                                        if(key!=-1)cout<<key<<endl;
                                        if (key == 27 || key == 1048586) {
                                                if (waitKeyValue == 10)waitKeyValue = 0;
                                                else waitKeyValue = 10;
                                        }
                                }
                } else cout << "videoCapture not working" << endl;
                return 1;
        }
```

APÉNDICE A

ALGORITMO: Conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

DETECCIÓN: Recibe como entrada una imagen preprocesada y devuelve la mano detectada.

GESTO: Es una forma de Comunicación no verbal ejecutada con Alguna Parte del Cuerpo, y producida por el movimiento de las articulaciones y músculos de brazos, manos y cabeza.

IMÁGENES EN 2D: Imágenes ordinariamente creadas por ordenador y que no permiten (soportan) comportamientos propios de cuerpos en el espacio.

IMAGEN DIGITAL: Una imagen digital es una representación bidimensional de una imagen a partir de una matriz numérica, cada uno con un valor de intensidad o brillo asociado con cada pixel en la imagen. Es una representación gráfica, generada por computador o creada a través de algún dispositivo de captura como cámara digital, cámara web, escáner, etc. Y que se almacena en forma binaria.

INTERACCIÓN: Es una acción recíproca entre dos o más objetos, sustancias, personas o agentes.

MACADORES: Si se emplea una imagen como "marcador", al captar la imagen en cuestión con la cámara el proceso se ejecuta la aplicación correspondiente, reconocida la imagen se producirá la acción que corresponda.

PÍXEL: El elemento básico de una imagen es el pixel, es la abreviatura de las palabras inglesas "picture element"; de aplicabilidad individual un color o una intensidad que se puede diferenciar de los otros mediante un procedimiento determinado.

PROTOTIPO: Es un ejemplar o primer molde en que se fabrica una figura u otra cosa.

REALIDAD AUMENTADA: Superposición de datos generados por computadora sobre el campo de visión primaria

RECONOCIMIENTO: La acción de distinguir a una cosa, una persona o una institución entre las demás como consecuencia de sus características y rasgos se la designa como reconocimiento.

SOFTWARE: Al equipamiento lógico o soporte lógico de un sistema informático, comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware.

TECNOLOGÍA: Es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas.