Universidad de las Ciencias Informáticas

Facultad 5

Componente para la estimación de la posición y orientación de objetos en escenas del mundo real.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: José Enrique Hernández Martínez.

Tutores: Ing. Mileydi Moreno Mirabal.

Ing. Ernesto de la Cruz Guevara Ramírez.

Ciudad de La Habana

Junio 2011

"Las ciencias de la computación están tan poco relacionadas con las computadoras, como la astronomía con los telescopios."

Edsger Dijkstra

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

J. Enrique Hernández Martínez

_____

Firma del autor

Ing. Mileydi Moreno Mirabal

_____

Firma del tutor

Ing. Ernesto de la Cruz Guevara Ramírez

_____

Firma del tutor

DATOS DE CONTACTO

Nombre y Apellidos: Mileydi Moreno Mirabal
Edad: ___ años
Ciudadanía: cubano
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Categoría Docente: Profesor Instructor
E-mail: mmirabal@uci.cu
Graduado en Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas, con cuatro años de experiencia en el tema de la Gráfica Computacional. Actualmente Profesor Instructor de la Universidad de la Ciencias Informáticas impartiendo la asignatura de _____.

Nombre y Apellidos: Ernesto de la Cruz Guevara Ramírez.
Edad: ___ años
Ciudadanía: cubano
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Categoría Docente: Profesor Instructor
E-mail: elguevara@uci.cu
Graduado en Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas, con cuatro años de experiencia en el tema de la Gráfica Computacional. Actualmente Profesor Instructor de la Universidad de la Ciencias Informáticas impartiendo la asignatura de _____.

*"*Quisiera agradecer, a todas aquellas personas que de una forma u otra contribuyeron en la realización de este trabajo: a mis tutores por guiarme, durante todo proceso de realización de esta investigación, a mis compañeros del proyecto: Alejandro, Tony y Frank, a mis familiares, en especial a mi tío Jesús, a mis amigos: Alexei, Denis, Julio y Yudiel, por ser los mejores amigos del mundo. A mi padre por haber podido contar con él siempre que lo necesité. A mi novia Lina por su constancia y su amor, por ser mi mano derecha y mi mano izquierda. Y muy en especial a mi queridísima madre por su devoción, su infinito amor y su ejemplo.“

"A mi madre, por ser la luz de mi universo.

Te quiero mucho."

RESUMEN

La presente investigación está dirigida a brindar una herramienta a desarrolladores de aplicaciones de Realidad Mixta que posibilite la interacción de escenas virtuales con sus usuarios finales a partir de imágenes obtenidas de una escena real. Surge a partir de la necesidad de desarrollar un componente para la Línea de Núcleo Gráfico del Departamento de Visualización y Realidad Virtual que permita calcular la posición aproximada de un objeto del mundo real con respecto a la cámara que lo capta.

Durante la investigación se exponen los principales elementos referentes a la Visión por Computador y técnicas de *tracking*. Se propone una solución para la estimación de la posición de objetos en escenas reales utilizando la biblioteca de desarrollo de aplicaciones de Visión por Computador OpenCV. Por último se presenta una descripción de la solución propuesta, respaldada por la correspondiente implementación de un componente que estima la posición de un objeto durante una secuencia de imágenes, donde los datos obtenidos a partir de esta estimación, son fácilmente usados para lograr una interacción de dicho objeto con escenas virtuales, creadas en cualquier biblioteca de gráfico por computadora. Dicho componente pretende servir de base para apoyar la creación de aplicaciones de Realidad Mixta en la Facultad 5 de la Universidad de las Ciencias Informáticas.

PALABRAS CLAVE

Visión por Computador, técnicas de *tracking,* Realidad Mixta.

INDICE DE FIGURAS

INDICE DE TABLAS

Índice de contenido

INTRODUCCIÓN

La Realidad Mixta (RM) es el nombre dado a todas aquellas aplicaciones que muestran información tridimensional combinada con datos y/o imágenes reales o viceversa. Dichas aplicaciones permiten agregar información gráfica a escenas reales en el caso de la Realidad Aumentada, o permitir a usuarios manipular imágenes virtuales en el caso de la Realidad Virtual, o utilizar la información que brinda un video para recrear y modificar una escena virtual para el caso de la Realidad Virtualizada. (1)

Existen diversas formas de clasificar los actuales sistemas de RM de acuerdo con el tipo de interfaz de usuario usada. En ese caso pueden mencionarse: los Sistemas de Ventanas, Inmersivos, de Realidad Aumentada, de Telepresencia, de Realidad Virtual en Pecera, y finalmente los llamados Sistemas de Mapeo por Video. Estos últimos basan su enfoque en la filmación, mediante cámaras de video, de una o más personas y la incorporación de esas imágenes a la pantalla del computador donde podrán interactuar en tiempo real con otros usuarios o con imágenes gráficas generadas por el computador. (2)

Para lograr desarrollar sistemas como los anteriormente mencionados es preciso desarrollar antes Sistemas de Visión por Computador o Visión Artificial que es una rama de la informática ubicada dentro del campo de la Inteligencia Artificial, cuyo propósito es programar un computador para que "entienda" una escena o las características de una imagen. Este propósito se consigue por medio de reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de gráficos entre otros campos. (3)

En la actualidad todas las tecnologías desarrolladas sobre la base de la Visión Artificial cuentan con numerosas áreas de aplicación como son la exploración petrolera, el desarrollo de juegos inmersivos, el diseño virtual en arquitectura, diseño industrial, ingeniería civil y diseño automotriz. Además de varias áreas de investigación como son las siguientes: visualización científica, visualización de información, interfaces hombre máquina, ambientes inmersivos para educación y entretenimiento, juegos, realidad virtual y realidad aumentada, entre otras (1).

No obstante, a pesar de la infinidad de posibilidades que abre, todavía en la Universidad de Ciencias Informáticas (UCI), queda un largo camino por recorrer en el ámbito referente a Sistemas de Visión por Computador (SVC).

La creación de escenarios virtuales que interactúen en tiempo real con el usuario es la base para muchos tipos de simulaciones y uno de los objetivos de desarrollo de la Línea de Núcleo Gráfico del Departamento de Visualización y Realidad Virtual de la facultad 5 en la UCI. En este aspecto se han hecho avances

significativos, aunque mayormente en lo referente a Realidad Aumentada, campo en el cual, se han desarrollado varias aplicaciones utilizando la herramienta de desarrollo ARToolkit. Aún así se están dirigiendo miradas hacia otras direcciones, con el objetivo de evitar algunos inconvenientes propios de esta biblioteca, como son los altos precios de las licencias de comercialización necesarias para vender un producto desarrollado usando esta herramienta. Por otra parte, las técnicas basadas en marcadores cuadrados, usadas por ARToolkit, presentan ciertas desventajas, tales como el fallo ante oclusiones parciales de los marcadores o ante condiciones de iluminación adversas, ya existen ocasiones en donde los marcadores enfrentan a la cámara en cierto ángulo y estos al ser planos, producen un brillo que dificulta o impide su detección, además está el hecho de que por su naturaleza plana estos marcadores deben, necesariamente de ubicarse de manera frontal a la cámara, limitando considerablemente su movilidad. No obstante, esta metodología basada en marcadores cuadrados no es la única que puede ser utilizada en la RM, también existen otras alternativas de probada utilidad en este campo. En ese sentido, se están realizando acciones en la búsqueda de tales alternativas, para poder incorporar nuevas técnicas. Una alternativa viable a esta situación seria, un componente de Visión por Computador para apoyar la realización de aplicaciones de RM, sin las restricciones que impone ARToolKit. Un punto de partida para alcanzar tal propósito, es lograr el reconocimiento de las posturas de los objetos del mundo físico, y la utilización de los datos obtenidos, para recrear escenas virtuales.

Por tanto el **problema científico** es:

¿Cómo estimar la posición y orientación de un objeto situado en una escena del mundo real?

Objeto de estudio:

Visión por Computador.

Campo de Acción:

Seguimiento 3D de objetos en flujo de imágenes.

Con el fin de resolver el problema científico se trazó el siguiente **objetivo general:**

> Desarrollar un componente de software que brinde las funcionalidades necesarias para registrar la posición y rotación de un objeto en 3 dimensiones (3D), con respecto a la cámara que mira una escena del mundo real.

Para el total cumplimiento del objetivo general se trazan las siguientes **tareas investigativas:**

1. Revisión de la bibliografía existente y análisis del estado del arte sobre Visión por Computador para analizar las mejores opciones y técnicas de implementación que den solución óptima a los objetivos trazados.
2. Implementar un sistema que permita el cálculo de la traslación y la rotación de un objeto en una imagen, y brinde las funcionalidades necesarias para crear una interfaz de usuario, que facilite la entrada de dato al mismo.
3. Ajustar la arquitectura del sistema a una arquitectura de componente.
4. Crear un prototipo de aplicación de RM que valide la funcionalidad del sistema a desarrollar y muestre algunas de sus principales aplicaciones.
5. Crear un manual de usuario, que sirva de guía a los usuarios del componente, para el correcto uso y comprensión de las funcionalidades del componente a desarrollar.

Para la realización de estas tareas se utilizarán los siguientes **métodos de investigación:**

Métodos Teóricos:

Analítico – Sintético: Para facilitar su estudio mediante la descomposición en sus principales partes y seguidamente resumir los distintos enfoques que le dan las metodologías de desarrollo de aplicaciones de RM, Visión por Computadora y técnicas de *tracking**.

Inductivo – Deductivo: Partir del estudio de los específicos y distintos procesos de desarrollo de aplicaciones de RM, y técnicas, analizadas, para obtener las ideas generales que permitan determinar las mejores características que se pueden reutilizar en la propuesta de solución concreta que se va a elaborar.

Capítulo I

En este capítulo se hace un análisis de estado del arte de los temas relacionados con el objetivo de la presente investigación. Se exponen tópicos acerca de a aplicaciones de RM, además de conceptos referentes a la Visión por Computador, así como los algoritmos computacionales para la estimación de posiciones de objetos en escenas reales más utilizados en la actualidad, entre otros temas estrechamente vinculados con la presente investigación.

1.1 Aplicaciones de Realidad Mixta.

Como ya se ha dicho, la RM permite la incorporación de objetos gráficos generados por ordenador en una escena tridimensional del mundo real o bien la incorporación de objetos reales en un mundo virtual. Una explicación visual de los campos que abarca la RM y sus relaciones aparece en (4). Según se describe en (1), las principales características de la RM son tres:

- Permite combinar ámbitos reales y virtuales.
- Es una tecnología interactiva y en tiempo real.
- Se puede registrar en tres dimensiones.

