Universidad de las Ciencias Informáticas

Facultad 5

Visualización Gráfica y Realidad Virtual

Sistema de Interacción Natural con Entornos Virtuales.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Alberto Yusniel Rivero Nuñez

Tutor: Ing. Orlay García Ducongé

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

_____	_____
Firma del autor	**Firma del tutor**
Alberto Yusniel Rivero Nuñez	**Orlay García Ducongé**

DATOS DE CONTACTO

Nombre y apellidos: Orlay García Ducongé

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Categoría docente: Profesor Instructor

Categoría científica: Ingeniero

Años de experiencia en el tema: 4 años

Años de graduado: 3 años

Correo electrónico: oducunge@uci.cu

AGRADECIMIENTOS

A mi mamá, que siempre tiene amor para su hijito malcriado.

A mi papá, uno de los hombres más inteligentes que he conocido.

A mis hermanos, que tanto le han tenido que aguantar a este chiquillo malcriado.

A todos los que compartieron tiempo conmigo en los continentes de Kalimdor, Eastern Kingdoms, Outlands, Northrend y restantes lugares de ese reino, que no pertenece precisamente a los olvidados.

A todos los que de alguna manera, explícita o implícita, han puesto su granito de arena para que se haga realidad mi sueño.

Para ser sincero, no encuentro forma de agradecerles por todo a la mitad de ustedes, ni la mitad de lo que realmente querría y lo que yo querría es menos de la mitad de lo que la mitad de ustedes se merece.

DEDICATORIA

A Cacha, mi mami querida, que posee la increíble habilidad de entenderme…

A Eivis, Durdú, Tata, Cheo y Yane, mis hermanos queridos…

A Papi, ese estudioso incansable…

RESUMEN

En el presente trabajo se propone una alternativa para la comunicación de los humanos con un mundo virtual. La principal idea es lograr la comunicación con el sistema mediante el uso de cámaras web, siendo este dispositivo de bajo costo y de fácil adquisición. Usando la biblioteca para visión por computadoras OpenCV, que brinda un gran número de funciones para el trabajo con imágenes, posibilitando procesar la información proveniente de la cámara y tomar acciones respecto a los datos obtenidos. El sistema utiliza la visión por computadoras para generar una interacción más amable y natural con los entornos virtuales. Este tipo de interfaz ha pasado a jugar un rol importante en las nuevas tecnologías, incluyendo Realidad Virtual (RV), robótica, ingeniería médica y juegos digitales. Su meta principal es lograr que las computadoras sean capaces de reconocer los gestos de los usuarios y realizar las acciones apropiadas. La interacción se logra a través de la captura de imágenes, en las cuales se identifican estos gestos.

Palabras clave:

OpenCV, Mundo Virtual, Visión por Computadoras, Entornos Virtuales.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La constante evolución de las tecnologías y los crecientes avances que se han alcanzado en materia de informática, han permitido al hombre incursionar en nuevas esferas de investigación, como la referente a la RV, abarcando todos los mundos virtuales que existen dentro de los sistemas generados por las computadoras. Con la utilización de las nuevas tecnologías y el empleo de RV el hombre puede crear un mundo en el cual el tiempo y espacio son modificados para obtener información hasta ahora desconocida. La interacción del hombre con estos mundos virtuales es objeto de investigación constante, con el fin de lograr que su manipulación sea de la forma más natural y realista posible.

La RV se puede clasificar en dos tipos: **inmersiva** y **no inmersiva**. Los métodos inmersivos son aquellos en los que está presente un mundo virtual creado por computadoras, las personas utilizan dispositivos especiales para interactuar con este mundo virtual, estos dispositivos pueden ser: cascos, guantes, mandos o controles a distancia, sensores de movimiento y posicionamiento, y otros dispositivos no convencionales. La utilización de estos dispositivos hace que la integración con el mundo virtual sea de una forma más realista.

En los métodos no inmersivos el mundo es representado a través de una ventana de escritorio. Este método no necesita de dispositivos especiales para que las personas puedan interactuar con el sistema, permitiendo interactuar en tiempo real con diferentes personas en espacios y ambientes que en realidad no existen. Los usuarios controlan estos métodos con los medios convencionales de comunicarnos con las computadoras, tales como el *mouse* y el teclado.

Con la meta de mejorar la relación entre las personas y las computadoras, se investigan nuevas formas de interactuar con ellas. El principal objetivo es lograr que las máquinas sean capaces de “comprender” las acciones tomadas por las personas, permitiendo una interacción más amable e intuitiva. De esta forma la comunicación con las máquinas sería más natural.

La interacción natural busca crear sistemas que “entiendan” las acciones comunicativas de las personas (gestos, expresiones y movimientos), teniendo en cuenta que una actividad natural es aquella para la que los humanos están hechos y que se encuentra implícita en la mente y en el cuerpo. Estas acciones se utilizan como base para generar una interacción más amable de las personas con su entorno y reducir al mínimo la carga de pensamiento consiente necesaria para hacer funcionar el sistema, dejando que éstas se concentren en sus objetivos.

Al diseñar un sistema para interactuar con las computadoras no se debe pensar sólo en la apariencia del mismo, se debe tener en cuenta la forma en que las personas van a interactuar con él y la facilidad o complejidad con que lo hacen. En este tipo de sistemas se pretende anular los dispositivos que se utilizan para la obtención de respuestas por parte del sistema, en otras palabras, se pretende una interacción intuitiva, con el fin de respetar la percepción humana, haciendo de esto un acto fácil y atractivo.

Actualmente en la facultad los proyectos que trabajan con entornos virtuales utilizan los medios convencionales de interacción para comunicarse con estos entornos, impidiendo que la interacción entre las personas y las computadoras se realice de una forma sencilla, que permita al usuario lograr una comunicación más realista y natural.

Por tanto, la **situación problémica** se puede resumir en: La falta de naturalidad en la manipulación de los objetos constituye una barrera para suprimir la percepción de estar interactuando con una máquina.

Teniendo en cuenta lo anterior se formula el siguiente **problema científico:** ¿Cómo lograr una interacción natural que mejore la relación entre las personas y los entornos virtuales?

El **objetivo general** de la investigación para darle solución al problema científico es: desarrollar un sistema que permita interactuar de forma natural con los entornos virtuales.

El cual precisa como **objeto de estudio**: Interacción con entornos virtuales.

Luego queda definido el siguiente **campo de acción:** Manipulación de objetos en entornos virtuales.

Idea a defender: Con este trabajo se alcanzará una vía de comunicación natural con los sistemas de RV, facilitando la interacción entre las personas y los entornos virtuales.

Para cumplir el objetivo se proponen las siguientes **tareas investigativas**:

- Elaboración del marco teórico a partir del estado del arte existente sobre el tema actualmente.
- Selección de la biblioteca de procesamiento de imágenes que responda a las necesidades del sistema.
- Selección de las tecnologías que no involucren dispositivos convencionales para obtener una interacción natural.
- Selección de herramientas y lenguajes para la implementación del sistema.

- Diseño del sistema de interacción natural.
- Implementación del sistema de interacción natural.
- Validación del sistema elaborado.

Entre los métodos científicos utilizados se destacan:

Métodos Teóricos:

- **Analítico-Sintético:** Con el uso de este método de investigación se van a analizar las informaciones obtenidas, mediante su desglose, para después realizar una síntesis de las mismas y arribar a las principales ideas.
- **Histórico-Lógico:** Mediante este método se analizará la trayectoria y la evolución de las diferentes formas de manipular los objetos en laboratorios virtuales para seleccionar la más adecuada.
- **Modelación:** Este método se utilizará para crear un prototipo funcional del sistema de interacción natural.

Métodos Empíricos:

- **Consulta de las fuentes de información**: Para seleccionar la información necesaria para construir el marco teórico.
- **Consulta de especialistas:** Para recibir los criterios de validación sobre la aplicabilidad y utilidad de lo realizado.
- **Observación:** Para reunir información visual sobre lo que el objeto de estudio hace y cómo se comporta.
- **Pruebas:** Para valorar el desempeño del sistema elaborado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo presenta los conceptos relacionados con la RV y las distintas formas de interacción que existen actualmente con los entornos virtuales.

1.1 Conceptos fundamentales del tema

Los siguientes conceptos son de gran importancia para el entendimiento del tema en cuestión:

- Realidad virtual
- Imagen digital
- Procesamiento de imágenes digitales
- Técnicas de interacción

Realidad virtual

Peter Dyson en su diccionario de informática define RV como: “entorno generado por computadora, el cual presenta una ilusión de realismo”.

Otros autores la definen como:

- Sistema o interfaz informático que se encarga de generar entornos sintéticos que se suceden en tiempo real, es decir, la RV lo que propone es la representación de determinadas cosas, situaciones, a través de medios electrónicos, como por ejemplo la computadora, los cuales darán lugar a una realidad perceptiva sin soporte objetivo y que solamente encontrará su razón de ser y entidad dentro del ordenador que la haya inventado o propuesto, por esto es que puede llamarse también una pseudo realidad alternativa (1)
- Es un sistema de computación que crea un mundo artificial, donde el usuario tiene la impresión de estar en ese mundo, posibilitando la habilidad de navegar y manipular objetos (2).
- Simulación de medios ambientales y de los mecanismos sensoriales del hombre mediante el uso de las computadoras, de tal manera que se le proporcione al usuario la sensación de inmersión y la capacidad de interacción con medios ambientales artificiales (3).

- Es un sistema de computación usado para crear un mundo artificial donde que el usuario tiene la impresión de estar en ese mundo y la habilidad de navegar y manipular objetos en él (4).
- La RV es un camino que tienen los humanos para visualizar, manipular e interactuar con computadoras y con información extremadamente compleja (5).

En los últimos tiempos la vía más lucrativa para el campo de RV radica en los juegos por computadoras, pero en este campo también se sitúa la informática educativa, la cual explota todas las potencialidades de la RV en el desarrollo de laboratorios, talleres y otros entornos simulados.

Imagen digital

Una imagen puede ser definida matemáticamente como una función bidimensional, $f(x, y)$, donde x y y son coordenadas espaciales (en un plano), y f en cualquier par de coordenadas es la intensidad o nivel de gris de la imagen en esa coordenada.

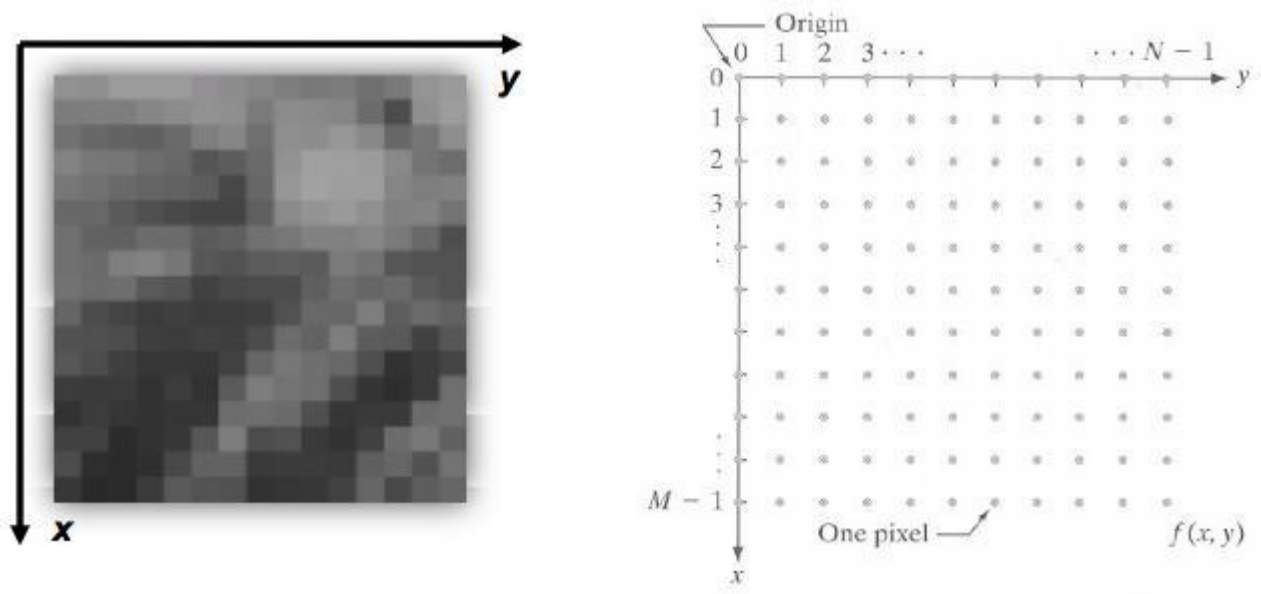

Figura 1: *Píxeles de una imagen digital*

Cuando x, y y los valores de f son todas cantidades finitas, discretas, decimos que la imagen es una *imagen digital*. Una imagen digital se compone de un número finito de elementos, cada uno con un lugar y valor específicos. Estos elementos son llamados píxeles (6).

Así mismo, una imagen digital puede ser convenientemente representada por una matriz $f(x,y)$ de tamaño M x N de la forma:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(0,1) & \cdots & f(0,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Figura 2: *Imagen digital representada por una matriz*

Los elementos de la matriz (píxeles) en una *imagen monocromática* (*niveles de gris*) típica son del orden de 2^8 o 256 niveles de gris de tal manera que $f(x,y)=0$ y $f(x,y)=255$ donde 0 representa el negro y 255 el blanco, por tanto, pueden ser representados como caracteres en la mayoría de los lenguajes de programación. Las columnas y las filas de las matrices de imágenes digitales, por tanto, tienen un rango de: $0 \leq i \leq N-1, 0 \leq j \leq M-1$.

Figura 3: *Imagen digital*

Procesamiento de imágenes digitales

Los distintos especialistas no se ponen de acuerdo para decir dónde termina el campo del Procesamiento Digital de Imágenes (PDI) y dónde empiezan otros campos como el Análisis de Imágenes y la Visión por Computadora (VC).

Uno de los paradigmas (6) más utilizados para definir el procesamiento de imágenes digitales es la utilización de tres etapas de procesos computarizados en el siguiente orden: bajo, medio y alto nivel.

Procesos de bajo nivel:

Se caracterizan por que sus entradas son imágenes y sus salidas también. Utilizan operaciones como el *pre-procesamiento de imagen* para reducir el ruido, procesos subjetivos como la *mejora del contraste* y los *filtros de enfoque o suavizado*.

Procesos de nivel medio*:*

Se caracterizan por que sus entradas son generalmente imágenes, pero sus salidas son atributos extraídos de esas imágenes (*contornos, bordes, identidad de objetos individuales)*. Llevan a cabo operaciones como *segmentación y clasificación* de objetos individuales.

Procesos de alto nivel*:*

Implica el obtener algún significado de un *conjunto de objetos reconocidos* – análisis de imágenes – y, finalmente, realizar las funciones *cognitivas* asociadas con la vista.

Técnicas de interacción con las computadoras.

Estas técnicas son llevadas a cabo para realizar una tarea a través de una interfaz de usuario, que incluye *software* y *hardware*. Este tipo de interfaz busca la inmersión del usuario en una realidad alternativa, tanto física como mentalmente, teniendo en cuenta el sentido de presencia que tiene una persona dentro de un entorno, provocando la sensación de que se encuentra en él. La inmersión mental hace referencia al estado de estar profundamente relacionado con el entorno virtual y la física es cuando el cuerpo interactúa con un medio, que captura los estímulos físicos con el uso de dispositivos sintéticos.

Tabla 1: Tareas y metas de la interacción

Tareas de interacción universales:	**Metas del diseño de una interacción:**
Selección: • Seleccionar uno o más objetos dentro de un conjunto **Manipulación:** • Especificar la posición y orientación de un objeto • Especificar la escala, forma y otros atributos **Navegación:**	**Rendimiento (medida cuantitativa)** • Eficiencia • Precisión • Productividad **Usabilidad (experiencia cualitativa)** • Fácil de usar • Fácil para aprender • Cómodo para el usuario **Utilidad**

• Recorrido: Componente del motor • Señalización: Componente cognitivo **Control del sistema:** • Otras interacciones, usualmente llevadas a cabo usando comandos • Puede estar compuesto por otras tareas	• La interacción ayuda a lograr las metas del sistema • Una interfaz relativamente transparente para el usuario permite que este se concentre en las tareas.

Técnicas para la selección de los objetos:

Para seleccionar objetos dentro de un entorno virtual existen dos técnicas principales, la selección local y distante. En la selección local el objetivo está cerca del usuario, haciendo posible la interacción directa con este y en el caso de la distante, se usa cuando no hay un contacto directo entre el usuario y el objeto, esta puede ser realizada a través de dispositivos láseres.

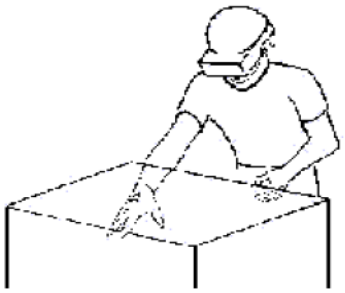

Figura 4: *Avatar tocando un objeto virtual*

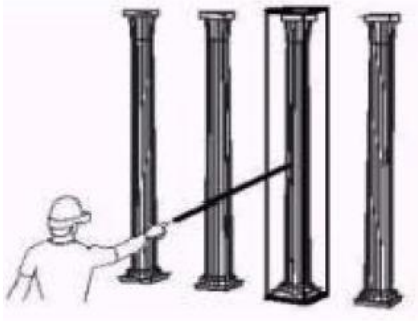

Figura 5: *Avatar utilizando un rayo para seleccionar un objeto en un entorno virtual*

Técnicas para la manipulación de los objetos:

Uno de los modos más importantes e interactivos para manipular un objeto dentro de un entorno virtual es la especificación de la posición y la orientación de este. Esto se consigue haciendo un seguimiento de las acciones del usuario, permitiéndole mover objetos como en el mundo real, con lo cual la interacción debería ser realista.

Tareas básicas de manipulación:

- Seleccionar

- Mover
- Rotar

Tabla 2: Variables que modifican los usuarios y clasificación de las técnicas de interacción

Variables que los usuarios modifican de los objetos:	**Clasificación de las técnicas de manipulación:**
• Distancia • Tamaño • Traslación • Rotación • Densidad	• **Isomorfas (realistas)** • **No – isomorfas** – La mayor parte de las técnicas son de este tipo – Usan herramientas virtuales como: ▪ Rayos laser ▪ Brazos elásticos

1.2 Sistemas de interacción no convencionales

Los sistemas de interacción no convencionales (7) son aquellos que les permiten a los usuarios experimentar la sensación de encontrarse dentro de un ambiente específico. Para comunicarse con las computadoras que simulan el mundo virtual, se utilizan diversos dispositivos de entrada, uno de los más utilizados es el mouse tridimensional, extensión del mouse común, usado en los equipos de cómputo personales con un conjunto de botones que indican una cierta acción pero que, además, añaden la posibilidad de mover los objetos en los ejes de rotación cartesianos, con lo que se extiende su funcionalidad a espacios 3D.

Guantes sensitivos

Los guantes sensitivos funcionan como un localizador que percibe la posición y orientación de la mano; para ello, cuentan con algunos sensores de flexión que permiten medir la curvatura de los dedos.

Figura 6: *Guantes sensitivos*

Cuando se mueve el guante en el espacio, el cursor en 3D del ambiente virtual reacciona y se dirige a la dirección indicada por el guante; así, se pueden tomar, arrastrar y soltar objetos virtuales, moverlos horizontal o verticalmente, en profundidad y rotarlos alrededor de los tres ejes coordenados.

Mandos

El *Wiimote* conocido también como Mando *Wii*, *Wii Remote*, es el mando principal de la videoconsola de sobremesa *Wii* producida y distribuida por Nintendo muy similar al control remoto de televisión, que además de los botones que posee, contiene un acelerómetro en los tres ejes de posición, una cámara de infrarrojos de alta resolución, un altavoz, un motor de vibración, cuatro luces tipo LED y conectividad inalámbrica vía *Bluetooth*.

Figura 7: *Mandos*

Cascos [head-mounted display (HMD)]

Este sistema de interacción no convencional despliega la información en un vidrio parcialmente plateado para evitar que se aparte la vista de lo que se está observando en el EV. Incorporan pequeños monitores que se colocan uno frente a cada ojo. Cada monitor visualiza la perspectiva que el ojo correspondiente vería en un ambiente real.

Figura 8: *Casco*

1.3 Visión por computadoras

En los últimos años el término “visión por computadora” o lo que es lo mismo “visión artificial” ha estado muy de moda. La experiencia en el mundo actual está cubierta por una gran variedad de objetos, animados e inanimados. Así pues, la visión es un medio para un fin, conocer el mundo observándolo; la visión artificial es exactamente lo mismo salvo que el medio por el cual se adquiere el conocimiento ahora es un instrumento de cómputo en lugar del cerebro de algún ser vivo. Sin duda, esto es una explicación muy amplia y necesita de más rigor científico para construir una definición. El tema de la visión artificial es extenso: los asuntos tales como la restauración de imágenes, mejoramiento de imagen, inspección visual automatizada, visión robótica, escenas tridimensionales, percepción y cognición visual todas forman parte del término “Visión por Computadora”.

La visión artificial incluye muchas técnicas que son útiles para sí misma. Ejemplo de ello es el procesamiento de imágenes (que se refiere a la transformación, codificación, y transmisión de las imágenes) y reconocimiento de patrones, entre los cuales los patrones visuales son sólo una instancia.

Más explícitamente, la visión artificial (8) incluye técnicas para la descripción útil de la forma y del volumen, para modelos geométricos, y para el llamado proceso cognoscitivo. Así, aunque la visión artificial se refiere ciertamente al procesamiento de imágenes, estas imágenes son solamente la materia prima de una ciencia mucho más amplia, la misma que se esfuerza en la última instancia para emular las capacidades perceptivas del hombre.

Algunos autores han definido “Visión artificial” como una disciplina:

- Es una rama de la inteligencia artificial que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales por computadora. (2)
- Capaz de interpretar el contenido de escenas naturales. (9)
- Desarrolladora de las bases teóricas y algorítmicas para obtener información sobre el mundo real a partir de una o varias imágenes. (10)
- Consiste en hacer que una computadora vea y ese es un problema no resuelto. (4)

- Tiene como finalidad, extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador. Se trata de un objeto ambicioso y complejo que actualmente se encuentra en una etapa primitiva.

- Algunos glosarios y diccionarios informáticos la denominan como **visión asistida por computadora**, entendiendo por ello "el diseño de sistemas de cómputo para discernir (interpretar) las imágenes visuales." En este caso se define como proceso tecnológico. (8)

- El único dispositivo utilizado para interactuar con el sistema es una cámara web, la cual se encarga de recolectar las imágenes que serán procesadas.

Figura 9: *Cámara web*

1.4 Bibliotecas para sistemas de visión por computadora.

- **IPL (Intel Image Processing Library, siglas en inglés)** - Está orientada al procesamiento de imágenes a bajo nivel. Su desarrollo fue abandonado en el 2000, año en que fue sustituida por IPP (Intel Performance Primitives, siglas en inglés), algo más eficientes y que incluyen otras aplicaciones (como, por ejemplo, sonido, criptografía y vídeo), pero son más difíciles de manejar y no gratuitas (11).

- **CMVision (Color Machine Vision Library, siglas en inglés)-** Es una biblioteca de procesamiento de imágenes en color a bajo nivel gratuita, que dispone de herramientas de binarización y de cálculo de componentes conexas de imágenes de video (60 imágenes por segundo) (12).

- **Mavis:** Es un paquete de visión por computadora. Fue desarrollado inicialmente para un proyecto de robótica llamado "Leaf", está específicamente diseñado para serle útil a robots móviles. En su última versión también se detectan rostros. Esta biblioteca puede ser ejecutada únicamente sobre Windows XP. Está implementada como una dll, por lo que Mavis puede ser usada desarrollando la aplicación en cualquier lenguaje que se conecte con dlls incluyendo C/C++, Visual Basic, Java, Lisp, Prolog (13).

- **Gandalf:** Es una biblioteca de visión por computadora y algoritmos numéricos escritos en C, permite desarrollar aplicaciones portables y de rápido funcionamiento. Permite el uso de estructuras dinámicamente reconfigurables y permite el uso eficiente de la memoria (14).

- **OpenCV (Open Source Computer Vision, siglas en inglés)-** Es una biblioteca libre de visión artificial originalmente desarrollada por INTEL y publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas, en el 2002. La biblioteca es multiplataforma, y puede ser usada en Mac OS X, Windows y Linux. Está orientada al tratamiento de imágenes en tiempo real, lo que es posible si encuentra en el sistema las Primitivas de Rendimiento Integradas de Intel (IPP). La biblioteca implementa funciones de reconocimiento facial y de detección de objetos (15).

OpenCV presenta un conjunto de funciones (15) implementadas en lenguaje C para el procesado de imagen. Las funciones que emplea OpenCV para el reconocimiento de objetos están basadas en los algoritmos propuestos por Paul Viola y Michael Jones y luego mejorados por Rainer Lienhart. Emplea 38 etapas de decisión en su clasificador y en torno a 6000 características en sus funciones de detección de caras. Los datos provenientes del entrenamiento se guardan en archivos XML.

1.5 Estructura de la biblioteca OpenCV

La biblioteca OpenCV está dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus áreas de aplicación destacarían: interacción hombre-máquina; segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura del movimiento; y robots móviles. Proporciona varios paquetes de alto nivel para el desarrollo de aplicaciones de visión. Todos ellos se pueden agrupar en biblioteca de C/C++ dirigidas a usuarios avanzados, a usuarios de nivel medio (ideal para practicar con las distintas técnicas de procesamiento de imágenes y visión).

En definitiva, OpenCV ofrece grandes posibilidades para el tratamiento de imágenes, calibración de cámaras, y otras muchas aplicaciones más. En el caso del seguimiento de objetos, el principal inconveniente es que no ofrece un producto completo, tan sólo algunas piezas que sirven como base para montar sobre ellas un producto final. Otro de los inconvenientes que tiene es la necesidad de utilizar la biblioteca IPL para tener acceso a funciones de bajo nivel. Sin embargo, la presencia de funciones muy interesantes, y las posibilidades ya comentadas que ofrece la biblioteca hacen que estos inconvenientes no sean realmente significantes. La instalación de la biblioteca en Windows es algo problemática ya que se debe registrar los ficheros .dlls y además configurar las variables de entorno, siendo la instalación en Linux mucho más sencilla.