Algunos ejemplos son los siguientes:

- Aplicaciones de realidad aumentada, creadas con el ARToolkit. Dichas aplicaciones permiten agregar información gráfica a patrones 2D previamente definidos.
- Aplicaciones de realidad virtual, creadas con VRJuggler, donde se crea información 3D que el usuario puede manipular.
- Aplicaciones de realidad virtualizada, en donde se crean imágenes 3D con información de video real. Un ejemplo de este tipo de aplicaciones son las teleconferencias 3D, en donde cada participante puede ver una imagen 3D de los demás en un ambiente virtual, creada por adquisición de video. (1)

Teniendo en cuenta todo esto se puede inferir que dentro de las aplicaciones de RM, aquellas relativas a la realidad virtualizada, son las más cercanas en cuanto su finalidad a esta investigación. Por otra parte es necesario tener en cuenta que “el desarrollo de la RM implementa preferentemente sistemas de seguimiento basados en Visión Artificial ya que resultan más económicos y sencillos de implementar que los basados en sensores de hardware”. (5)

1.2 Visión por Computador.

Los Sistemas de Visión Artificial nacen con el desarrollo de la inteligencia artificial, con el fin de que una máquina pueda asimilar todos los elementos de una imagen, (6) concibe la visión artificial como el "proceso de extracción de información del mundo físico a partir de imágenes utilizando para ello un computador. La visión artificial para la comprensión de imágenes, describe la deducción automática de la estructura y propiedades de un mundo tridimensional, tanto estático como dinámico, a partir de una o varias imágenes bidimensionales del entorno físico.

La Visión por Computador es una gran herramienta para establecer la relación entre el mundo tridimensional y sus vistas bidimensionales. Por medio de esta teoría se puede hacer, por una parte, una reconstrucción del espacio tridimensional a partir de sus vistas y, por otra parte, llevar a cabo una simulación de una proyección de una escena tridimensional en la posición deseada a un plano bidimensional.

Según se explica en (7), el campo de aplicaciones de la Visión Artificial abarca áreas como:

- **Fotogrametría:** En la fotogrametría se persigue realizar mediciones del espacio 3D a partir de fotografías tomadas de él. De esta manera es posible medir superficies, construcciones, objetos, etc. Asimismo se puede llevar a cabo una topología de un terreno.
- **Rectificación Métrica:** Mediante esta técnica es posible hacer correcciones de perspectiva y correcciones de distorsión de lente de una cámara.
- **Reconstrucción 3D:** A partir de las vistas, mediante la técnica de triangulación, es posible obtener un modelo 3D de un objeto proyectado en varias vistas.
- **Computación Gráfica:** Si se tiene un modelo de la formación de la imagen *f:* 3D - 2D, es posible entonces simular gráficamente las vistas bidimensionales que se obtendrán de un objeto tridimensional. Las aplicaciones de realidad virtual emplean esta teoría.
- ***Matching**** **y** ***Tracking*****:** Por medio del *Matching* y *Tracking* es posible encontrar la correspondencia entre puntos en varias imágenes. Los puntos correspondientes son aquellos que representan una proyección del mismo punto físico en el espacio 3D.
- **Estimación de Movimiento:** Mediante una cámara que toma imágenes de un objeto en movimiento es posible estimar el movimiento del objeto a partir de los puntos de correspondencia en la secuencia de imágenes.

La teoría desarrollada en las dos últimas décadas por la comunidad de *Computer Vision* ha estado dirigida a analizar en detalle la información que puede ser obtenida a partir de un conjunto de vistas de una escena. De acuerdo al criterio de (7): "la teoría que mejor modela la formación e interpretación de las imágenes es la Geometría Proyectiva, esta conjunto con la teoría de gráficos, reconocimiento de patrones, aprendizaje estadístico, procesado de imágenes, conforman los principales campos involucrados con la visión artificial."

Como se puede ver la Visión Artificial proporciona las técnicas necesarias para el reconocimiento de objetos y patrones predeterminados, entre tantas otras funcionalidades, pero sobre todo estas dos, son muy utilizadas en el desarrollo de aplicaciones de RM, Aumentada y Virtual. En el caso de esta investigación es más conveniente dirigir la discusión a temas relacionados con el *tracking* y reconocimiento de objetos.

A continuación se exponen los distintos modelos y métodos existentes para establecer la función de transferencia 2D-3D en un Sistema de Visión Artificial, además de analizar los algoritmos más conocidos para llevar a cabo la estimación de la posición de un objeto con respecto a un dispositivo de visión digital.

1.3 Seguimiento de un objeto en una secuencia de imágenes.

Una línea esencial de desarrollo dentro de muchos SVC, es aquella dirigida a permitir el seguimiento o rastreo (*tracking*) de objetos de interés en una secuencia de imágenes. Estos sistemas dependen de un seguimiento eficiente de los distintos marcadores de referencia, en caso de necesitarlos, en dependencia del algoritmo de *tracking* a emplear.

Según explica (8), la estimación de la posición de un objeto en tiempo real es un problema investigativo de creciente popularidad, con la finalidad de resolver la localización y orientación de objetos en movimiento en una escena dada. Las aplicaciones de esta técnica son muchas: estimación de la posición de la cabeza, guía inteligente de vehículos, *tracking* 3-D de objetos, por solo citar algunos ejemplos.

De acuerdo con (9), la tecnología, metodología y algoritmos para lograr este fin son tan variados como las situaciones en que se requiera su empleo. Cada sistema y su implementación deben adaptarse a las necesidades y recursos de los desarrolladores del mismo. Estos se pueden dividir en dos grupos fundamentales: los que utilizan cámaras estéreo, o sea dos o más cámaras para obtener una visión de una misma escena desde varias perspectivas y los que usan una sola cámara. Los primeros se basan principalmente en cálculos geométricos de triangulación, por ejemplo: en un sistema de visión binocular se

debe detectar la posición de un objeto relativo a la ubicación de cada una de las cámaras (ojos) y, al mismo tiempo, el movimiento ya sea del objeto o de las cámaras genera un desplazamiento de la imagen en relación al plano de cada cámara. Por otra parte los segundos están basados mayoritariamente en marcadores, aunque también existen metodologías para la localización de objetos que prescinden de marcadores (basadas en el análisis de contornos) todas ellas emplean varias técnicas de tracking mediante la implementación de distintos algoritmos de estimación de la geometría proyectiva. (9)

1.3.1 Técnicas de *tracking* de objetos.

Definición de *tracking* óptico(tracking), según (10): El *tracking* óptico es el control de los parámetros (posición, orientación y zoom) de un determinado objeto dentro del campo de visión, en función de las características extraídas de la imagen.

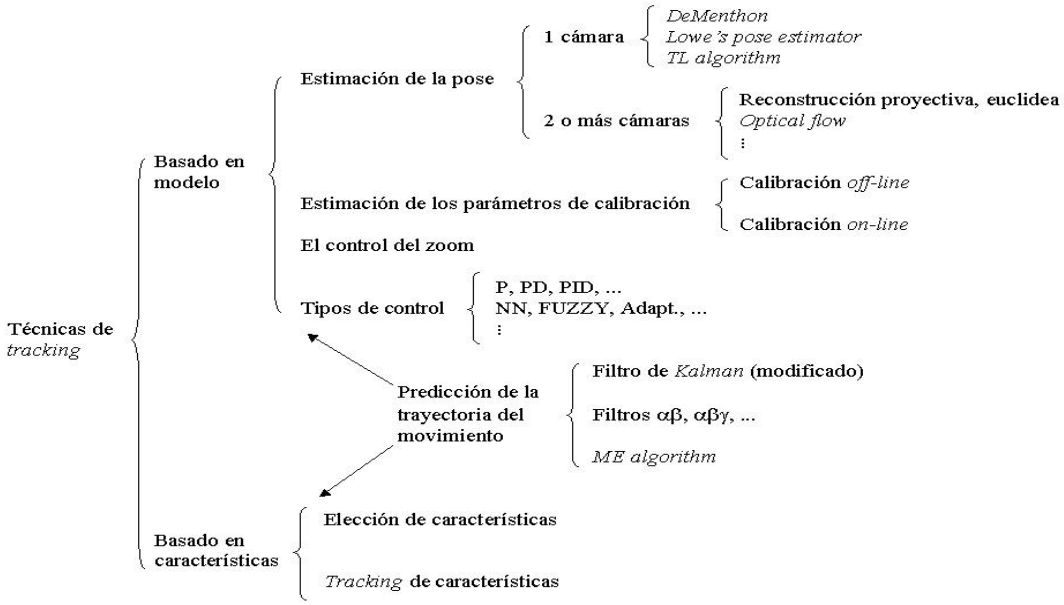

Figura 1: Técnicas de *tracking* y sus respectivos algoritmos.

1.3.2 *Tracking* basado en características.

Este tipo de *tracking* se basa en las características extraídas de la imagen y no en la búsqueda de un modelo conocido. La idea fundamental en la que se basa el *tracking* de características es: ¿Por qué hacer *tracking* del objeto entero cuando se puede obtener el mismo resultado haciendo *tracking* solo de las

características? Este planteamiento suele ser computacionalmente más eficiente que el basado en modelo, pero es menos robusto. (10)

Elección y *tracking* de características: El paso más importante antes de hacer el *tracking* es determinar las características a seguir. (Ej. Textura, puntos, esquinas, colores, etc.). Discriminar entre buenas y malas características. Se calcula para ello el vector gradiente de la imagen en ese punto y con los valores propios, podremos saber la calidad de la característica. Sólo aquellas que cumplan un cierto umbral serán finalmente elegidas.

Una vez seleccionadas las características buenas a seguir, es sencillo llevar un registro del movimiento de cada característica, con lo que queda determinada la trayectoria de movimiento del objeto. (10)

1.3.3 Predicción de la trayectoria.

Se realiza cuando los requerimientos dinámicos o la velocidad del objeto a seguir es lo suficientemente alta como para que su localización entre una imagen y la siguiente de la secuencia sea muy diferente. La predicción proporciona información sobre la posible *pose** cuando se esté analizando una ventana de la imagen, y existan oclusiones parciales o totales del objeto o incluso éste salga del marco de imagen.

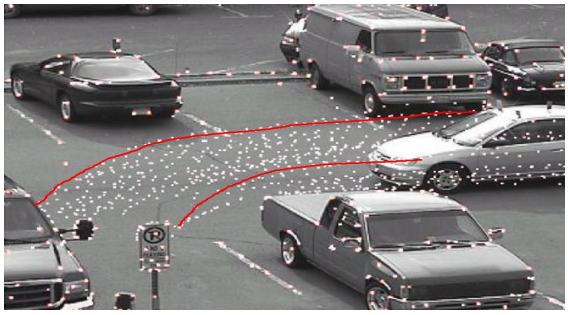

Figura 2: Trayectoria de movimiento del objeto.

Filtro de Kalman (modificado): El filtro de Kalman con sus extensiones EKF y EIKF son los algoritmos de predicción de la posición más utilizados (junto con el IMM – Interacting Multiple Model). Se usa una combinación de los datos de obtención de la pose y predicción para hacer más robusto el algoritmo. (10)

Filtros ab, *abg*: El filtro ab, es un sencillo estimador de la posición con un tiempo de cómputo muy bajo y que proporciona unos resultados satisfactorios. Como modificación del primero, se tiene el filtro *abg*, que realiza la estimación de la aceleración, efecto que no se contemplaba y que provocaba el mayor error de estimación.

ME algorithm (Moving Edges algorithm): Algoritmo de estimación que calcula el desplazamiento normal entre dos imágenes sucesivas a lo largo de la proyección del contorno del modelo del objeto.

1.3.4 Tipos de control.

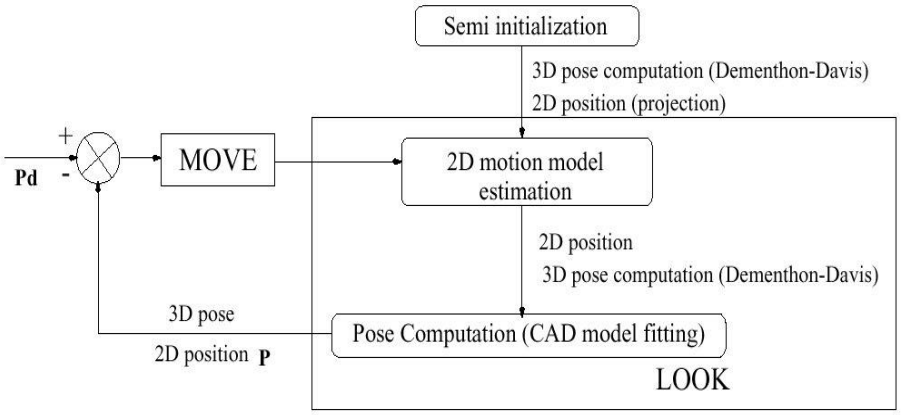

Figura 3: Esquema para tipos de control.

El control a aplicar responde al esquema presentado en la figura. Donde el bloque MOVE implementa un regulador de los ya conocidos: P, PD, PID, NN, FUZZY, *Adapt*. (10)

El control del Zoom

Una de las características del *tracking* basado en modelo es que una vez obtenida la pose, se puede extender con facilidad el control para el manejo del *zoom* en función de la profundidad o alejamiento del objeto. Con ello mantenemos el objeto con el mismo tamaño en la imagen aunque se acerque o aleje.

Estimación de los parámetros de calibración.

Los parámetros de calibración son necesarios para la obtención de la pose del objeto. Estos parámetros cambian con variaciones del *pan*, *tilt* o *zoom*, por lo que es necesario calibrar la cámara en todo el rango de estas variables. La calibración se puede realizar off-line (antes del proceso de *tracking*) u on-line (durante el proceso de *tracking*).

1.3.5 *Tracking* basado en modelo.

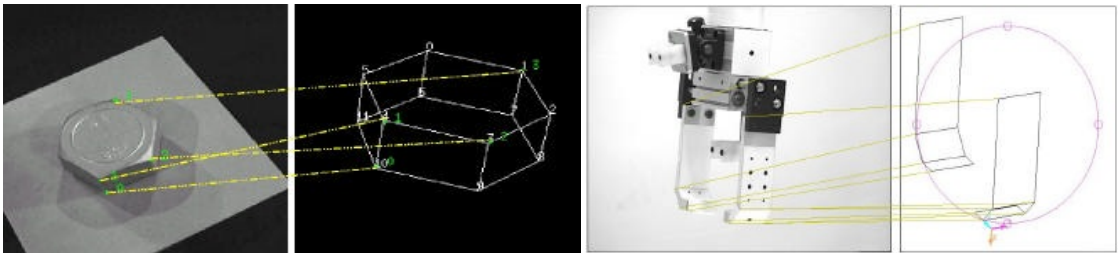

Figura 4: Ejemplos de *Tracking* basado en modelo.

Este tipo de técnicas se basan en el conocimiento del modelo del objeto. Este modelo puede ser un modelo CAD (*Computer-Aided Design*) o una proyección del objeto. Esta técnica es más robusta que la basada en características.

Estimación de la pose:

Usando 2 o más cámaras: Teniendo varias imágenes, esh posible obtener la pose de un objeto estableciendo la correspondencia entre puntos. Para ello se puede emplear alguna de las técnicas propuestas: Reconstrucción proyectiva-euclidiana: *Optical flow*.

Usando 1 cámara: Con la información aportada por una sola imagen, se puede obtener la pose de un objeto si se conoce su modelo. Para ello es posible emplear alguno de los algoritmos propuestos: Lowe, TL, DeMenthon.

***Lowe's pose estimator* (usando 1 cámara):** Lowe propone una *reparametrización** de las ecuaciones de proyección para simplificar el cálculo y expresar la traslación en términos de coordenadas de la cámara. La estimación de la pose basada en modelo usando cuatro puntos se llama "problema de la perspectiva de cuatro puntos" (P4P).

***TL algorithm* (usando 1 cámara):** Emplea el cálculo iterativo basado en el método de Gauss-Newton para resolver el sistema de ecuaciones propuesto. Algoritmo del tipo P4P con menor gasto computacional que Lowe y basado también en estimación de la pose basada en modelo.

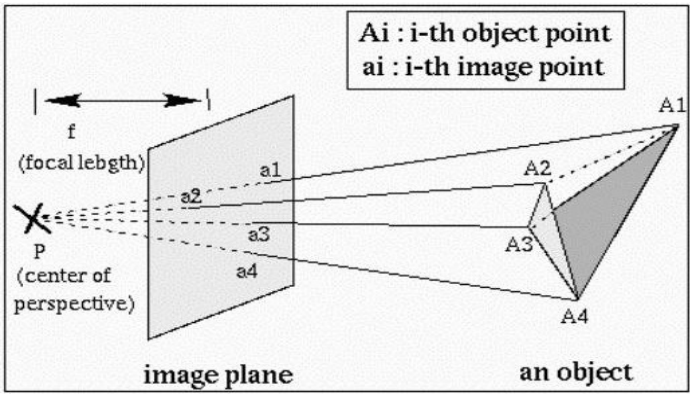

Figura 5: TL algorithm.

Algoritmo de DeMenthon (usando 1 cámara): Este algoritmo conocido como *the POSIT algorithm* empieza con una aproximación de la proyección perspectiva, refinándola de forma iterativa hasta que el error sea menor que un determinado umbral. Para ello hemos de indicarle puntos no coplanarios, teniendo en cuenta que la precisión de la pose obtenida dependerá de la distancia de uno de los puntos hasta el plano definido por los otros tres. Este algoritmo resuelve el problema de la pose de un objeto con rapidez superior a los algoritmos de una sola cámara mencionados anteriormente.

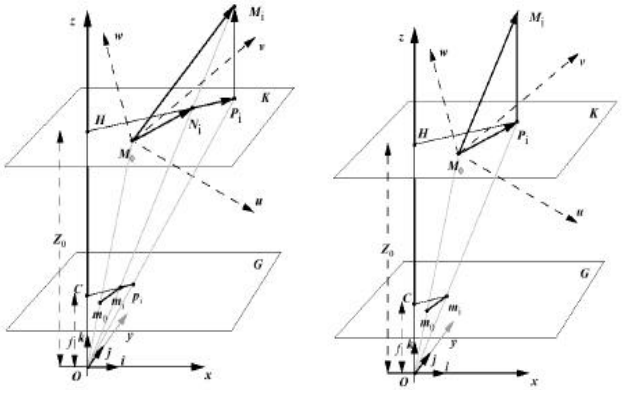

Figura 6: DeMenthon.

Luego de haber explorado una serie de soluciones computacionales al problema del cálculo de la pose de un objeto, se consideran más atractivas las técnicas de *tracking* basadas en modelos, y dentro de ellas, específicamente, la estimación de la pose mediante el algoritmo DeMenthon (*POSIT algorithm*), tomando en consideración el hecho de que este ofrecería una menor utilización de dispositivos de visión digital y una mayor rapidez de cálculo.

1.4 Estimación de la pose mediante la correspondencia de cuatro puntos no coplanarios.

Atendiendo a lo mencionado en (9), la estimación de la pose es un tema polémico tanto en la fotogrametría como en la Visión Artificial, y es ampliamente usando sobre todo en la navegación robótica. Principalmente, una certera y rápida estimación de la pose es la base para la Realidad Aumentada, en tal caso se necesita computar la posición relativa de la cámara con respecto al marcador con alta precisión y a tiempo real. Diferentes tipos de metodologías pueden ser usadas para la estimación de la pose, tales como la correspondencia entre puntos, líneas, curvas y superficies.

Explica (9) que: la herramienta para Realidad Aumentada ARToolkit usa la correspondencia entre cuatro líneas rectas. El paradigma de la correspondencia entre líneas posee el mérito de que las líneas son fáciles de detectar. Sin embargo, estos sistemas también tienen la desventaja de estar propensos a fallar en caso de oclusión. Si parte del rectángulo (marcador) no es visible ARToolkit no podrá determinar la pose del marcador. Por otra parte la estimación por medio de curvas y superficies es un método relativamente nuevo, pero resulta demasiado sofisticado e implica mayores dificultades en la detección de tales características. Es por eso que el enfoque va dirigido a la correspondencia entre puntos para dar solución al problema de la estimación de la pose.

La correspondencia entre puntos conocida también como “el problema de perspectiva de n puntos” (PnP), basada o no en la matriz intrínseca de la cámara, conoce a priori la ubicación de una cierta cantidad de puntos. De acuerdo con Haralick (11), el problema de perspectiva de tres puntos (P3P) fue resuelto por primera ocasión por el matemático alemán Grunert en 1841. Sin embargo no es hasta 1981 que este problema es llevado a la comunidad de *Computer Vision* por Fishler y Bolles (12). Ellos arribaron a una solución directa a este problema que reveló que a través de un conjunto de tres puntos no se puede obtener una solución única (generalmente existirían cuatro ambigüedades). Luego Long Quan and Zhongdan Lan (13), propusieron una familia de métodos lineares que brindan una solución única a la determinación de la posición para puntos cuatro genéricos de referencia. No obstante, su algoritmo requiere doble descomposición vectorial y es muy costoso computacionalmente. Por último DeMenthon y Davis combinaron algoritmos directos e iterativos para resolver el problema de P4P con solo veinticinco líneas de código lo cual se conoce como *the POSIT algorithm*. Es certero, rápido pero necesita de al menos cuatro puntos no coplanarios. (9)

1.4.1 Algoritmo POSIT (POSIT algorithm).

Este algoritmo fue el seleccionado para darle solución a los objetivos trazados. Su implementación permite la ubicación de la posición de un objeto a partir de una sola imagen, asumiendo que se puedan detectar en la misma cuatro o más puntos no coplanarios que definan dicho objeto, además de conocer su respectiva geometría.

Según (14), el Posit combina dos algoritmos, el primero (*POS _Pose from Orthography and Scaling),* el cual aproxima la proyección perspectiva con una proyección ortográfica escalada y encuentra la matriz de rotación y el vector de traslación resolviendo un sistema lineal de ecuaciones. El segundo algoritmo (*POSIT _POS with Iterations*) usa en su ciclo de iteración la pose aproximada encontrada por *POS _Pose* en orden de calcular mejor la proyección ortográfica escalada de los respectivos puntos, luego aplica el *POS _Pose* a los nuevos puntos calculados en vez de los obtenidos de la proyección original de la imagen, *POSIT _POS* converge en unas pocas iteraciones a un preciso posicionamiento de dimensiones.

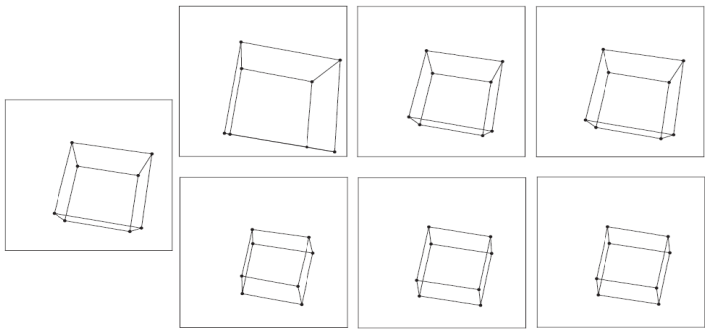

Figura 7: Imágenes de Perspectiva (fila superior) y proyección ortográfica (fila inferior) para el cálculo de la pose en las primeras tres iteraciones (izquierda a derecha) del algoritmo POSIT para la imagen de un cubo (izquierda).