Este conjunto de bibliotecas (15) permiten realizar análisis de la forma y estructura de una imagen, reconocimiento de objetos, análisis y seguimiento de movimiento y reconstrucción de imágenes en 3D.

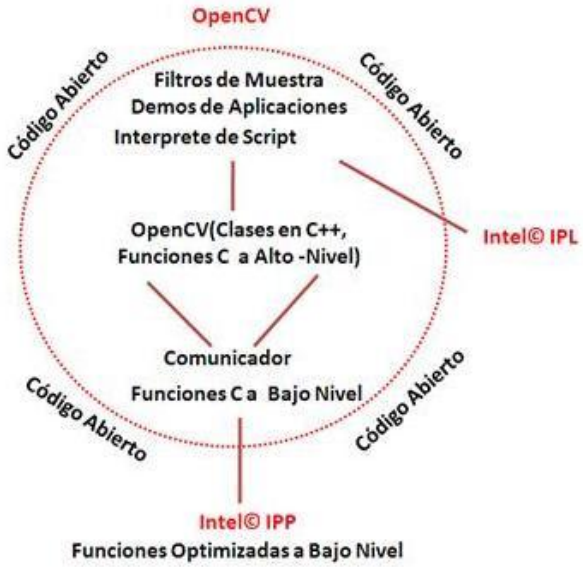

Figura 10: *Estructura de la biblioteca OpenCV*

1.6 Técnicas para el reconocimiento de los gestos de la mano

Este tipo de interfaz ha pasado a jugar un rol importante en las nuevas tecnologías, incluyendo RV, robótica, ingeniería médica y juegos digitales. Su meta principal es que las computadoras sean capaces de reconocer los gestos de los usuarios y realizar las acciones apropiadas. La interacción se logra a través de la captura de imágenes, en las cuales se identifican los gestos de los usuarios.

1.6.1 Análisis de la escena para el reconocimiento de los gestos de la mano y la detección de la dirección de los objetos.

Para expandir el alcance de los comandos visuales y que sean ejecutados en tiempo, el gesto de la mano es aumentado por otro objeto, una flecha o un lápiz, en este caso es un lápiz el que brinda información adicional como la dirección. Vamos a asumir una secuencia de imágenes que contienen una mano en forma de pata, un puntero (objeto firma), y varios objetos extras.

Figura 11: *Gestos de la mano*

Generalmente el proceso cuenta con dos etapas, desconectado y conectado. En la primera etapa un set de aprendizaje es utilizado para extraer las características de la escena. En la segunda etapa una imagen desconocida es analizada para reconocer los gestos de la mano en ella y la dirección del puntero (objeto firma).

Por otra parte, el puntero puede ser colocado en cualquier dirección entre 0 y 360 grados de forma circular. Los objetos extras pueden ser cualquier cosa no similar a la mano en forma de pata o al objeto puntero (un lápiz en este caso). El objeto puntero debe ser distinguible al menos de formas similares, esto se logra en el ejemplo poniéndole un marcador al final. Esto se hace para hacerlo una forma de flecha distinguible. Para un sistema que reconozca una secuencia de estos gestos, debe ser capaz de determinar (a) que gesto está haciendo la mano del usuario, y (b) en qué dirección está apuntando la flecha.

La flecha, en este caso el lápiz, es movida por la mano y colocado en una dirección deseada y la mano forma un gesto. La combinación del gesto de la mano y la dirección de la flecha constituye un comando (instrucción). La forma es codificada por un alfabeto de letras (A a la Z) y este último por un número (0 a 360).

La siguiente figura muestra una secuencia de imágenes empezando por una instrucción o comando.

Figura 12: *Secuencia de imágenes empezando por una instrucción o comando*

Esta imagen que constituye una instrucción es llamada Cuadro de Instrucción (CI). La secuencia termina con otro CI. Todas las imágenes entre estos dos CI son imágenes en movimiento, en las cuales no se encuentra ningún comando involucrado. Todas las imágenes son primero analizadas para determinar si son imágenes de movimiento o CI. Posteriormente los CI son analizados para determinar la instrucción a la que hacen referencia, mientras que las imágenes de movimiento son descartadas. La secuencia de video comienza con una instrucción de inicio y es detenida por una instrucción de parada (16).

1.6.2 Selección y reconocimiento de los gestos de la mano.

Para explicar el esquema propuesto se construye una colección de gestos, la misma es realizada tomando imágenes de un video grabado con una gran variedad de personas, incluyendo adultos, niños, hombres y mujeres. Las imágenes en color son convertidas a imágenes en gris intenso para eliminar el matiz y la saturación mientras se mantiene la iluminación (16). El proceso comienza con la captura de un *frame* (cuadro) por la cámara conectada a la computadora que luego es procesado para detectar regiones candidatas de interés. Después de que una mano es detectada en la región de interés es seguida por el *hand tracker* (sistema de seguimiento), todos los procesos subsecuentes son en esta región. Mientras es detectada la mano, características estáticas y dinámicas son recogidas, las características estáticas son filtradas por un ARTMAP *Neural Network Classifier* (clasificador de redes neuronales) y las características dinámicas son analizadas para obtener características de alto nivel.

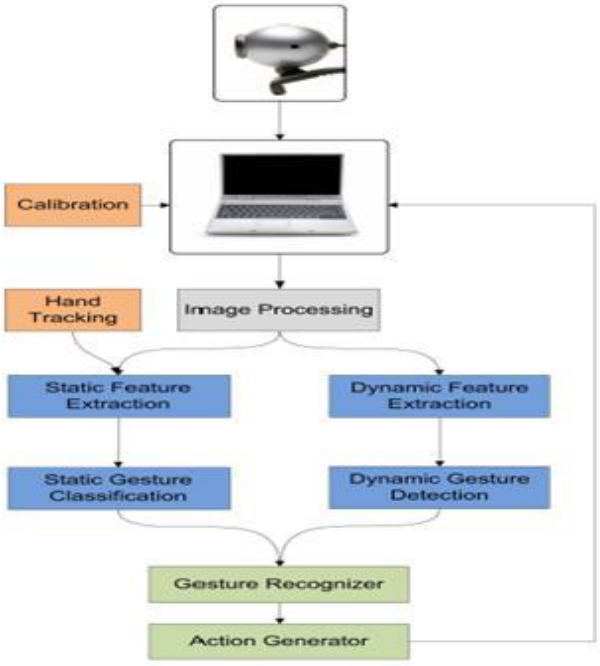

Figura 13: *Proceso para el reconocimiento de los gestos de la mano*

1.6.3 Reconocimiento estático y dinámico de los gestos de las manos para la integración humana con las computadoras.

La información estática y dinámica es analizada por el *Gesture Recognizer* (reconocimiento de gestos), quien le comunica al *Action Generator* (generador de acciones) que tipo de gesto es el reconocido. Finalmente el generador de acciones pasa los comandos o instrucciones necesarias al sistema para crear el resultado final.

Inicialmente una referencia o *frame* clave es colocado. Esto se hace unos pocos segundos después de que la cámara web es activada para permitir un mejor tiempo en la inicialización del hardware. Este *frame* clave pasa a ser el primer *frame* al principio de ejecución de la aplicación, el cual es usado para determinar diferencias en las sub-secuencias de *frames*, ayudando en el proceso de detección de movimiento.

Después que este proceso es completado, regiones candidatas son analizadas para detectar tipos de dedos. Si alguna característica de una mano humana es encontrada esta región, la misma es considerada como contenedora de una mano, y es procesada como una sub-imagen utilizando el *frame* clave. Una vez que la mano es detectada características de gestos estáticos son extraídos de *frames* individuales. Concurrentemente, siguiendo el movimiento de los gestos de la mano las características dinámicas son establecidas.

El sistema especifica algunas restricciones que los usuarios deben seguir en orden para que la detección y el reconocimiento de la mano humana se hagan de forma robusta. Las restricciones son:

- Los usuarios y el entorno en que se encuentran deben ser estáticos, no se deben mover cuando se está ejecutando un gesto.
- El color y la intensidad característica de la mano deben ser distintos al color de fondo.

La meta principal de este artículo (17) es ofrecerle una forma natural e interactiva de comunicación con las computadoras.

1.6.4 Reconocimiento de los gestos de la mano usando una cuadrícula de memoria auto-asociativa (CMA).

En el método propuesto el reconocimiento de los movimientos de la mano está dividido en tres fases principales. La detección de la región de la mano, la extracción de las características y su reconocimiento.

La detección de la región de la mano es conseguida mediante la inyección de color saturado basado en el espacio de *Hue Saturation* (Saturación de Contraste).

La técnica es confiable, es relativamente inmune al cambio de la luz ambiental y provee buena cobertura del color de la piel humana. En la etapa intermedia, los momentos invariantes son calculados, formando un vector de características descriptivas del gesto en un instante del tiempo en particular. Finalmente este vector de características es usado por el entrenador de memoria auto-asociativa para reconocer el gesto realizado. La estructura de esta memoria asociativa está basada en una técnica computacional llamada álgebra de cuadrículas.

Se trabaja con imágenes estáticas en formato PGM (120x150 píxeles). En estas imágenes, lo importante es el reconocimiento de los gestos de la mano. Consecuentemente, las manos son segmentadas de la imagen.

Segmentación de la mano.

Para segmentar la mano la imagen es filtrada usando inyección de color en la segmentación de las imágenes a color en el subespacio de *Hue Saturation* (HS). Esto significa una transformación linear en el espacio RGB. Este método ha sido desarrollado por Blanco (18), añadir un vector de color a la imagen capturada en el espacio RGB con el objetivo de incrementar la separación entre el objeto y los tipos de fondos en el plano HS.

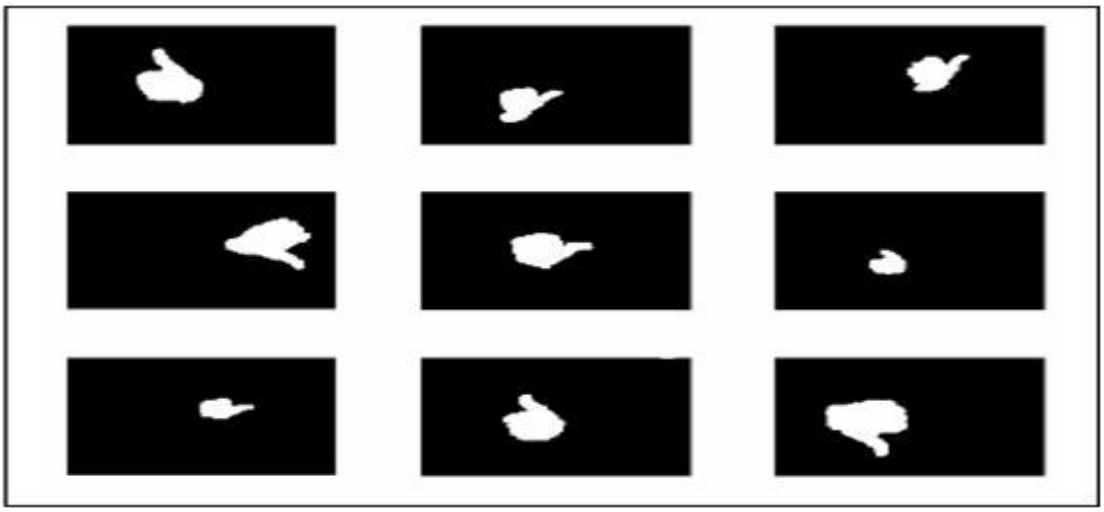

Figura 14: *Segmentación de los gestos de la mano*

Extracción de los gestos

Para la extracción de los gestos los momentos invariantes son calculados a partir de un vector con características descriptivas de los gestos, conservando las características geométricas de la imagen en un instante de tiempo dado. En particular, HS, define siete valores, calculados mediante la normalización de momentos centrales hasta el tercer orden, estos son invariantes a la escala del objeto, posición y orientación.

1.9234027e-001	4.0453688e-003	1.9599183e-003	3.1473352e-004	2.4653730e-007	1.6681613e-005	1.7971640e-008
1.9890711e-001	5.2776464e-003	2.7400779e-003	1.2651924e-004	4.5168441e-009	-4.9337147e-006	7.4356086e-008
1.9190930e-001	4.5732637e-003	1.5880146e-003	1.2670570e-004	4.9615690e-008	2.4483235e-006	2.7723348e-008
1.9095454e-001	2.3843508e-004	2.3346995e-003	8.7643965e-005	2.9433852e-008	-1.3316620e-006	2.6560261e-008
1.8254910e-001	2.4002872e-003	1.2814243e-003	1.8046667e-004	8.6745123e-008	8.6580191e-006	-2.6129019e-009
1.7308121e-001	5.0915424e-004	4.6345203e-004	4.4898371e-005	3.8656339e-009	6.4699457e-007	-5.1964847e-009
1.9286379e-001	6.4966967e-003	1.1378315e-003	2.9612510e-004	1.6784468e-007	2.3566927e-005	-3.7074223e-008
3.1233498e-001	1.2898774e-002	1.5465407e-001	1.6801571e-001	2.7004981e-002	1.7941773e-002	-2.0614040e-003
1.8246349e-001	3.3471033e-004	1.5370682e-003	1.1531753e-004	4.8383392e-008	1.6412929e-006	4.0191583e-009

Figura 15: *Valores calculados mediante la normalización de momentos centrales*

Reconocimiento usando cuadrículas de memoria auto-asociativa.