El algoritmo Posit tiene una complejidad computacional mucho menor que las demás técnicas de estimación basadas en modelo abordadas anteriormente. Para N concordancias entre los puntos del objeto y los de la imagen Posit requiere alrededor de 24 N operaciones aritméticas mas raíz cuadrada de 2 cálculos por iteración, o sea para 8 puntos característicos y 4 iteraciones, se necesitarían alrededor de 800 mas raíz cuadrada de 8 operaciones aritméticas. A manera de comparación, las soluciones propuestas por Long Quan and Zhongdan Lan (13) resolverían un sistema lineal de 2N ecuaciones usando la matriz pseudoinversa, lo cual requiere cerca de 1012 N + 2660 operaciones aritméticas (multiplicación y división). Para el caso de *Lowe's method,* para cada iteración, cerca de 202 N + 21 N^2 operaciones solo para preparar el ciclo de iteración, luego 36 N^2 + 108 N+ 460 en cada iteración, para 8

puntos característicos en 4 iteraciones el total de operaciones es alrededor de 25 veces mayor que con Posit (14). Basándose en estas comparaciones, se puede asumir que el método propuesto para esta solución presenta definitivas ventajas sobre los anteriores, para aplicaciones de tiempo real.

1.5 Problema del *Tracking*.

De acuerdo con (15), dentro del problema del *tracking* de la cámara existen diferentes técnicas para su resolución, entre las que se encuentra el *tracking* óptico. Esta alternativa basada en Visión por Computador se ha convertido en una de las más populares debido a sus buenos resultados y a su bajo coste en comparación con el resto. El *tracking* óptico puede dividirse en dos grupos: basado en marcadores y sin marcadores. El segundo de ellos utiliza las características naturales del entorno tales como bordes, esquinas, texturas, etc. para calcular la pose de los objetos o de la cámara en el caso de los sistemas de Realidad Aumentada. Sin embargo, además del problema de la inicialización tiene dos grandes problemas como es la acumulación del error y la reinicialización. Dentro de este grupo también aparece un nuevo subgrupo, basado en el modelo, que consiste en utilizar un modelo 3D de la escena. Esta solución tiene el inconveniente de que algunas veces la generación del modelo 3D puede resultar difícil y puede que requiera mucho tiempo su elaboración.

En la alternativa basada en marcadores, por el contrario, el principal problema viene dado por el trabajo y limitación que supone intervenir el entorno. Siempre que se quiera exportar un sistema a otro lugar habrá que adecuar el nuevo entorno al sistema, añadiendo los marcadores a sus nuevas localizaciones. Además, debido a que se necesita tener en todo momento un marcador visible para obtener la pose de la cámara, existen varias propuestas que utilizan múltiples marcadores para conseguir un sistema más robusto. No obstante, el aumento de robustez supone incrementar la intervención del entorno y una menor adecuación del entorno supone un sistema menos robusto. (15)

Tal como se explica en (15), existen diferentes tipos de marcadores. En el proceso de identificación de los marcadores basados en patrones, un criterio común a utilizar para comparar la imagen actual con una base de datos de *templates** es por medio del *L2 norm,* el cual es un algoritmo de comparación de valores matriciales. Se calcula el *L2 norm* del marcador actual con cada uno de los marcadores almacenados y se devuelve aquel marcador almacenado cuyo *L2 norm* sea menor. Para calcular el *L2 norm* se utilizan los niveles de gris del interior de los marcadores, mostrando una correlación entre ambos. Tras las comparaciones, se aceptará un emparejamiento entre marcadores en relación a un umbral prefijado. Este

proceso tiene el problema de que dos marcadores similares darán resultados muy parecidos, provocando interpretaciones erróneas, y la diferente configuración de los niveles de gris en función de la rotación del marcador. Además, se necesita una fase de entrenamiento para fijar el umbral de aceptación.

Muchos marcadores visuales como movimiento, textura, estructura y color han son empleados base para el *tracking* de objetos. Entre todos estos el color es una efectiva forma de detectar la presencia de un marcador en una escena. El color ofrece muchas ventajas sobre modelos geométricos, tales como robustez bajo oclusión, redimensionamiento y cambios de resolución, así como transformaciones geométricas. Además el costo computacional para el procesamiento de colores es considerablemente menor comparado con aquellos asociados con el procesamiento de complejos modelos geométricos. (5)

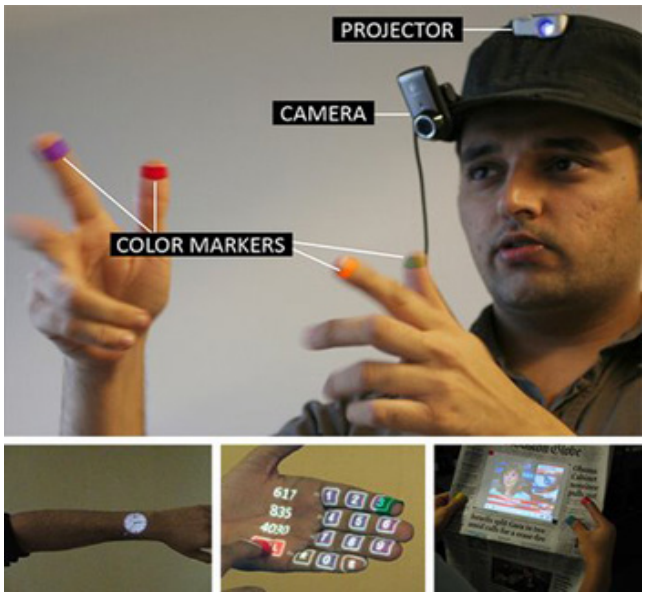

Figura 8: Utilización de marcadores de colores aplicada a la Visión por Computador.

El sistema de marcadores que se seleccionó para el *tracking* de objetos de esta investigación es el basado en la detección de colores, ya que ofrece una mayor robustez a la solución en cuestión, además de menor costo computacional lo que le aporta mayor velocidad de respuesta lo que lo convierte más atractivo para requerimientos de interacción a tiempo real.

1.6 OpenCV.

OpenCV es una biblioteca abierta de Visión por Computadora. Esta biblioteca está escrita en C y C++ y corre en las plataformas de Linux, Windows y Mac OS X. También existe un activo desarrollo en interfaces para Python, Ruby, Matlab, y otros lenguajes.

OpenCV ha sido diseñado para una eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real. Su principal aporte es brindar una infraestructura de fácil uso que ayuda a los desarrolladores a construir confiables y sofisticadas aplicaciones de Visión por Computadora en corto tiempo. Esta biblioteca ofrece más de quinientas funciones las cuales se pueden esparcir en muchas áreas de la visión, incluyendo inspección de productos de fábrica, visión médica, seguridad, interfaz de usuario, calibración de cámara, visión estéreo, y robótica (16), véase además (17).

1.7 Consideraciones finales para el Capítulo I

Las aplicaciones de realidad virtualizada ubicadas dentro del campo de las aplicaciones de RM utilizan la información de un video para recrear objetos en un mundo virtual, por lo que se apoyan en técnicas de Visión por Computador para lograr interpretar los datos de la fuente, ya sea una cámara o un archivo de video. Un Sistema de Visión Artificial utiliza ciertas técnicas para darle seguimiento a un objeto en una secuencia de imágenes. Dentro de estas llamadas técnicas de *tracking*, se halla el *tracking* basado en modelos, el cual puede ser resuelto mediante la implementación del Algoritmo Posit. Este algoritmo tiene como salida la matriz de rotación y el vector de traslación del objeto captado en las imágenes secuenciales, dicha matriz y vector, pueden ser utilizados fácilmente para dibujar cualquier objeto virtual con una orientación y una traslación bastante aproximada a la del objeto real con respecto a la cámara que captura o capturó el video. El algoritmo Posit para su ejecución satisfactoria requiere de 4 puntos no coplanarios como entrada, para rastrear dichos puntos en la imagen se usan varias técnicas como el *tracking* óptico, dentro de esta última las basadas en marcadores, donde los marcadores basados en color ofrecen ciertas ventajas sobre otros tipos de marcadores. La biblioteca de código abierto OpenCV brinda múltiples funcionalidades orientadas a facilitar la implementación de SVC y es la utilizada para darle solución al problema planteado en esta investigación.

Capítulo II

En este capítulo se propone una solución al problema de la investigación. También se define la estructura principal del prototipo funcional elaborado y se explica de forma detallada la solución.

2.1 Descripción de la solución propuesta.

La implementación de la solución a los objetivos planteados está orientada a desarrolladores de aplicaciones de realidad virtualizada. Dicha solución debe brindar al programador las herramientas necesarias para que una escena virtual diseñada por él, interactúe con una secuencia de imágenes de un objeto real ya sea desde un archivo de video o captadas por una cámara, todo esto en tiempo real. Para ello el sistema debe ofrecer ciertas funcionalidades que realicen las operaciones correspondientes a la parte de Visión Artificial, y que por otra parte permitan la adaptación del sistema a los diferentes escenarios según las necesidades de los usuarios.

Para dar respuesta a los objetivos trazados se seleccionó OpenCV como biblioteca de desarrollo por las características expuestas en el epígrafe 1.6. Como método de *tracking* de objetos se eligió el basado en modelos y para la estimación de la pose se utilizó el algoritmo Posit. Finalmente para el *tracking* óptico de marcadores se optó por los basados en colores, por las ventajas expuestas en el epígrafe 1.6.

Para el proceso de desarrollo de la aplicación prototipo se seleccionó la metodología RUP, el lenguaje de modelado UML y la herramienta Rational Rose Enterprise Edition 2003 (Rational). Se optó por el lenguaje C++ y el compilador de Visual C++ que ofrece Microsoft Visual Studio 2008 para la implementación y para manejo de los gráficos por computadora la biblioteca OpenGL.

Todo lo planteado posibilitó tener las herramientas necesarias para comenzar a desarrollar un componente de Visión por Computador e integrarlo a una aplicación prototipo, de manera demostrativa. Dicho componente fue nombrado VInterax.

2.1.1 Estructura funcional de la solución.

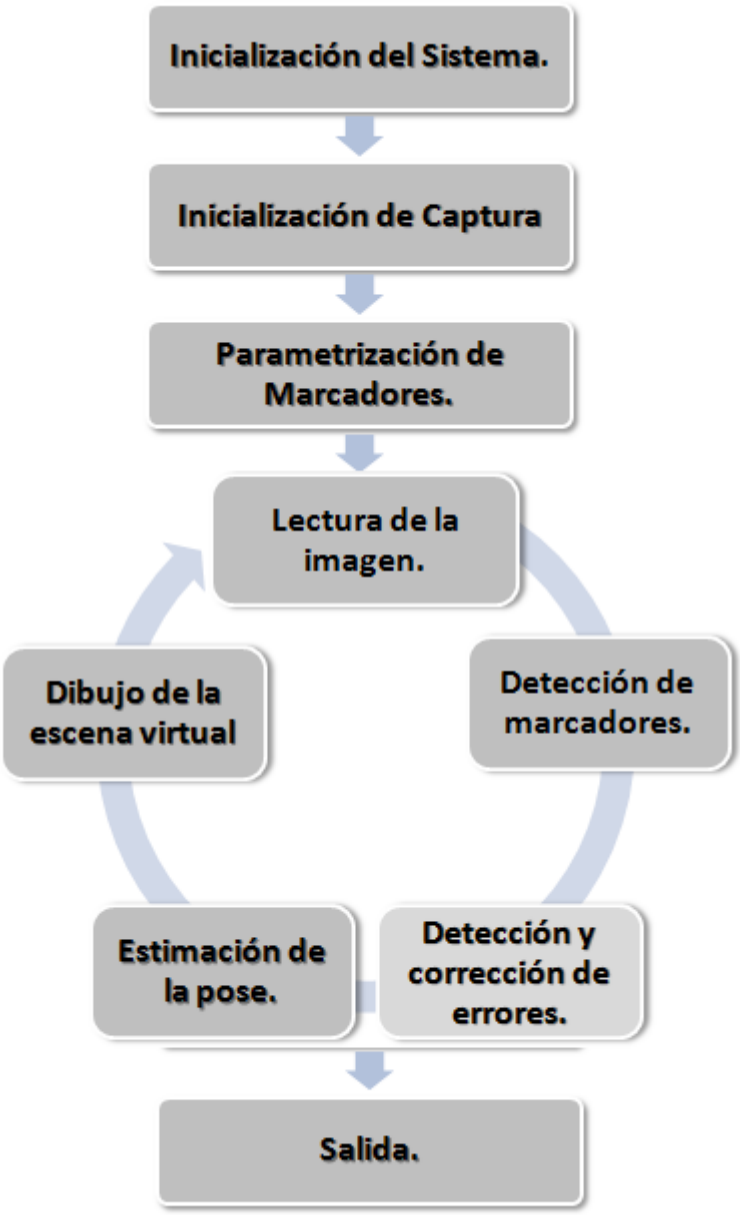

Figura 9: Estructura funcional de la solución.

No.	Fase	Descripción
1.	Inicialización del sistema	En esta fase el sistema de inicializa los datos para la captura de video, el cálculo de la pose y se prepare para el reconocimiento de los patrones.
2.	Inicialización de la captura de video.	En esta fase se le orienta al sistema que comience la captura de imágenes desde un archivo de video o desde una cámara.
No	Fase	Descripción
3.	*Parametización* de los marcadores(o patrones)	En esta etapa se reciben como parámetro los colores a los que deben dársele seguimiento en la imagen. También se pueden redefinir varias características del sistema que pudieran mejorar el reconocimiento de los patrones.
4.	Lectura de la imagen	El sistema lee el fotograma de video que se esté mostrando en ese momento.
5.	Detección o Reconocimiento de marcadores.	El sistema busca en la imagen por cuatro objetos puntuales (patrones) con los colores indicados en la fase 3. Estos representan los puntos no coplanarios que se necesitan para ejecutar efectivamente el algoritmo de estimación de la pose.
6.	Detección y corrección de errores.	En esta fase se comprueba que los datos obtenidos por la etapa anterior sean correctos de acuerdo con una serie de reglas establecidas empíricamente. Además se usan otros métodos de corrección para brindarle cierta estabilidad frente a errores al sistema. Los datos que el sistema utiliza para el reconocimiento de patrones son actualizados y

		optimizados siempre que sea necesario, esta es una manera de prevenir errores.
7.	Cálculo de la pose del objeto.	Con los datos obtenidos y procesados se computa el algoritmo Posit obteniendo como salida la estimación de la matriz de rotación y el vector de traslación del objeto captado en la imagen.
8.	Dibujo de la escena virtual.	Los datos obtenidos en el paso anterior son interpretados por el usuario y usados para dibujar su escena virtual.
9	Salida.	Se le orienta al sistema terminar el *tracking*, el cálculo de la pose y liberar los recursos tenga en uso.

Tabla 1: Descripción de las fases de la solución propuesta.

La fase 6 es opcional. Es posible hacer que al sistema prescindir de esta fase en aras de lograr una mayor velocidad de respuesta, lo cual implicaría una disminución de la robustez del sistema en cuestión.

2.3. Captura de imágenes.

La captura de imágenes se puede realizar desde un archivo de video, o en tiempo real mediante una cámara (webcam) conectada a la PC, también aunque no es objetivo fundamental se pueden cargar imágenes de archivos de tipo ".jpg" o".png". Es preciso enfatizar en que la efectividad de todo el sistema depende mayormente de la calidad del video captado, ya que por una parte, la correcta detección de los marcadores de colores es la base para el resto del proceso de estimación, por otra parte el sistema no brinda funcionalidades para mejorar la calidad del video en cuanto a contraste de colores o reducción del ruido en la imagen. Aunque si brinda una funcionalidad que identifica los bordes de los colores en la imagen, asumiendo que en una imagen donde los bordes entre los colores estén bien definidos, la detección de los mismos es más efectiva. Esta funcionalidad puede resultar útil si se quiere tener un criterio de la calidad de las imágenes capturadas.

2.4 Detección de marcadores.

La detección de los marcadores basados en colores es una de las principales tareas del sistema a partir de esta se desarrollan los demás procesos. Una mala interpretación de la imagen implicaría una salida errónea o poco precisa, es por eso que en dicha tarea se ocupa el mayor peso en costo computacional de la solución.

En el caso de la presente investigación la imagen es registrada en busca de 4 colores diferentes, cada uno representa un punto de específico del objeto al cual se le está efectuando el *tracking* (centro, ancho, altura y profundidad). Nótese que cualquier figura puede estar definida por estos puntos ya que los mismos definen las tres dimensiones espaciales existentes.

2.4.1 *Parametrización* de marcadores o patrones.

Esta funcionalidad está orientada a brindar la facilidad de que se puedan escoger los colores a los que se le va a dar seguimiento, y a ajustar la precisión de este procedimiento. Es válido agregar que absolutamente cualquier color puede ser utilizado para crear un patrón, y deben ser 4 patrones con colores diferentes ubicados en cada una de las esquinas que definen las tres dimensiones del objeto al que se le efectúa el *tracking*.

Un patrón de color es preferentemente un objeto puntual con un color determinado y un tamaño relativamente pequeño para que su localización no ocupe un área que dé lugar a imprecisiones en cuanto a la verdadera ubicación del patrón y relativamente grande para que permita al algoritmo detectarlo o sea que sea visible a cierta distancia de la cámara en dependencia de las necesidades del usuario.

Para una correcta interpretación de la localización de los patrones en la imagen es necesario que los colores estén lo mejor definidos posibles. Obviamente la selección de los patrones es esencial en el buen desempeño del sistema. Por una parte, el hecho de que los patrones requeridos para la ejecución del algoritmo de estimación de la pose sean patrones de colores, facilita mucho las posibilidades de ser creados de una manera sencilla por cualquier usuario. Además la facilidad de poder escoger los colores a seguir reduce enormemente las limitaciones para su creación, a la vez que le brinda al sistema mayor adaptabilidad a las necesidades y posibilidades de los usuarios. Por otra parte el reconocimiento de colores exige cierta calidad en la imagen capturada, además los ciertos requisitos en la confección de patrones expuestos anteriormente.

Otro requisito es especificar correctamente la correspondencia de los patrones de color con las dimensiones del objeto captado, en cuanto al centro, alto, ancho y profundidad del objeto respectivamente. Esto consiste en seleccionar el patrón ubicado en una de las esquinas de las dimisiones del objeto y especificar que ese color va a ser tomado como esa determinada dimensión y no otra.

2.4.2 Selección del patrón en la imagen.

Este proceso consiste en seleccionar el color que va definir a cierto patrón para que cuando el algoritmo de detección lo encuentre pueda reconocer su ubicación. El sistema brinda la posibilidad de que esta entrada de datos pueda ejecutarse simplemente haciendo clic en la región de la imagen donde se ubica el patrón. Esta facilidad ofrece una gran comodidad y agiliza notablemente el proceso de *parametrización*. Además se cuenta con la funcionalidad de poder ampliar y reducir el área de selección alrededor del puntero del mouse. Esta funcionalidad produce ciertos cambios en el la efectividad de la detección de marcadores por eso se debe ajustar a las necesidades de los usuarios, un área de selección óptima es aquella que pueda seleccionar la mayor parte del marcador sin ser más grande que el área que ocupa el marcador en la imagen.

2.4.3 Proceso de la detección de colores.

El proceso de detección de los colores correspondientes a los patrones o marcadores, se realiza haciendo una búsqueda de una porción de la imagen, con características similares en cuanto a coloración, con respecto a la muestra de color inicializada en la selección de colores.

Cuando se realiza la selección explicada en el epígrafe anterior el sistema almacena una copia de la imagen tomada con la funcionalidad de hacer clic en la imagen. Esa imagen es tomada como *template** o plantilla, para realizar las búsquedas de una porción parecida en la imagen general. Un *template* se puede definir en este caso como una muestra de color sacado de una pequeña parte de la imagen captada (el proceso de creación de un *template*, ocurre con cada evento de clic sobre la ventana de reproducción de video, en realidad el *template* es una copia de la pequeña área en la imagen alrededor del puntero mouse). La metodología de buscar el punto donde coincide un *template* en la imagen que se esté reproduciendo en ese momento está basada en la suposición de que la porción de la imagen que representa el *template* no debería cambiar mucho en el intervalo de un fotograma a otro.

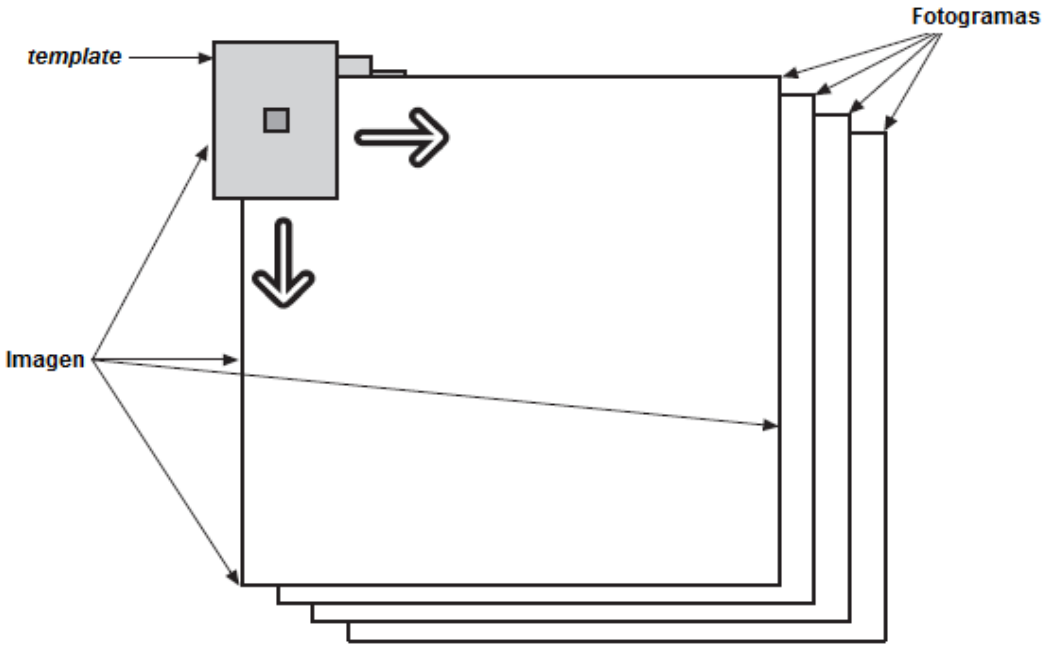

Figura 10: Uso del *template* en el reconocimiento de colores.