En los años recientes las operaciones con matrices, basadas en cuadrículas de memoria auto-asociativa han encontrado una amplia aplicación en las ciencias de la ingeniería. En estas aplicaciones, las usuales operaciones con matrices de adición y multiplicación son reemplazadas por las correspondientes operaciones en enrejados de memoria. Las operaciones con matrices inducidas con cuadrículas conllevan a una perspectiva totalmente diferente de un tipo de transformaciones no lineales.

El reconocimiento de patrones es una aplicación del modelo CMA. El concepto de este modelo proviene de la teoría del álgebra de las imágenes desarrollada por Ritter. La regla fundamental de aprendizaje que se aplica en la teoría de CMA es la siguiente:

$$W_{xy} = \bigwedge_{\xi=1}^{k} [y^{\xi} \times (-x^{\xi})] \quad y \quad M_{xy} = \bigvee_{\xi=1}^{k} [y^{\xi} \times (-x^{\xi})]$$

Figura 16: *Fórmula que se aplica en la teoría de cuadrícula de memoria auto-asociativa*

Resultados.

La memoria de cuadrículas auto-asociativa (19) fue usada para salvar y recobrar un conjunto de vectores con características descriptivas, debido a esto la implementación ha sido dividido en dos fases. Primero el

vector o patrón es almacenado, calculando la respectiva matriz de peso. Segundo, empezando por una configuración arbitraria, la memoria es estabilizada en el patrón almacenado, el cual está más cercano a las configuraciones iniciales en términos de balance. En caso de que la versión del patrón almacenado sea incompleta o demasiado chillona (*noisy*), la red debe ser capaz de reconocer el patrón originalmente almacenado.

1.7 Reconocimiento de gestos soportados en OpenCV

En OpenCV existen funciones diseñadas para generar plantillas de imágenes de movimiento que pueden ser usadas para determinar rápidamente cuando ocurre un movimiento, cómo ocurre, y en qué dirección pueden ser usadas para el reconocimiento de manos y piernas. También permite definir regiones de movimiento producidas por el movimiento de objetos de interés, no siendo necesario el cálculo de la dirección del movimiento para toda la imagen.

En OpenCV también existe soporte para la detección de gestos estáticos (20), que pueden localizar la posición de la mano y definir su orientación en la imagen y crear una imagen con la máscara de la mano. Esto puede ser usado potencialmente para reconocer los variados gestos de la mano. Debido a que este método usa dos cámaras, se decidió basar el reconocimiento en las funciones de plantillas de movimientos (patrones).

1.7.1 Captura o seguimiento de movimiento.

Para la captura de movimientos (15) y preservar la información respecto a los mismos se diseña una clase *Motion*, en la que se guarda el inicio y fin de cada movimiento realizado. También se guarda el centro del movimiento y su orientación en cada *frame*. En esta parte es donde realmente se depende de las capacidades de OpenCV.

En concreto se usan las siguientes funciones: cvUpdateMotionHistory, cvCalcMotionGradient, cvSegmentMotion y cvCalcGlobalOrientatio. Cuando el movimiento termina se compara con plantillas de movimiento previamente definidas para buscar si el movimiento es uno de aquellos en los que se está interesado.

1.7.2 Comparación de movimiento.

La plantilla de movimiento (*Class Motion Template*) es el movimiento deseado y cuando se compara con el movimiento de la captura (movimiento capturado) se provee información de similitud. Primero se compara el tiempo de duración, luego todas las direcciones de movimiento. Estos datos tienen que estar

cercanos a los valores de la plantilla. Si se pasa la prueba se compara el curso (la dirección) del movimiento. Para esto se usa una orientación de movimiento que ha sido guardada por el seguidor de movimientos.

Todos los movimientos obtenidos son retenidos para un uso futuro. Las plantillas son guardadas en formato XML (20) y cargadas al inicio del programa de esta forma sería más fácil agregar nuevas plantillas de movimiento.

1.8 Motores gráficos.

A continuación se hará una revisión de los motores gráficos más empleados y que están presentes en el mercado, algunas de sus características y en qué basan su dibujado.

- ***Ogre (Object-Oriented Graphics Rendering Engine)***

  Es un motor gráfico desarrollado en C++, multiplataforma, libre y diseñado para hacer más fácil e intuitiva la creación de juegos y aplicaciones de RV, haciendo uso de las capacidades 3D presentes dentro del *hardware*. Presenta una biblioteca de clases basada en *Direct3D* y *OpenGL* que contiene un conjunto de clases que permiten la generación de mundos virtuales. Permite que las escenas se dividan en “páginas” (pedazos de un mapa muy grande se procesan uno a la vez, permitiendo cargar escenas muy grandes). Cada página puede tener sus propios *HeightMaps*. Su renderizado está basado en Octree, BSP (*Binary Space Partition*), y portales. (21)

- ***Crystal Space***

  Es una biblioteca de clases multiplataforma de desarrollo de software para gráficos 3D en tiempo real, con especial atención a los juegos. Escrita en C++ y basada en *OpenGL*. Su dibujado lo realiza mediante portales-sectores (22).

- ***Fly 3D***

  Está basado en renderizado por árboles BSP, PVS (*Potentially Visible Sets*) y portales. Entre sus características se destaca la escalabilidad proporcionada por su sistema de *plugins*. La plataforma que soporta es *Windows*. Licencia libre sin coste y acceso a todo el código fuente. (23)

- ***Genesis3D***

Basado en renderizado por portales, BSP y radiosidad. La única plataforma que soporta es *Windows*. Bajo licencia libre y derecho a modificar el código fuente, a cambio de tener que mostrar el logotipo del motor en las aplicaciones que lo utilicen o bien bajo licencia comercial de costo USD 10 000 por título, para que no salga el logotipo de génesis 3D y código abierto. (23)

- ***3D-GameStudio***

  Es un *kit* de desarrollo para la realización de juegos de ordenador. Provee de un motor 3D, un motor 2D, un editor de niveles y modelos, compilador de *scripts* y bibliotecas de modelos, texturas, etc. Manipula con igual rendimiento escenas de interior y de exterior. Visibilidad por árboles BSP, portales y PVS.

  El principal objetivo que esta aplicación persigue es que el creador del juego no necesite ser un programador experimentado. Ofrecen tres posibilidades para crear un juego: juegos diseñados a base únicamente de ratón, juegos o efectos diseñados con algo de programación utilizando *C-scripts* y juegos o efectos programados en C++ o *Delphi*. (24)

Conclusiones del capítulo.

En este capítulo se hace un estudio sobre el objeto de la investigación, enunciando los conceptos que están relacionados con el procesamiento de imágenes digitales y la captura de movimientos. Se expusieron las bibliotecas que permiten el procesamiento de imágenes digitales y algunas formas de capturar los movimientos de objetos.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

Introducción.

Para alcanzar la solución técnica propuesta se estudiaron de forma exhaustiva las técnicas y los métodos que se abordaron en el primer capítulo. De éstos se seleccionaron las que más se adecuaban para resolver el problema planteado.

2.1 Metodología y herramientas de desarrollo.

Las metodologías (25) imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería.

La metodología es un proceso que define quien debe hacer qué, cómo y cuándo debe hacerlo.

2.1.1 Metodologías de desarrollo de software.

Metodología eXtreme Programming (XP)

XP es una metodología creada por Kent Beck, se caracteriza por retomar las prácticas de uso tradicional y llevarlas al extremo de ahí proviene su nombre. En busca de mejoras en el desarrollo de software y con bases fundamentadas de ingeniería de software, esta metodología detalla varios valores de los métodos ágiles. Se basa en la **realimentación** continua entre el cliente y el equipo de desarrollo, la **comunicación** fluida entre todos los participantes, la **simplicidad** en las soluciones implementadas y el **coraje** para enfrentar los cambios.

Básicamente, la metodología XP se basa en los siguientes valores:

- **Comunicación:** La comunicación permanente es fundamental en XP. Dado que la documentación es escasa, el diálogo frontal, cara a cara, entre desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.

- **Simplicidad:** La sencillez es esencial para que todos puedan entender el código, y se trata de mejorar mediante recodificaciones continuas.

- **Feedback:** Básicamente el continuo contacto con el usuario, al irle entregando las sucesivas versiones, en funcionamiento del producto, permite que este nos dé su valoración y nos comunique, cada vez mejor, lo que realmente quiere en el producto.

- **Coraje:** Básicamente es trabajar muy duro durante las horas dedicadas a ello.

2.1.2 Herramientas de desarrollo.

Microsoft Visual Studio.NET 2008: Como IDE (entorno integrado de desarrollo), se utilizó por las facilidades que brinda. Permite desarrollar aplicaciones utilizando las últimas tecnologías y administrar el ciclo de vida de estas. Este IDE se puede configurar de forma sencilla para la utilización de bibliotecas externas. Brinda soporte para la utilización de varios lenguajes de programación, entre los cuales se encuentran: VB.Net, C #, C, C++.

Visual Paradigm: Esta herramienta CASE se utilizó para el modelado de la aplicación, por el alto nivel de desempeño que se obtiene al diseñar y modelar aplicaciones en ella. Presenta una interfaz que permite modelar las aplicaciones de una forma sencilla y constituye una potente herramienta para el apoyo a la ingeniería de software. Soporta los 13 tipos de diagramas UML que existen actualmente y proporciona ingeniería inversa para diversos lenguajes de programación, entre los cuales se encuentra C++ que fue el que se utilizó en el desarrollo de la aplicación.

2.1.3 Lenguajes

Lenguaje de programación.

El lenguaje de programación escogido fue el C++ ya que constituye uno de los leguajes más robustos y más usados a nivel mundial para aplicaciones de RV, es un lenguaje orientado a objetos y permite un gran control sobre la memoria que se emplea en el programa, siendo esto muy importante para aplicaciones que manejan un nivel de memoria variable.

Lenguaje de modelado.

Se escogió como lenguaje de modelado para la realización del análisis y diseño el UML (Lenguaje Unificado de Modelado), ya que constituye un estándar mundial para el modelado de aplicaciones, además de que se usa para modelar sistemas que usan tecnología orientada a objetos y visualiza, especifica, construye y documenta los artefactos que se crean durante el proceso de desarrollo.

2.2 Información que se maneja.

Para el desarrollo de la aplicación se hace énfasis en los temas referentes a los métodos y funciones que permiten el reconocimiento de los gestos de la mano, partiendo del análisis de una imagen digital.

Después de realizar un análisis de las más importantes bibliotecas de tratamiento de imágenes existentes se decidió utilizar OpenCV por todas las facilidades que brindaba. Esta biblioteca es totalmente libre, desarrollada bajo la licencia BSD. Cuenta con tratamiento de imágenes en tiempo real y tiene implementada funciones para la detección de patrones previamente definidos.

La técnica que se empleó es una selección de varias particularidades que presentan las técnicas de reconocimiento de gestos de la mano vistas en el capítulo 1. El artículo (17) fue tomado como base para la lógica que se utilizó en el sistema, por la forma en que se realiza la extracción de las características de las imágenes procesadas. En este paso son extraídas las características estáticas y dinámicas de la imagen para luego ser analizadas por el reconocedor de gestos. Se realizó un análisis de las restricciones para los usuarios que están presentes en este artículo, por la importancia que estas representan en la optimización y buen funcionamiento del sistema.

En el artículo (16), se proponen una serie de características que se deben tener en cuenta a la hora de confeccionar un patrón determinado. En este artículo se define un set que contiene una serie de objetos que hacen resaltar las propiedades del patrón que se desea construir. Esto resultó de gran ayuda a la hora de elaborar el patrón que se utilizaría posteriormente en la aplicación, logrando de esta forma un patrón de buena calidad.

El método para la segmentación de los gestos de la mano que es utilizado en el artículo (19) se basa en la inyección de colores saturados (HS) en las imágenes para separar los objetos que aparecen en ella. Este método fue utilizado para aislar el patrón de los demás elementos de la escena analizada, esto permitió la identificar el patrón deseado de forma más eficiente.

Para entender el funcionamiento de la biblioteca OpenCV fue necesario analizar en profundidad el artículo (20) en el cual se aborda el tema referente al reconocimiento de gestos soportados en OpenCV. Esta biblioteca brinda un gran soporte para el reconocimiento estático de los gestos de la mano, el cual sirvió de gran ayuda para obtener la posición del patrón detectado. En el método propuesto en este artículo se trabajan con dos cámaras por lo que les es más factible la utilización de plantillas de movimiento, para detectar las direcciones en las que se realiza el mismo. Estas plantillas de movimiento al igual que las de gestos son previamente elaboradas y se utilizan para compararlas con los movimientos detectados por las

cámaras. Los movimientos obtenidos pueden ser guardados en un fichero XML para luego ser utilizados, de esta forma se pueden agregar nuevas plantillas de movimiento.

2.3 Propuesta del sistema.

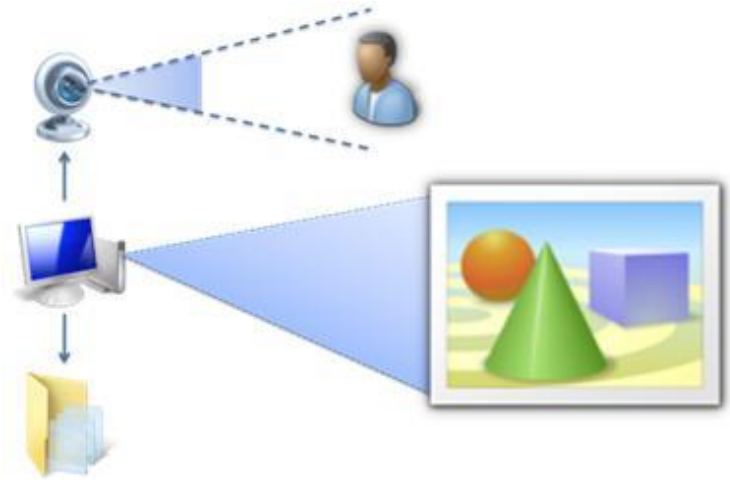

Figura 17: *Composición del sistema*

2.3.1 Captura de información.

El sistema está compuesto por una cámara que debe estar situada en frente del usuario a una distancia de 40cm – 100cm de forma tal que se pueda capturar la parte superior del mismo, ésta se encarga de capturar las imágenes que van a ser analizadas en el CPU. Mantiene un flujo de información constante hacia el CPU, que contiene todos los datos que son capturados.

2.3.2 Procesamiento de los datos capturados.

El flujo de datos que se obtiene desde el dispositivo de captura contiene información innecesaria que afecta la detección del patrón. Estas imágenes necesitan ser modificadas para optimizar el proceso de detección, por tal motivo se decidió aplicar una serie de transformaciones.

Lo primero que se hace es reducir el tamaño a una cuarta parte, en caso de que el dispositivo de captura tenga una resolución mayor que la de 640 x 480 píxeles, la imagen es reducida hasta su octava parte. Con esto se logra que el área a escanear sea más pequeña que la inicial, optimizando en gran medida los cálculos realizados por el algoritmo de detección del patrón.

El método utilizado para escalar la imagen es *Gaussian Pyramid* (Pirámides Gaussianas) (15) que reduce el tamaño construyendo la pirámide Gaussiana. Para producir una capa de la pirámide (G_{i+1}), partiendo

de la anterior (G_i), se transforma (G_i) con un núcleo Gaussiano (15) y se remueve cada columna y fila par. Repitiendo este proceso en la imagen de entrada (G_0) se produce la pirámide completa.

Luego de reducir el tamaño de la imagen es necesario aplicar un filtro para eliminar los datos que no se utilizan de la imagen. En este caso se aplicó un filtro de tipo *Gaussian Smooth* (Desenfoque Gaussiano) que se encarga de remover las líneas que menos resaltan en la imagen y que interfieren en el proceso de detección del patrón. Esto se logra deformando cada píxel de la imagen de entrada aplicando un filtro promediador Gaussiano con un tamaño de máscara 5x5, que es el único tamaño permitido por la función utilizada.

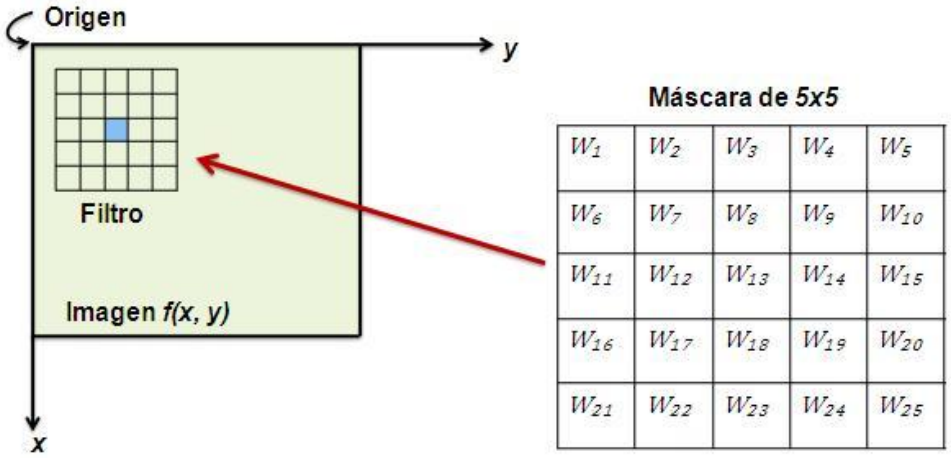

Figura 18: *Filtro promediador*

La máscara de filtrado o convolución utilizada es aplicada a cada píxel, este es procesado teniendo en cuenta los valores de los coeficientes de la vecindad.

Figura 19: *Imagen inicial*

Figura 20: *Imagen transformada*

Esta transformación resalta los contornos de los objetos dentro de las imágenes, quedando solamente en ellas las líneas que se encuentran bien definidas, las líneas pequeñas y finas son eliminadas por el filtro.

2.3.3 Proceso de detección del patrón.

Una vez que se transforma la imagen y se eliminan los datos innecesarios, se procede a buscar dentro de la misma el patrón que se desea reconocer. Este proceso involucra gran cantidad de cálculos por parte del procesador y para evitar que interfiriera con el resto de la aplicación se creó un hilo de ejecución aparte solo para este proceso. El algoritmo utilizado busca el patrón dentro de la imagen en todas las escalas y para optimizar este proceso se definió un tamaño inicial, evitando que se busque en zonas demasiado pequeñas como para que pueda existir el patrón en ellas.

Carga del patrón a detectar.

Antes de comenzar a escanear la imagen, es necesario cargar el patrón, este se encuentra guardado en un fichero de tipo XML que contiene todas las características Haar del patrón. Esta medida Haar es una manera de asignar un "volumen invariante" a los subconjuntos de grupos topológicos localmente compactos. Al generarse, se almacena en forma de cascada de rechazo, de forma tal que al ir recorriéndola podamos ir descartando las secciones de la imagen que no contienen el patrón.

Los nodos que componen esta cascada de rechazo fungen como filtros que, empleando cierta característica Haar del patrón, determinan si el segmento de imagen que se analiza no puede hacerse coincidir con este, de lo contrario se pasa al nodo siguiente. Estos nodos están ordenados de menor a mayor complejidad, minimizando la cantidad de cálculos que se realizan cuando se procesan partes simples de la imagen. Si se logra recorrer la cascada completa entonces se asume que el patrón está contenido en la muestra analizada.

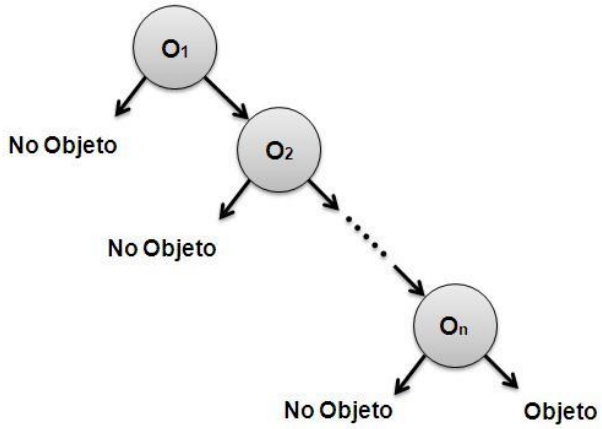

Figura 21: *Cascada de rechazo*

El fichero es cargado por una función que mantiene en memoria un clasificador de características Haar para ser utilizadas por el algoritmo de detección.

Escaneando la imagen.

Este proceso recorre toda la imagen seleccionando segmentos de ella y buscando el patrón en ellos. Los segmentos van desde el tamaño mínimo definido hasta el tamaño máximo de la imagen. El tamaño mínimo puede ser establecido para optimizar el proceso de detección y ahorrar cálculos innecesarios.

Se establece un factor de escala medio, que determina el tamaño de los saltos entre cada sección de la imagen. Este factor puede aumentarse para obtener mayor velocidad de cálculo al costo de no poder detectar objetos de un tamaño específico.

Para que se decida que el patrón se encuentra contenido dentro de una de las secciones de la imagen deben existir un número mínimo de coincidencias en esa sección. Poner este número en un valor muy alto implica un gran costo de procesamiento y si se pone demasiado bajo es posible obtener falsos positivos. Por este motivo se decidió un término medio, para no tener una gran cantidad de cálculo y que el reconocimiento sea lo más preciso posible.

Determinando la posición del patrón detectado.

Para determinar la posición del patrón que se detectó, es necesario encerrarlo en un rectángulo que contenga su envoltura convexa. Envolviendo toda la sección de la imagen donde se encuentra el patrón se garantiza que al calcular el punto medio de este rectángulo estemos obteniendo la posición del centro del patrón detectado.

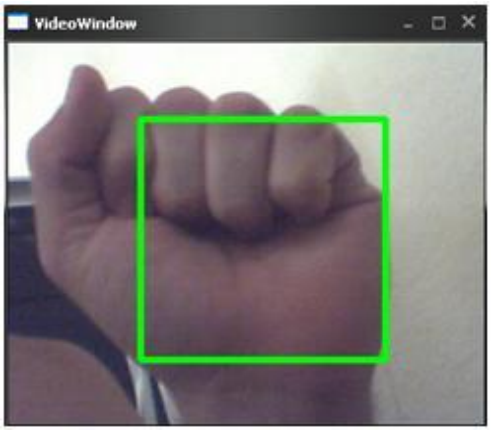

Figura 22: *Rectángulo que contiene la envoltura convexa del patrón*

2.3.4 Selección del objeto en el entorno virtual.

La selección del objeto que se encuentra ubicado en el entorno virtual comienza en el momento en que es detectado el patrón. Para poder empezar con este proceso es necesario convertir la posición en que se encuentra el patrón dentro de la imagen, debido a que las posiciones del entorno virtual y las generadas por el proceso de detección son distintas. Para esto se utilizan las siguientes fórmulas:

Fórmula para calcular la posición en el eje X:

$$\frac{(posX \times anchoEntornoVirtual)}{anchoImagenCapturada} - \frac{anchoEntronoVirtual}{2}$$

Fórmula para calcular la posición en el eje Y:

$$\frac{(posY \times altoEntornoVirtual)}{altoImagenCapturada} + \frac{altoEntronoVirtual}{2}$$

Fórmula para calcular la posición en el eje Z:

Esta posición es la que determina la profundidad a la que se encuentra el patrón dentro del entorno virtual, que es la distancia que existe entre la cámara web y el patrón detectado. Como el sistema funciona con una cámara que no es infrarroja, las cuales son utilizadas por sistemas que permiten estimar distancia, es necesario calcular esta posición y para esto se usa el radio de la circunferencia inscrita en el rectángulo que contiene la envoltura convexa del patrón. Este radio representa el factor de crecimiento de los valores de la posición en el eje Z. A medida que aumenta el radio es que el patrón está más cerca de la cámara, por lo que aumentaría el valor en profundad del objeto que se encuentre seleccionado y viceversa.

Una vez convertidas las posiciones se traza un rayo en profundidad partiendo desde la posición en que se detectó el patrón, el primer elemento con el que el rayo coincida es seleccionado.

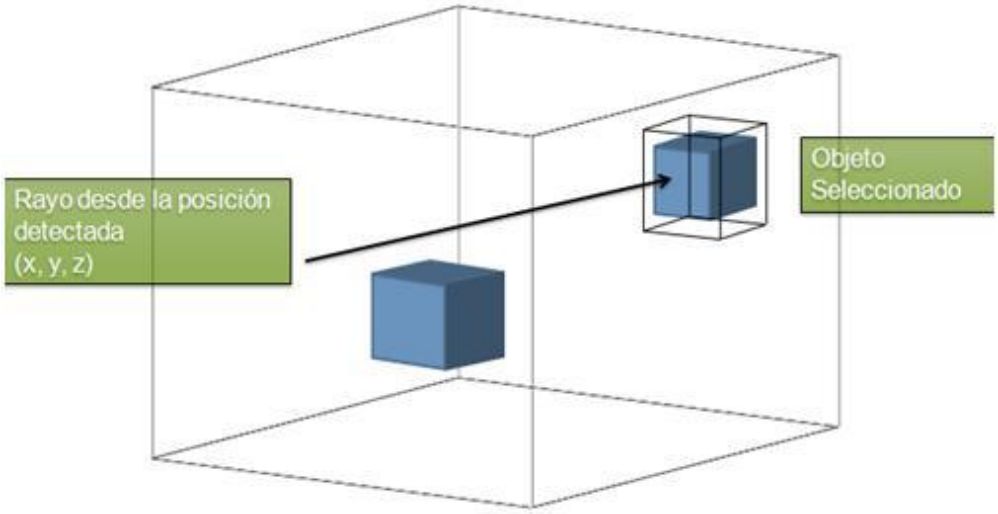

Figura 23: *Objeto seleccionado*

Cuando un objeto es seleccionado sus posiciones se mantienen actualizadas hasta que se deje de detectar el patrón. Al actualizar constantemente las posiciones del objeto que está seleccionado, logramos que al cambiar la posición del patrón el objeto se mueva a consecuencia de esto, dando la sensación de que se tiene “agarrado” el objeto.