El algoritmo para localizar la posición de un *template* en la imagen tiene cierto rango de inexactitud, el objetivo de esto es encontrar su ubicación en una región de la imagen bastante parecida pero no necesariamente igual, ya que inevitablemente siempre van a haber cambios en el nuevo fotograma a procesar ya sea por el movimiento de los puntos a seguir, así como por los cambios de iluminación además del ruido en la imagen en el caso de grabaciones de poca calidad.

2.4.3.1 Optimización. Detección y corrección de errores.

Para lograr una mejor adaptabilidad del sistema a los cambios antes mencionados, el algoritmo también implementa un método para actualizar cada *template* encontrado correctamente e ir guardando una lista de los 5 últimos identificados con éxito en los 5 fotogramas anteriores. Con esto se logra, para la mayoría de los casos una adaptabilidad a los cambios de iluminación graduales, ya que el *template* a utilizar para cada fotograma es el actualizado con los datos del fotograma anterior. Además en caso de errores se puede volver a atrás 5 fotogramas en el tiempo, para poder recuperar el primero de los 5 últimos *templates* encontrados correctamente.

Debido a la flexibilidad del algoritmo de búsqueda, incluso en una grabación de un objeto estático donde los marcadores lógicamente no se mueven, el algoritmo tiende a ubicar el *template* dentro del área con el color incluyendo los bordes de la misma. En este caso al realizar la actualización del *template* se obtendría una muestra de una imagen con una región poco nítida donde se unen dos colore distintos, o sea esto sucede cada vez que el algoritmo localiza el *template* en el borde. Como consecuencia de esto el algoritmo suele perder la noción de cual color realmente está buscando dando erróneas ubicaciones que afectan el realismo del cálculo de la pose.

Para evitar esto se implementó una técnica para lograr que la actualización de los *templates* ocurra fuera del borde del marcador siempre que sea posible. Esto se logra haciendo que la muestra se actualice en la imagen en dirección a la parte más nítida de la de la propia muestra.

En todo momento el sistema va comprobando si han ocurrido errores esto se logra comparando la nueva posición del marcador con la posición que tenía en el fotograma anterior, teniendo en cuenta que la diferencia no debería ser demasiado significante. En caso de ser detectado un error la posición anterior se mantiene como actual hasta que aparezca una actualización correcta, entones el proceso continúa normalmente. Esta técnica le brinda a sistema incluso cierta estabilidad ante oclusión de marcadores.

2.4. Estimación de la pose.

Para la estimación de la pose es necesario haber inicializado los datos geométricos para que el algoritmo identifique a qué tipo de figura le va a hacer la aproximación en el caso de esta solución toda figura se va a tomar por defecto como un cubo, aunque el usuario en el momento de la inicialización puede indicarle al sistema las dimensiones de la figura o sea no es necesario el uso de una figura cúbica solo que marcadores estén ubicados en las esquinas del objeto.

Una vez encontrado los 4 puntos no coplanarios se procede al cálculo de la aproximación de la pose. El algoritmo Posit usado para resolver el problema de la estimación, tiene como salida la matriz de rotación y el vector de traslación, luego estos datos son encapsulados en una sola matriz (matriz de la pose), debido a que toda la implementación de esta solución está orientada a facilitar el proceso de dibujo en la escena virtual, por ese motivo además se crean otras matrices como la matriz de proyección y de escalado. Estas tres matrices son todo lo que se necesita para lograr una posicionar un objeto virtual con la misma posición y orientación que la del objeto real.

2.5 Dibujo de la escena virtual.

Los datos obtenidos en el paso anterior son interpretados por el usuario y usados para dibujar su escena virtual. Lo más común en la mayoría de los casos es cargar la matriz de proyección, luego multiplicar la matriz de proyección por la de escalado y por último multiplicar eso por la matriz de la pose. Encapsulando todo el cálculo geométrico anterior dentro de la matriz en la que se encuentra el objeto virtual (ver Anexo VInterax), se logran las transformaciones geométricas que se dan al objeto dibujado un comportamiento similar al objeto real. Pero los datos que ofrece la matriz de la pose pueden ser usados para ser interpretados de diferentes maneras, en dependencia de las necesidades del desarrollador, solo para dar una noción de lo que esto significa, se pudiera aclarar que una interpretación solo un poco más profunda de la matriz de la pose puede dar respuesta a cuestiones como: si el objeto está ubicado de frente o no, si está inclinado hacia la derecha o a la izquierda, hacia arriba o hacia abajo, en caso de necesitar un *tracking* de cabeza esta interpretación pudiera ser útil.

2.6 Consideraciones finales para el Capítulo II.

La solución propuesta para dar cumplimiento a los objetivos de esta investigación fue diseñada pensado fundamentalmente en cuestiones como: la estabilidad y rapidez del sistema a implementar, en ofrecer facilidades tanto para los programadores como para los usuarios finales en cuanto a la entrada de datos, además de la posibilidad de cambiar valores internos del sistema en aras de lograr una mayor adaptabilidad a las diferentes necesidades de los usuario. El uso de los valores que retorna el sistema está dirigido a aplicaciones de gráfico por computadora, no obstante los valores de traslación de los objetos en *tracking*, si pudieran ser usados con otras finalidades.

Capítulo III

En este capítulo se realiza la descripción de la solución propuesta a través de la metodología de desarrollo de software utilizada y los diagramas que propone UML. Se precisan las reglas del negocio y el modelo conceptual, se especifican los requisitos funcionales y no funcionales del prototipo funcional. Además se describen los procesos obtenidos a partir de los requerimientos funcionales.

3.1 Reglas del Negocio.

No.	Reglas
1	Para la visualización de la escena utilizará solo una cámara conectada a la computadora mediante USB, o un archivo de video de extensión ".avi", en caso de usar una imagen de archivo deberá tener formato ".jpg" o ".png".
2	El objeto al que se le pretende dar seguimiento deberá tener cuatro marcadores de colores diferentes y bien definidos.
3	Las condiciones de iluminación y la calidad de la imagen del video deberán permitir una buena diferenciación de los colores en la escena.
4	Los marcadores deberán indicar cada una de las dimensiones del objeto (centro, alto, ancho y profundidad), con colores diferentes para cada caso.
5	El tamaño de los marcadores deberá ser lo suficientemente grande como para que los mismos sean visibles para todos los posibles recorridos del objeto dentro de la escena, y lo suficientemente pequeño como para que ubiquen exactamente la posición de cada una de las esquinas del objeto.
6	Los marcadores deberán ser correctamente señalados por el usuario antes comenzar el seguimiento, especificando cuales dimensiones definen cada uno de los mismos.
7	Los movimientos del objeto a seguir deberán ser moderados.

Tabla 2: Reglas del negocio.

3.2 Modelo de Dominio

El objetivo principal del modelo de dominio es identificar los conceptos más importantes y relacionarlos con el propósito de comprenderlos, estos se representan en el siguiente diagrama:

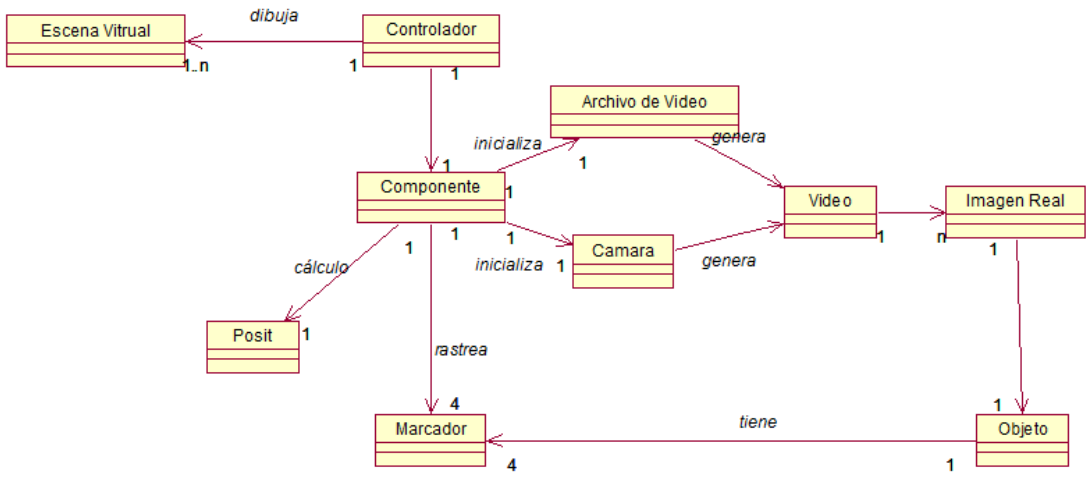

Figura 11: Modelo de Dominio.

Glosario de Términos del Dominio.

En aras de lograr un mayor entendimiento del modelo de dominio, a continuación se enuncia el significado de los conceptos identificados en este:

Video: Está compuesto por una secuencia de imágenes de la escena. Este es generado por la cámara, o por un archivo de video.

Objeto: Representa la figura a la cual se le va dar seguimiento. Contiene cuatro marcadores de colores.

Marcador: Objeto relativamente pequeño, de forma puntual ubicado en cada una de las esquinas del objeto.

Posit: Interfaz encargada de efectuar los cálculos de estimación de la pose.

Componente: Interfaz encargada de captar, procesar y devolver los datos del sistema.

Controlador: Aplicación de RM.

3.3 Captura de Requisitos

La captura de requisitos es una de las actividades del proceso de desarrollo de requisitos. En ésta actividad se identifican las capacidades, condiciones y restricciones que debe tener un software. A continuación se muestran los requisitos funcionales y no funcionales identificados.

3.3.1 Requisitos Funcionales.

RF1. Inicializar captura de video.

RF1.1 Configurar el tipo de entrada (para cámara, archivo de video, o imagen de archivo).

RF1.2 Nombrar la ventana en la que se visualiza en video.

RF1.3 Comenzar y detener la reproducción de video.

RF 2. *Parametrizar* marcadores.

RF2.1 Seleccionar el color a detectar, haciendo clic en una región de la imagen.

RF2.2 Escoger el tamaño del área de selección.

RF2.3 Mostar el área seleccionada en el momento de la selección.

RF2.4 Mostar en el video para cada marcador la región en la cual está siendo detectado.

RF3. Realizar el *tracking* del objeto.

RF3.1 Comenzar y detener el *tracking*.

RF3.2 Elegir si usar o no el sistema de detección de errores.

RF3.3 Estimar la pose del objeto a seguir.

RF3.4 Brindar la matriz de la pose, de proyección y de escalado del objeto a seguir.

RF4. Terminar la ejecución del sistema.

RF4.1 Liberar memoria.

RF4.2 Liberar recursos.

3.3.2 Requisitos No Funcionales.

Usabilidad: El componente implementado podrá ser utilizado por cualquier desarrollador de gráfico por computadora que esté interesado en hacer interactuar escenas virtuales en correspondencia con los movimientos de un objeto real.