2.4 Modelo de dominio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Es un diagrama con los objetos que existen relacionados con el proyecto y las relaciones que hay entre ellos (26). Con el uso de este proceso de modelado se van a definir los procesos y roles relacionado con la obtención de requerimientos y el análisis-diseño. Debido a que no se cuenta con una definición total de los procesos de negocio, se plantea confeccionar un modelo de dominio que servirá como referencia a los posibles usuarios para la comprensión de todos los términos utilizados en la elaboración del sistema y como guía para el futuro diseño de este.

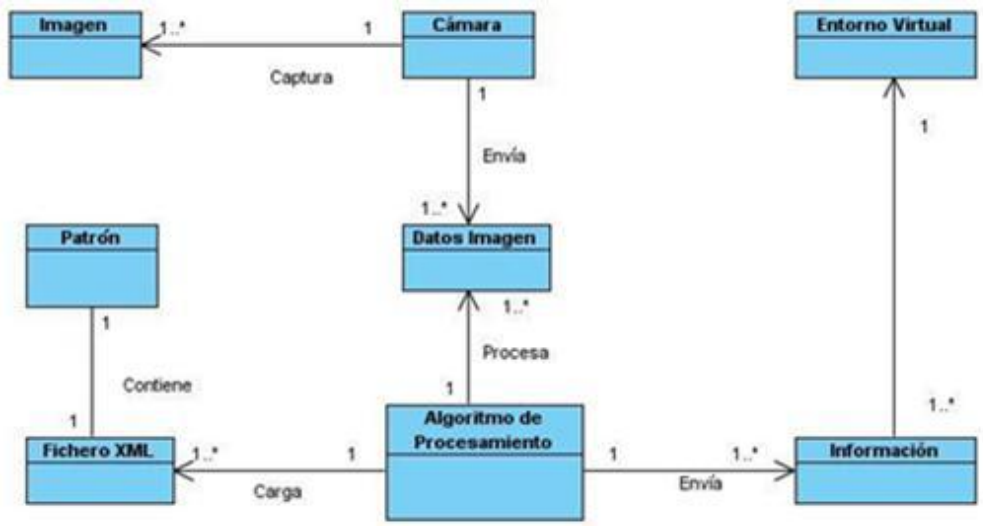

Figura 24: *Modelo de dominio*

2.5 Personal relacionado con el sistema.

Se define como persona relacionada con el sistema a aquella que está de una forma u otra vinculada al proceso de desarrollo de la aplicación, así como las que interactúan con el mismo.

Tabla 3: Personal relacionado con el sistema

Personal relacionado con el sistema	Justificación
Desarrollador	Persona encargada de implementar e integrar las funcionalidades de la aplicación.
Usuario	Persona que va a interactuar con la aplicación.

2.6 Especificaciones de los requisitos de software.

2.6.1 Requisitos del sistema:

Requisitos funcionales:

Teniendo en cuenta las necesidades que existen para la confección del sistema quedan definidos los siguientes requerimientos funcionales:

- Detectar el dispositivo de captura conectado.
- El sistema debe ejecutar en un hilo de procesamiento la visualización del sistema y en otro el proceso de detección.
- El sistema debe obtener flujo de información proveniente del dispositivo de captura.
- Cargar un patrón de un fichero XML.
- Procesar las imágenes provenientes del dispositivo de captura.
- Detectar si el patrón cargado se encuentra en la imagen.
- Obtener la posición del patrón detectado.
- Convertir la posición del patrón detectado en una posición válida para el entorno virtual.
- Seleccionar el objeto que se encuentre en la posición obtenida.
- Mantener el objeto seleccionado en el caso de que se mantenga detectando el patrón.

Requisitos no funcionales:

Hardware:

- Cámara web
- CPU Pentium 4
- 512 RAM
- La aceleración gráfica es proporcional a la aplicación gráfica donde se utilice el sistema.

Software:

- Sistema operativo Window XP / Window 7.
- Controlador de la cámara web

Rendimiento:

- La aplicación funciona en tiempo real.
- La aplicación debe tener un alto nivel de procesamiento o cálculo, y un buen manejo de la memoria.

2.7 Fase de exploración.

Durante esta etapa se define el alcance del proyecto, además los clientes proponen a grandes rasgos las historias de usuarios que son de gran interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, las tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con las tecnologías.

2.7.1 Historia de usuarios.

Las historias de usuario (HU) son utilizadas en la metodología de desarrollo ágil XP para representar una breve descripción del comportamiento del sistema.

Las historias de usuario tienen tres aspectos:

- **Tarjeta:** en ella se almacena suficiente información para identificar y detallar la historia.
- **Conversión:** cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación).
- **Pruebas de aceptación:** permite confirmar que la historia ha sido empleada correctamente.

En la fase de exploración se identificaron las siguientes historias de usuario:

Tabla 4: HU1 - Detectar dispositivo de captura conectado

Historia de usuario	
No. 1	**Nombre:** Detectar el dispositivo de captura conectado.
Usuario: Desarrollador.	

Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 1 semana.	**Iteración asignada:** 1
Descripción: Se encarga de detectar si hay una cámara web conectada.	
Información adicional (Observaciones): La aplicación necesita conocer si existe conectada una cámara web para poder empezar la captura de imágenes.	

Tabla 5: HU2 - Obtener flujo de información del dispositivo de captura conectado

Historia de usuario	
No. 2	**Nombre:** Obtener flujo de información proveniente del dispositivo de captura conectado.
Usuario: Desarrollador.	
Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 2 semanas.	**Iteración asignada:** 1
Descripción: Permite obtener la información proveniente de la cámara web que esté conectada al sistema.	
Información adicional (Observaciones): Para la aplicación es muy importante la obtención continua de información proveniente de la cámara web.	

Tabla 6: HU3 - Cargar un patrón desde un fichero XML

Historia de usuario	
No. 3	**Nombre:** Cargar un patrón desde un fichero XML.

Usuario: Desarrollador.	
Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 1 semana.	**Iteración asignada:** 2
Descripción: Permite cargar el patrón que se va a necesitar en el proceso de detección.	
Información adicional (Observaciones): Para poder empezar la detección de un patrón es necesario cargar los datos del mismo.	

Tabla 7: HU4 - Detección del patrón

Historia de usuario	
No. 4	**Nombre:** Detección del patrón.
Usuario: Desarrollador.	
Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 2 semanas.	**Iteración asignada:** 1
Descripción: En este proceso es donde se aplican los algoritmos de detección para detectar si la imagen procesada contiene el patrón previamente cargado.	
Información adicional (Observaciones): La aplicación necesita localizar el patrón en la imagen para poder ubicar su posición.	

Tabla 8: HU5 - Convertir la posición del patrón detectado

Historia de usuario	
No. 5	**Nombre:** Convertir la posición del patrón detectado en una que

	sea válida para el entorno virtual.
Usuario: Desarrollador.	
Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 3 semanas.	**Iteración asignada:** 2
Descripción: Para poder interactuar con el entorno virtual es necesario convertir las posiciones en donde se encuentra el patrón detectado.	
Información adicional (Observaciones): Este paso es de elevada importancia para poder interactuar con los objetos en el mundo virtual.	

Tabla 9: HU6 - Selección del objeto

Historia de usuario	
No. 6	**Nombre:** Selección del objeto que se encuentre en la posición obtenida.
Usuario: Desarrollador.	
Prioridad en el negocio: Alta.	**Nivel de complejidad:** Alta.
Tiempo de estimación: 2 semanas.	**Iteración asignada:** 3
Descripción: En este paso es donde se selecciona un objeto dentro del mundo virtual, aplicando la posición proveniente de la captura del patrón de la mano y previamente convertida.	

Información adicional (Observaciones): Trazando un rayo virtual en profundidad a partir de la posición detectada es posible seleccionar un objeto que se encuentre en su dirección.

2.8 Fase de planeamiento.

2.8.1 Estimación de esfuerzo por historias de usuario.

Se realizó una estimación del esfuerzo realizado en cada historia de usuario identificadas en la fase de exploración con el propósito de realizar un buen desarrollo del sistema propuesto.

Tabla 10: Estimación de esfuerzo por historias de usuario

Historias de usuario	**Puntos de estimación**
Detectar el dispositivo de captura conectado.	1 semana
Obtener flujo de información proveniente del dispositivo de captura conectado.	2 semanas
Cargar un patrón desde un fichero XML.	1 semana
Detección del patrón.	2 semanas
Convertir la posición del patrón detectado en una que sea válida para el entorno virtual.	3 semanas
Selección del objeto que se encuentre en la posición obtenida.	2 semanas.
Total:	**11 semanas.**

2.8.2 Plan de iteraciones.

Las HU seleccionadas para cada entrega son desarrolladas y probadas en el ciclo de iteración, de acuerdo al orden establecido.

Iteración 1

Esta iteración tiene propósito de darle cumplimiento a las historias de usuario que se consideraron de mayor importancia para el desarrollo de la aplicación. Al finalizar esta iteración se obtuvo una versión de prueba con funcionalidades muy básicas.

Iteración 2

En esta iteración se realizó la implementación de las historias de usuario con **prioridad de negocio** alta y se corrigieron los errores encontrados en la iteración anterior. Al final de esta iteración se le mostró al cliente la segunda versión obtenida, con las nuevas funcionalidades implementadas para ver la aceptación del mismo, y saber si era necesario realizar algún cambio.

Iteración 3

En esta iteración se implementan las restantes historias de usuarios para la obtención del producto final. Después de una revisión de lo realizado en las anteriores iteraciones se obtuvo la versión 1.0 del sistema.

2.8.3 Plan de duración de las iteraciones.

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo de software XP, se crea un plan de duración de cada una de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las iteraciones.

Tabla 11: Plan de duración de las iteraciones

Iteración	**Historias de usuarios**	**Duración total de las iteraciones**
Iteración 1	Detectar el dispositivo de captura conectado.	5 semanas
	Obtener flujo de información proveniente del dispositivo de captura conectado.	
	Detección del patrón.	
Iteración 2	Cargar un patrón desde un fichero XML.	4 semanas

	Convertir la posición del patrón detectado en una que sea válida para el entorno virtual.	
Iteración 3	Selección del objeto que se encuentre en la posición obtenida.	2 semanas

CAPÍTULO 3: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción:

En el presente capítulo se abordarán las tres iteraciones que se concibieron durante la fase de planeamiento, exponiéndose las tareas generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

3.1 Diseño de la solución propuesta.

La Metodología de Desarrollo XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. La misma se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Para el diseño de las aplicaciones, XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación entre el equipo de desarrollo, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

3.1.1 Tarjetas CRC.

Para poder diseñar el sistema como un equipo, se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Debido a la facilidad de uso y entendimiento, que propician dichas tarjetas se decidió utilizarlas para diseñar la aplicación que se desea desarrollar.

Tabla 12: Tarjeta CRC OgreListener

Tarjeta CRC
Clase: OgreListener

Súper clase: ExampleFrameListener	
Sub clase(s): -	
Responsabilidades: • Trazar un rayo en la posición en que se detecto el patrón. • Capturar el objeto que se encuentra en la posición del rayo trazado. • Mover el cursor hacia la posición en que se encuentra el patrón. • Mantener actualizada la posición del objeto seleccionado y la del cursor.	**Colaboraciones:** Clase HandCaster Clase MouseCursor

Tabla 13: Tarjeta CRC HandCaster

Tarjeta CRC	
Clase: HandCaster	
Súper clase: Thread	
Sub clase(s): -	
Responsabilidades: • Conectar una cámara. • Desconectar una cámara.	**Colaboraciones:** Clase CvCapture Clase IplImage

• Iniciar un hilo de ejecución. • Detener un hilo de ejecución. • Cargar un patrón desde un fichero XML. • Analizar los datos provenientes de la cámara conectada. • Detectar los objetos que correspondan con el patrón previamente cargado. • Devolver la posición del objeto detectado. • Devolver el radio de la circunferencia circunscrita al objeto detectado para estimar la distancia del mismo a la cámara.	Clase CvHaarClassifierCascade

Tabla 14: Tarjeta CRC Thread

Tarjeta CRC	
Clase: Thread	
Súper clase: -	
Sub clase(s): -	
Responsabilidades:	**Colaboraciones: -**

• Inicializar un hilo de ejecución. • Cancelar o matar un hilo que se esté ejecutando.	

Tabla 15: Tarjeta CRC CvCapture

Tarjeta CRC	
Clase: CvCapture	
Súper clase: -	
Sub clase(s): -	
Responsabilidades: • Contiene toda la información del estado de la captura realizada.	**Colaboraciones: -**

Tabla 16: Tarjeta CRC IplImage

Tarjeta CRC	
Clase: IplImage	
Súper clase: -	
Sub clase(s): -	
Responsabilidades: • Esta estructura en si es un objeto de CvMat pero mejorado, de forma	**Colaboraciones:**

tal que la matriz pueda ser interpretada como una imagen. • Obtiene el origen en que se encuentra posicionada la imagen. • Calcula las dimensiones de la imagen capturada. • Detectar los canales de la imagen. • Almacena la profundidad de la imagen. • Contiene una región de interés, la cual es utilizada por los algoritmos de procesamiento de imágenes.	