Rendimiento: El sistema debe poseer alta velocidad de procesamiento o cálculo al ejecutarse para ofrecer los datos de estimación de la pose de manera que simulen una interacción en tiempo real.

Soporte: En su versión inicial deberá ser compatible con la plataforma Windows, pero debe estar preparado para migrar a GNU\Linux haciéndole pocas o ninguna modificación.

Hardware: Debe funcionar sobre microprocesador Intel Pentium (4) 2.8 GHz o superior, 256 MB RAM.

Software: La aplicación a desarrollar apoyándose en este componente se debe desarrollar en ANSI C++, puede utilizar cualquier motor de gráfico por computadora basado en OpenGL. Se hará uso de Programación Orientada a Objetos.

3.4 Casos de uso del sistema.

En el siguiente diagrama se pueden apreciar las diferentes acciones que un usuario en este caso un desarrollador puede realizar sobre el sistema.

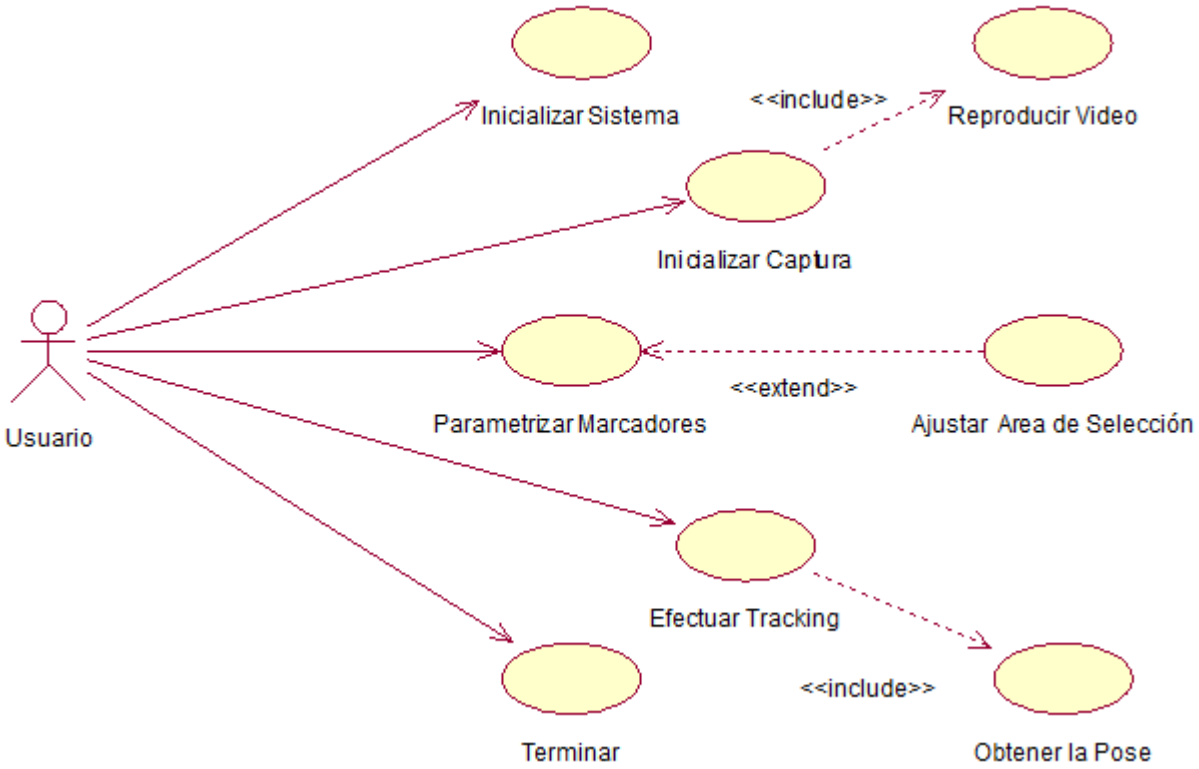

Figura 12: Diagrama de casos de uso del sistema.

3.5 Definición del actor del sistema.

El componente elaborado se diseñó para ser utilizado por desarrolladores de aplicaciones de RM pertenecientes a diferentes áreas de trabajo de la misma. Por ejemplo un programador de gráfico por computadora, solamente tendrá que preparar el objeto al que se le va a realizar el *tracking* Debido a esto se decidió nombrar al actor como "Desarrollador", el cual se representa en la tabla 3.

Actores	Justificación
Usuario	Es el que interactúa con el sistema, desarrolla su aplicación utilizando las funcionalidades del componente según sus propios propósitos. .

Tabla 3: Actor del Sistema

3.6 Expansión de casos de uso.

Caso de uso:	Inicializar Sistema.
Actor (es):	Usuario.
Propósito:	Iniciar el sistema. Preparar las condiciones para poder ejecutar todas las funcionalidades del sistema.
Resumen:	El CU comienza cuando el actor ejecuta el comando para que el sistema se inicialice. Luego de esto las variables internas están listas para responder a las necesidades funcionales del Usuario.
Referencias:	RF 1, RF 2,RF 3, RF 4
Pre-condiciones:	El componente está correctamente integrado a la aplicación.

Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Ejecuta el comando para iniciar el sistema.	1.1 Prepara las variables internas para el correcto funcionamiento del sistema.
Curso alternativo de los eventos.	
	1.1 Si el componente es ya está inicializado y recibe esta instrucción nuevamente, el sistema envía un mensaje de advertencia y continúa la ejecución del sistema ignorando la solicitud.
Post-condiciones:	Las variables internas están listas para responder a las necesidades funcionales del Usuario.
Prioridad	Crítico.

Tabla 4: Expansión del caso de uso Inicializar Sistema.

Caso de uso:	Inicializar Captura.
Actor (es):	Usuario.
Propósito:	Comenzar la captura de imágenes desde la fuente que se escoja.
Resumen:	El CU comienza cuando el actor ejecuta el comando para que el sistema inicialice un medio de captura. Y culmina mostrando la ventana donde se va a reproducir el video.
Referencias:	RF 1, RF 1.1, RF 1.2
Pre-condiciones:	La cámara está conectada a la PC. La dirección del archivo de video es correcta y la extensión del archivo es ".avi" o de la imagen es ".jpg"o".png".
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Ejecuta el comando para inicializar la captura de video	1.1 Prepara las condiciones para la lectura de las imágenes de la fuente seleccionada. 1.2 Inicializa una ventana para la reproducción del video.
Curso alternativo de los eventos.	
	1.1 Si la fuente seleccionada no está disponible para la captura de video, lanza un mensaje de error. 1.2 En caso de solicitar le captura desde un archivo se mostrara un mensaje de error en caso de que la extensión no sea ".avi" 2.1 Si existen varias cámaras el sistema muestra una ventana donde se pide que se especifique cual será la cámara de donde se va reproducir el video. Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	Se muestra la ventana de reproducción de video.
Prioridad	Crítico.

Tabla 5: Expansión del caso de uso Inicializar Captura.

Caso de uso:	Reproducir Video.
Actor (es):	Usuario.
Propósito:	Comenzar o detener la reproducción del video
Resumen:	El CU comienza cuando el actor ejecuta el comando para comenzar o detener el video. Y culmina mostrando en la ventana de reproducción el video a reproducir o la última imagen antes de detener el video.
Referencias:	RF 1,RF 1.3
Pre-condiciones:	El medio de captura esta inicializado
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Ejecuta el comando para reproducir el video. 2. Ejecuta en comando para detener el video.	1.1 Comienza a reproducir el video. 2.1 Detiene el video dejando en la ventana de reproducción la última imagen proyectada.
Curso alternativo de los eventos.	
	Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	Se reproduce o se detiene el video.
Prioridad:	Crítico.

Tabla 6: Expansión del caso de uso Reproducir Video.

Caso de uso:	*Parametrizar* Marcadores.
Actor (es):	Usuario.
Propósito:	Seleccionar para cada dimensión del objeto a seguir, el marcador de color que la va a definir.
Resumen:	El CU comienza cuando el actor ejecuta el comando para que el sistema reconozca los colores específicos para cada marcador. El sistema solo requiere de una especificación de la dimensión define el marcador a seleccionar, luego espera por un evento de clic sobre la ventana de reproducción de video. El color en la pequeña área alrededor del puntero del mouse será interpretado como la dimensión especificada.
Referencias:	RF 2, RF 2.1, RF 2.2, RF 2.3, RF 2.4
Pre-condiciones:	La ventana de reproducción de video está mostrando una imagen, o sea el video puede estar reproduciéndose o detenido.
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Hacer clic en la ventana de reproducción de video. (Automáticamente el sistema espera un evento de clic en la ventana de reproducción siempre que este mostrando una imagen.)	1.1 Espera por un evento de clic en la ventana de reproducción de video. 1.2 Una vez ocurrido el evento de clic, se actualizan las variables para reconocer en la imagen el color situado en el área de selección.
Curso alternativo de los eventos.	
	.

Post-condiciones:	El sistema conoce para cada una de las dimensiones del objeto cual es el color que la define.
Prioridad:	Crítico.

Tabla 7: Expansión del caso de uso *Parametrizar* Marcadores.

Caso de uso:	Ajustar Área de Selección.
Actor (es):	Usuario.
Propósito:	Aumentar o disminuir el área de selección alrededor del puntero del mouse, con el objetivo de incrementar la precisión del algoritmo.
Resumen:	El CU comienza cuando el actor ejecuta el comando para que el sistema actualice las variables que define el área a seleccionar por el usuario final.
Referencias:	RF 2, RF 2.1, RF 2.2, RF 2.3, RF 2.4
Pre-condiciones:	El sistema debe estar iniciado.
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Ejecuta el comando ajustar el tamaño del área de selección, escogiendo el tamaño deseado.	1.1 El sistema actualiza las variables internas para modificar el área de selección.
Curso alternativo de los eventos.	

	Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	El sistema área de selección es modificada.
Prioridad:	Opcional.

Tabla 8: Expansión del caso de uso Ajustar Área de Selección.

Caso de uso:	Efectuar *Tracking*.
Actor (es):	Usuario.
Propósito:	Ejecutar todo el proceso de estimación de la pose, para obtener los datos de la ubicación del objeto real.
Resumen:	El CU comienza cuando el actor ejecuta el comando para que el sistema comience con el *tracking* del objeto en la escena. Luego el sistema realiza la estimación y actualiza para cada instante de tiempo la nueva posición del objeto.
Referencias:	RF 3, RF 3.1, RF 3.2, RF 3.3, RF 3.4
Pre-condiciones:	El sistema ha recibido los datos que indican para cada una de las dimensiones del objeto el color la define.
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
2. Ejecuta el comando para comenzar el *tracking*. 3. Ejecuta el comando para detener el *tracking*.	1.1 Comienza a efectuar el *tracking* con o sin detección de errores, e dependencia del tipo de instrucción dada. 1.2 Actualiza internamente el valor de la matriz de la pose del objeto para cada fotograma de video. 1.3 Muestra en la ventana de reproducción las nuevas

	ubicaciones donde se están detectando los marcadores. 2.1 Detiene la ejecución del algoritmo de detección y *tracking*.
Curso alternativo de los eventos.	
	Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	El sistema conoce la posición y orientación del objeto en la escena.
Prioridad:	Crítico.