Tabla 17: Tarjeta CRC CvHaarClassifierCascade

Tarjeta CRC	
Clase: CvHaarClassifierCascade	
Súper clase: -	
Sub clase(s): -	
Responsabilidades: • Contiene las características del Haar (27), que son cargadas desde un fichero de tipo XML.	**Colaboraciones:**

3.1.2 Diagrama de componentes.

Un diagrama de componentes muestra las dependencias lógicas entre componentes de software, sean éstos componentes fuentes, binarios o ejecutables, que ilustran las piezas del software, controladores embebidos, etc. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema, es decir, para describir la vista de implementación estática de un sistema. Los diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones pero un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución

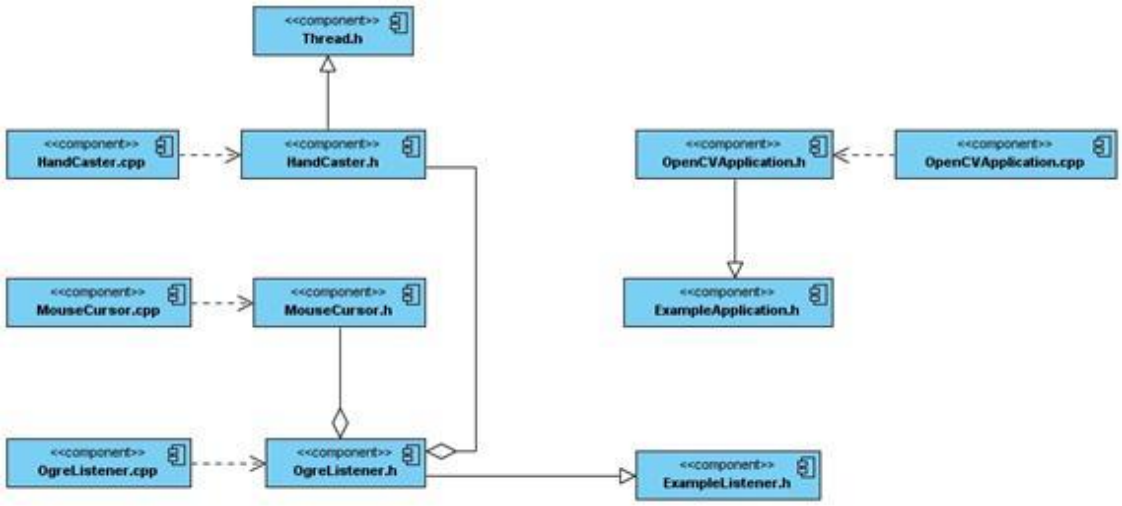

Figura 25: *Diagrama de componentes*

3.1.3 Patrones de diseño.

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Existen problemas durante el desarrollo de software que se repiten o que son análogos, que responden a cierto patrón. Durante el desarrollo del sistema de interacción natural se utilizaron un conjunto de patrones entre los que se destacan:

- **Encapsulamiento (ocultación de datos):** Propone el ocultamiento de los datos miembro de un objeto, de manera que solo se puede cambiar mediante las operaciones definidas para ese objeto.

- **Subclase:** Este patrón propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta a través de resoluciones sobre que implementación debe ser ejecutada.

3.1.4 Estándar de codificación.

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Para la implementación del sistema se decidió utilizar las convecciones para la programación en el lenguaje C++, por la flexibilidad que este brinda.

Algunas pautas que establece este estilo de código:

Indentación.

- 4 espacios para la indentación.
- Un espacio después de los siguientes *tokens*: paréntesis, comas, llaves, corchetes.

Declaración.

Variables.

- Se utilizaron nombres significativos, para que su simple lectura pueda identificar su función dentro de la aplicación, sin tener que consultar manuales o hacer demasiados comentarios.
- Para nombres que se usen con frecuencia o para términos largos, se utilizaron abreviaturas para que éstos tuvieran una longitud razonable.
- Se evitaron identificadores que comiencen con uno o dos caracteres de subrayado para evitar que se confundan con los que el compilador selecciona.
- Para los nombres que contengan más de un término, el primero debe empezar con minúscula y los siguientes con mayúscula.

```
int radius;

IplImage*  frameCaptured;
```

Clases.

- Siempre comenzar con mayúscula y la segunda palabra también debe comenzar con mayúscula.
- El nombre debe ser significativo y de fácil comprensión, para que pueda conocerse su principal función sin la necesidad de consultar alguna ayuda.

  class HandCaster{};.

Funciones.

- Siempre comenzar con la primera palabra en minúscula, la cual debe indicar la acción que realiza la función.

  CvHaarClassifierCascade* loadObjectDetector(const char* cascadePath);

3.2 Desarrollo de las iteraciones.

En la fase de planificación se detallaron los HU correspondientes con cada una de las iteraciones a desarrollar, teniendo en cuenta las necesidades requeridas por el cliente. Durante el transcurso de las iteraciones se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de que sea necesario. Como parte de este plan, se descomponen las HU en tareas de programación o de ingeniería, asignando a un equipo de desarrollo (o una persona), responsable de su implementación, aplicando la práctica de la programación en parejas. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

Teniendo en cuenta la planificación realizada con anterioridad, se llevó a cabo el desarrollo del sistema en tres iteraciones, obteniéndose como finalidad un producto con todas las restricciones y características deseadas por el cliente. A continuación se detallan cada una de las iteraciones.

3.2.1 Iteración 1.

En esta iteración se da cumplimiento a la implementación de las HU que corresponden con los números 1, 2 y 4 consideradas de mayor importancia para el desarrollo de la aplicación, con el fin de obtener una versión del producto con algunas funcionalidades criticas como: Conectar el dispositivo, habilitar la captura de los componentes del mismo.

Tabla 18: HU abordadas en la primera iteración

Historias de usuario	**Tiempo de implementación(semanas)**	
	Estimación	**Real**
Detectar el dispositivo de captura conectado.	1	1
Obtener flujo de información proveniente del dispositivo de captura conectado.	2	1
Detección del patrón.	2	4

Tabla 19: Tarea de la HU1

Tarea de ingeniería.	
No. De la tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Detectar el dispositivo de captura conectado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 8/02/2011	**Fecha fin:** 16/02/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	
Descripción: Esta funcionalidad es de vital importancia para el funcionamiento del sistema, debido a que si no se encuentra ningún dispositivo de captura, en este caso sería una cámara web, el sistema no podría funcionar.	

Tabla 20: Tarea de la HU2

Tarea de Ingeniería.

No. De la tarea: 2	**No. De la HU:** 2
Nombre de la tarea: Obtener flujo de información proveniente del dispositivo de captura conectado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha inicio: 17/02/2011	**Fecha fin:** 28/02/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	
Descripción: En esta funcionalidad se procesa toda la información proveniente del dispositivo de captura conectado, para luego ser tratada. Es muy importante para poder realizar las siguientes tareas.	

Tabla 21: Tarea de la HU4

Tarea de ingeniería.	
No. De la tarea: 3	**No. De la HU:** 4
Nombre de la tarea: Detección del patrón.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha inicio: 2/03/2011	**Fecha fin:** 19/03/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	

Descripción: Esta es la funcionalidad principal del sistema. La misma se encarga de analizar el flujo de información proveniente desde el dispositivo de captura y buscar en cada imagen obtenida el patrón que se desea detectar.

3.2.2 Iteración 2.

En esta iteración se realizó la implementación de las historias de usuario 3 y 5. Estas historias de usuario brindan la funcionalidad de poder cargar un patrón definido y de convertir la posición en que se detectó el patrón, de forma que sea válida para el entorno virtual.

Tabla 22: HU abordadas en la segunda iteración.

Historias de usuario	**Tiempo de implementación(semanas)**	
	Estimación	**Real**
Cargar un patrón desde un fichero XML.	1	1
Convertir la posición del patrón detectado en una que sea válida para el entorno virtual.	3	2

Tabla 23: Tarea de la HU

Tarea de ingeniería.	
No. De la tarea: 4	**No. De la HU:** 3
Nombre de la tarea: Cargar un patrón desde un fichero XML.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 20/03/2011	**Fecha fin:** 27/03/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	

Descripción: Esta funcionalidad le permite al sistema cargar un patrón para ser utilizado en la detección de objetos.

Tabla 24: Tarea de la HU5

Tarea de ingeniería.	
No. De la tarea: 5	**No. De la HU:** 5
Nombre de la tarea: Convertir la posición del patrón detectado en una que sea válida para el entorno virtual.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 3
Fecha inicio: 28/03/2011	**Fecha fin:** 18/04/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	
Descripción: Esta funcionalidad convierte la posición obtenida al detectar el patrón en las imágenes provenientes del dispositivo de captura. Las primeras posiciones hacen referencia al lugar en que se encuentra el patrón dentro de un IplImage, estas tienen que ser convertidas para que sean válidas en un mundo virtual.	

3.2.3 Iteración 3.

En esta última iteración se implementó la historia de usuario 6, la cual desempeña la funcionalidad de seleccionar el objeto que se encuentre en la posición en la que fue detectado el patrón.

Tabla 25: HU abordada en la tercera iteración.

Historias de usuario	**Tiempo de implementación(semanas)**	
	Estimación	**Real**
Selección del objeto que se encuentre en la posición obtenida.	2	3

Tabla 26: Tarea de la HU6.

Tarea de ingeniería.	
No. De la tarea: 6	**No. De la HU:** 6
Nombre de la tarea: Selección del objeto que se encuentre en la posición obtenida.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2
Fecha inicio: 20/04/2011	**Fecha fin:** 9/05/2011
Programador responsable: Alberto Yusniel Rivero Nuñez.	
Descripción: Esta funcionalidad permite seleccionar el objeto que se encuentre en la posición obtenida luego de que el patrón buscado ha sido detectado.	

3.3 Pruebas.

Unos de los pilares de la metodología XP es el uso de las pruebas para componer el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar los efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que el final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente, siendo este el indicado para diseñar este tipo de pruebas.

3.3.1 Desarrollo dirigido por pruebas.

El desarrollo dirigido por pruebas (TDD), es una práctica de programación que involucra otras prácticas como: Escribir las pruebas primero (*Test First Development*) y Refactorización (*Refactoring*). Para escribir las pruebas generalmente se utiliza la Prueba Unitaria (*Unit Test*).

Principalmente se escribe una prueba y se verifica que las pruebas fallen, luego se implementa el código que haga que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del proceso guiado por pruebas es lograr un código limpio que funcione. La idea es que los requerimientos sean traducidos a pruebas, de este modo cuando las pruebas pasen se garantizará que los requerimientos se hayan implementado satisfactoriamente.

La TDD permite un diseño robusto, tanto es así que a menudo se piensa TDD como diseño manejado por las pruebas (*Test Driven Design*). En consecuencia, TDD facilita un diseño sustentable a través de la noción de pruebas. Estas pruebas obligan a reflexionar sobre el comportamiento del código y la forma de garantizar que funcione según lo previsto. La mayoría de las veces, el código influenciado por TDD es relativamente seguro y sin dudas bastante simple.

3.4 Pruebas de aceptación.

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Tabla 27: Prueba de aceptación para la HU1

Caso de prueba de aceptación.	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Comprobar la detección del dispositivo de captura conectado.	
Descripción: Evaluar que el sistema detecte el dispositivo.	
Condiciones de ejecución: El usuario debe comprobar que el dispositivo de captura está conectado e instalado correctamente en la computadora.	
Entrada/Pasos de ejecución: Confirmar que la aplicación se ejecute correctamente.	
Resultado esperado: El dispositivo de captura es detectado correctamente.	

Evaluación de la prueba: Bien.

Tabla 28: Prueba de aceptación para la HU2

Caso de prueba de aceptación.	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Evaluar la obtención de información proveniente del dispositivo de captura conectado.	
Descripción: Verificar que el sistema se mantenga recibiendo información del dispositivo de captura conectado.	
Condiciones de ejecución: El cliente debe asegurarse que el dispositivo de captura permanezca conectado al sistema.	
Entrada/Pasos de ejecución: Asegurarse de que la aplicación se sigue ejecutando.	
Resultado esperado: El sistema obtiene información constante del dispositivo de captura.	
Evaluación de la prueba: Bien.	

Tabla 29: Prueba de aceptación para la HU4

Caso de prueba de aceptación.	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Evaluar la detección del patrón.	
Descripción: Verificar que el sistema detecte el patrón.	

Condiciones de ejecución: El cliente debe asegurarse que el dispositivo de captura permanezca enfocado en dirección al patrón que se desea reconocer.
Entrada/Pasos de ejecución: Probar que el cursor de la aplicación se mueve.
Resultado esperado: El sistema detecta el patrón.
Evaluación de la prueba: Bien.

Tabla 30: Prueba de aceptación para la HU6

Caso de prueba de aceptación.	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Comprobar la selección de objetos.	
Descripción: Verificar que el sistema seleccione objetos en el entorno virtual.	
Condiciones de ejecución: El cliente debe asegurarse que el cursor se encuentre encima del objeto que se desea seleccionar.	
Entrada/Pasos de ejecución: Probar que al detectar el patrón sobre un objeto, este es seleccionado.	
Resultado esperado: El sistema selecciona el objeto.	
Evaluación de la prueba: Bien.	

Conclusiones del capítulo.

En este capítulo se elaboró el diagrama de componentes del sistema, para lograr una visión detallada de los elementos que lo componen. Se construyeron las tarjetas CRC, las cuales proporcionan un alto nivel entendimiento del sistema propuesto y se definió el estándar de codificación que se utilizó en la implementación del sistema. Para darle solución a las historias de usuarios planteadas se desarrollaron las tareas correspondientes en cada una de las iteraciones realizadas y quedó definido el modelo arquitectónico a emplear. Se realizaron las pruebas unitarias y se definieron las pruebas de aceptación. Con la finalización de este capítulo se da por terminada la propuesta de solución de la aplicación a implementar.

CONCLUSIONES

Dando cumplimiento al objetivo de este proyecto, y a los requisitos planteados, se obtuvo una aplicación para la interacción con los sistemas de RV, que permite seleccionar de manera natural los objetos que existen dentro de estos entornos. Estrechando la relación que existe entre las personas y las computadoras.

RECOMENDACIONES

Con el objetivo de mejorar el sistema elaborado se propone realizar un proceso de calibración para optimizar el rendimiento del sistema. Así como añadir nuevos patrones que hagan posible no solo la traslación de los objetos, si no otras acciones como la de rotación y escalado de estos. Logrando extender el alcance del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. **ABC.** Definición ABC. *Definición ABC.* [En línea] 4 de Septiembre de 2009. http://www.definicionabc.com/tecnologia/realidad-virtual.php.

2. **Domingo, Mery.** Vision por Computador. *Imagenes Digitales y Aplicaciones.* Chile : Universidad Catolica de Chile, 2004.

3. **Alfaomega.** *Vision por Computador.* Chile : Universidad Catolica de Chile, 2002.

4. **Louka, Michael.** *An Introduction to Virtual Reality.* 1997.

5. **Isdale, Jerry.** *What Is Virtual Reality.* 1998.

6. **Woods, R. C. Gonzalez and R. E.** *Digital ImageProcessing.* s.l. : Addison-Wesley, 1992.

7. **Shneiderman, Ben.** Chapter 9: Interaction Devices. *Designing the User Interface.*

8. **Suzuki and Kenichi, satochi.** *Topological structural analysis of digitized binary images by border following.* s.l. : Computer Vision, Graphics, and Image Processing, 1985.

9. **Cesar, A.** *Visión Artificial para la Detección y Ubicación Espacial.* Caracas : Universiadad Central de Venezuela, 2008.

10. Virtual World Software. *Virtual World Software.* [En línea] 21 de 5 de 2008. [Citado el: 12 de 1 de 2011.] http://www.hondumall.com/virtual-world/index.html.

11. **Corporation, Intel.** *Intel Imagen Processing Library, Reference Manual.*

12. **Andy Martignoni, Brian P. Gerkey, Brendan Burns, Ben Grocholsky.** http://www.cs.cmu.edu/~jbruce/cmvision/. [En línea] [Citado el: 18 de 2 de 2011.]

13. **Thomas J. Olson, Nicholas G. Klop, Mark R. Hyett and Shawn M. Carnell.** *MAVIS: A Visual Environment for Active Computer Vision.*

14. **McLauchlan, Philip F.** http://gandalf-library.sourceforge.net/. [En línea] [Citado el: 24 de 3 de 2011.]

15. **Kaehler, Gary Bradski and Adrian.** *Learning OpenCV.* 2008.

16. *VISUAL-BASED INTERFACE USING HAND GESTURE.* **SAFAE, A. CHALECHALE AND F.** Iran : Printed in The Islamic Republic of Iran., 2008, Vols. Iranian Journal of Science & Technology Vol. 32, No. B3, pp 279-293.

17. **S.M. Hassan Ahmeda, Todd C. Alexanderb, and Georgios C. Anagnostopoulos.** *Real-time, Static and Dynamic Hand Gesture Recognition for Human-Computer Interaction.* Miami : s.n., 2008.

18. **E. Blanco, M. Mazo, L. M. Bergasa, S. Palazuelos and A.B. Awawdeh.** Improvement of the Segmentation in HS Sub-space by means of a Linear Transformation in RGB Space. [aut. libro] T. Sobh. *Innovative Algorithms and Techniques in Automation, Indostrial Electronics and Telecommunications.* 2007.

19. **Ana Beatriz Alvarez, José Raimundo de Olivera.** *Hand Gesture Recognition using a Lattice Autoassociative Memory.* Campinas, Sao Paulo, Brazil : Departament of Computer Engineering and Industrial Automation School of Electrical and Computer Engineering, University of Campinas .

20. **Ondřej Novák, František Ondřej, Jan Ondřej.** *Gesture recognition.* 2006.

21. Ogre3D. *http://www.ogre3d.org.* [En línea] [Citado el: 15 de 1 de 2011.] http://www.ogre3d.org..

22. **http://www.crystalspace3d.org.** http://www.crystalspace3d.org. *http://www.crystalspace3d.org.* [En línea] [Citado el: 18 de 2 de 2011.] http://www.crystalspace3d.org/main/Main_Page.

23. **Catá, Jordi.** *Comparativa de motores gráficos para videojuegos.* 2002.

24. **Camacho Román, Yanoski Rogelio y Jiménez López, Fernando.** *Biblioteca gráfica para Sistemas de RV.* La Habana : s.n., 2004.

25. **Erly, Expósito Delgado.** http://www.monografias.com. [En línea] [Citado el: 6 de 2 de 2011.] http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml .

26. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software Volumen I.* La Habana : Félix Varela, 2004.

27. domotica.net. *domotica.net.* [En línea] [Citado el: 16 de 4 de 2011.] http://pt.domotica.net/Medida_de_Haar.

28. http://www.crystalspace3d.org. [En línea] [Citado el: 17 de 1 de 2011.] http://www.crystalspace3d.org.

29. **Rodríguez, Hugo.** Iniciación a la Imagen Digital. [En línea] [Citado el: 16 de 3 de 2011.] http://www.hugorodriguez.com/cursos/curso-idigital_01.htm.

30. **www.innovanet.com.** http://www.innovanet.com.ar/gis/TELEDETE/TELEDETE/tradiimg.htm. [En línea] [Citado el: 23 de 2 de 2011.]

31. **Padrón, Dailí Corcho.** *Visión por Computadora para la Interacción con Entornos Virtuales a partir de la Captura de Imágenes y Movinientos mediante Webcams.* 2009.

GLOSARIO DE ABREVIATURAS

2D: Dos dimensiones o bidimensional.

3D: Tres dimensiones o tridimensional.

LED: Diodo Emisor de Luz.

DLL: Biblioteca de Enlace Dinámico.

HS: Saturación de Contraste.

XML: Lenguaje de Marcas Extensible.

CPU: Unidad Central de Procesamiento.

RAM: Memoria de Acceso Aleatorio.

API: Interfaz de programación de aplicaciones. Es el conjunto de funciones que ofrece cierta biblioteca para ser utilizada por otro software como capa de abstracción.

C++: Lenguaje que abarca tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos (POO). C++ permite trabajar tanto a alto nivel como a un bajo nivel.

GLOSARIO DE TÉRMINOS

A

Acelerómetro: Se denomina acelerómetro a cualquier instrumento destinado a medir aceleraciones.

Algoritmo: Es un conjunto pre-escrito 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.

B

Binarización: La binarización de una imagen digital consiste en convertir la imagen digital en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen.

Bluetooth: Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una baja cobertura y basados en transceptores de bajo costo.

C

Cognición: Facultad de los seres de procesar información a partir de la percepción, el conocimiento adquirido (experiencia) y características subjetivas que permiten valorar la información.

Componente Conexa: Se llama componente conexa, a cada uno de los conjuntos maximales conexos.

Comando: Es una instrucción u orden que el usuario proporciona a un sistema informático, desde la línea de comandos (como una *shell*) o desde una llamada de programación.

Compilador: Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.

I

Interfaz: Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Intuición: Es aquel conocimiento que es directo e inmediato, sin intervención de la deducción o del razonamiento, siendo habitualmente considerado como evidente.

M

Matiz: Es una de las propiedades o cualidades fundamentales en la propiedad de un color, definido técnicamente (en el modelo CIECAM02), como “el grado en el cual un estímulo puede ser descrito como similar o diferente de los estímulos como rojo, amarillo y azul”.

Mundo Virtual: Es un tipo de comunidad virtual en la que simula un mundo o entorno artificial inspirado o no en la realidad, en el cual los usuarios pueden interactuar entre sí a través de personajes o avatares, y usar objetos o bienes virtuales.

Multiplataforma: Se les llama de esa forma a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

O

Octree: Es una estructura "árbol" de datos en la cual cada nodo interno tiene exactamente 8 "hijos".

P

Pixel: Es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico.

S

Segmentación: La segmentación es un proceso por el que la imagen se divide en diferentes regiones para aislar las regiones de interés.

ANEXOS

Anexo 1: Aplicación obtenida.

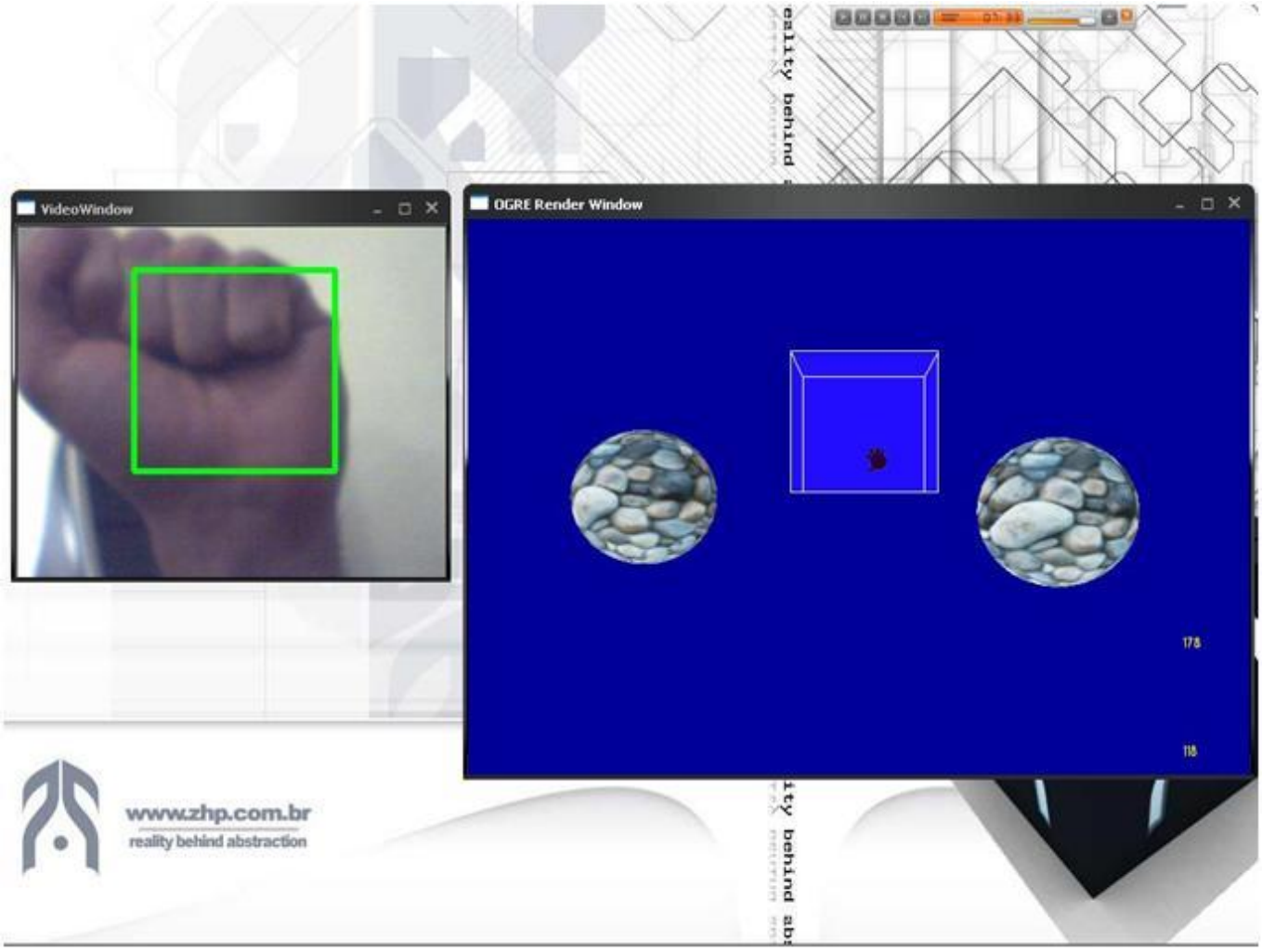

Figura 26: *Aplicación obtenida*

Anexo 2: Aplicación que utiliza el sistema de interacción natural.

Figura 27: *Aplicación que utiliza el sistema de interacción natural*