Tabla 9: Expansión del caso de uso Efectuar *Tracking*.

Caso de uso:	Obtener la Pose.
Actor (es):	Usuario.
Propósito:	Obtener las matrices necesarias para dibujar en la escena virtual.
Resumen:	El CU comienza cuando el actor ejecuta los comandos para obtener las distintas matrices de proyección para poder dibujar en la escena virtual. El sistema procede a devolver los datos solicitados.
Referencias:	RF 3.4
Pre-condiciones:	El *tracking* se debe estar ejecutando.
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**

1. Ejecuta el comando para obtener la matriz de proyección. 2. Ejecuta el comando para obtener la matriz de escalado. 3. Ejecuta el comando para obtener la matriz de la pose.	1.1 El sistema retorna la matriz de proyección. 2.1 El sistema retorna la matriz de escalado. 3.1 El sistema retorna la matriz de la pose.
Curso alternativo de los eventos.	
	Si no se está ejecutando el *tracking* el sistema retorna para cada una de las matrices solicitadas, la matriz identidad. Para no afectar innecesariamente las proyecciones en la escena virtual. Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	El sistema devuelve todas las variables necesarias para que el usuario modifique su escena.
Prioridad:	Crítico.

Tabla 10: Expansión del caso de uso Obtener la Pose.

Caso de uso:	Terminar.
Actor (es):	Usuario.
Propósito:	Liberar la memoria y los recursos usados por sistema.
Resumen:	El CU comienza cuando el actor ejecuta los comandos para liberar los recursos y la memoria en uso por el sistema.
Referencias:	RF 4, RF 4.1,RF 4.2

Pre-condiciones:	El sistema debe estar iniciado. El sistema no está ejecutando sus principales funcionalidades.
Curso normal de los eventos.	
Acciones del actor	**Respuesta del Sistema**
1. Ejecuta el comando para liberar recursos y memoria.	1.1 El sistema libera la memoria y los recursos que tenga en uso.
Curso alternativo de los eventos.	
	Si el sistema se encuentra ejecutando el *tracking* en el instante en que se da la instrucción de liberar recursos, se lanza un mensaje de error. Si el sistema no está inicializado se lanza un mensaje indicando este error.
Post-condiciones:	El sistema libera recursos y memoria. El sistema no podrá responder a ninguna otra acción del usuario.
Prioridad:	Secundario.

Tabla 11: Expansión del caso de uso Terminar.

3.7 Consideraciones finales del capítulo III.

En este capítulo se describió lo que el sistema debe ser capaz de hacer. Para ello se modelaron los principales conceptos, se establecieron las capacidades o funcionalidades que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener, es decir, los requisitos funcionales y no funcionales. Se agruparon los requisitos funcionales en casos de uso y estos últimos se describieron para un mejor entendimiento de los procesos que tendrán lugar en el producto desarrollado.

CAPÍTULO IV

En este capítulo se tratan los aspectos fundamentales referentes al diseño e implementación de la solución para la elaboración del prototipo funcional.

4.1 Diagrama de clases de diseño.

En la figura siguiente se muestra como quedó el diagrama de clases del diseño el cual resultó ser el punto de partida para comenzar a elaborar la solución en términos del lenguaje de programación seleccionado. Para restar un poco de complejidad al diagrama se decidió tener en cuenta solamente los atributos de las clases. Una descripción un poco más detallada de las clases que resultaron ser más importantes para la solución.

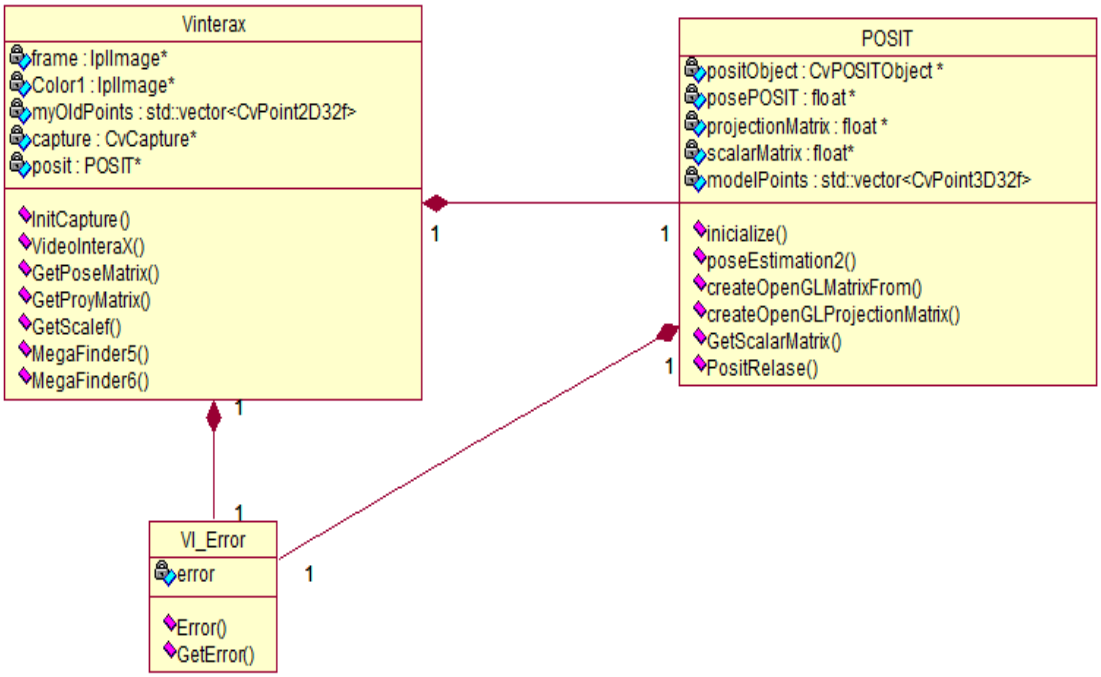

Figura 13: Diagrama de clases del diseño.

4.2 Descripción de clases de diseño

A continuación se proporciona una descripción de los atributos y métodos más significativos de algunas clases que se consideran importantes para la elaboración de la solución.

Nombre de la clase: VInterax	
Tipo de clase: Controladora.	
Atributo:	Tipo:
frame	IplImage*
Color1	IplImage*
myOldPoints	std::vector<CvPoint2D32f>
capture	CvCapture*
Posit	POSIT*
Responsabilidades de la clase:	
Nombre:	InitCapture()
Descripción:	Inicializa las variables y levanta una ventana para la reproducción del video.
Nombre:	VideoInteraX()
Descripción:	Controla la reproducción del video y la detección de los marcadores en la imagen.
Nombre:	MegaFinder5()
Descripción:	Dados cuatro colores específicos detecta su ubicación en una imagen dada. Detecta posibles errores, y siempre que sea posible los corrige o mitiga.

Nombre:	TheProcess ()
Descripción:	Dado un color específico detecta su ubicación en una imagen dada. Ejecuta las transformaciones pertinentes de los datos obtenidos para su correcta interpretación. Controla la detección y la recuperación ante errores.
Nombre:	Verifyer ()
Descripción:	Dada la ubicación de un color en la imagen verifica si pudiera ser errónea o no basado en la última posición correcta encontrada para ese color y su posición con respecto a los otros colores encontrados.
Nombre:	MegaFinder6()
Descripción:	Dados cuatro colores específicos detecta su ubicación en una imagen dada. No trabaja sobre la detección de errores.

Tabla 12: Descripción de la clase VInterax.

Nombre de la clase: Posit	
Tipo de clase: Controladora.	
Atributo:	Tipo:
posePOSIT	float*
projectionMatrix	float*
scalarMatrix	float*
IdentityMatrix	float*
positObject	CvPOSITObject*
modelPoints	std::vector<CvPoint3D32f>

Responsabilidades de la clase:	
Nombre:	GetPose ()
Descripción:	Retorna la matriz de la pose. En caso de no encontrarse el sistema ejecutando el *tracking*, se retorna la matriz de identidad.
Nombre:	GetProjectionMatrix ()
Descripción:	Retorna la matriz de proyección. En caso de no encontrarse el sistema ejecutando el *tracking*, se retorna la matriz de identidad.
Nombre:	GetScalarMatrix ()
Descripción:	Retorna la matriz de escalado. En caso de no encontrarse el sistema ejecutando el *tracking*, se retorna la matriz de identidad.
Nombre:	poseEstimation2 ()
Descripción:	Dada una lista de puntos no coplanarios construye la matriz de rotación y el vector de traslación.
Nombre:	createOpenGLMatrixFrom ()
Descripción:	Dada la matriz y de rotación y el vector de traslación construye la matriz de la pose.
Nombre:	initialize ()
Descripción:	Inicializa el objeto Posit, la matriz de proyección y de escalado.

Tabla 13: Descripción de la clase Posit.

Nombre de la clase: VI_Error	
Tipo de clase: Entidad.	
Atributo:	Tipo:
Error	char*
Responsabilidades de la clase:	
Nombre:	VI_Error ()
Descripción:	Inicializa el mensaje de error.
Nombre:	GetError ()
Descripción:	Devuelve el mensaje de error previamente inicializado.

Tabla 14: Descripción de la clase VI_Error.

4.3 Diagramas de secuencia.

En este epígrafe se presentan los diagramas de secuencia correspondientes a los casos de uso descritos anteriormente en el capítulo 3. Dichos diagramas representan los eventos generados por el actor, su orden y los eventos internos del sistema. Para lograr claridad en los diagramas se detallaron los eventos de mayor importancia.

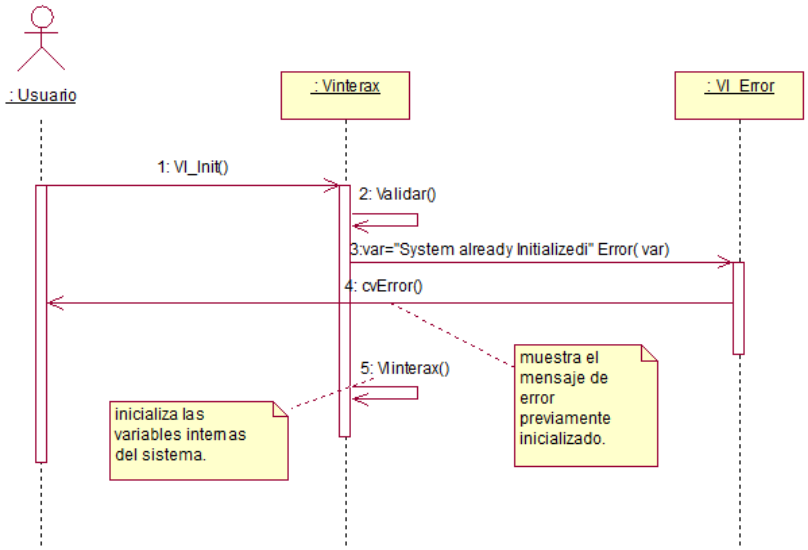

Figura 14: Diagrama de secuencia del diseño. Inicializar Sistema.

La figura 13 muestra el diagrama de secuencia para el caso de uso Inicializar Sistema. Durante la ejecución de este caso de uso, el actor ejecuta la inicialización del sistema, se verifica que el sistema no esté previamente inicializado, en caso de ser así se muestra un mensaje de error, en caso contrario se inicializan las variables internas del sistema.

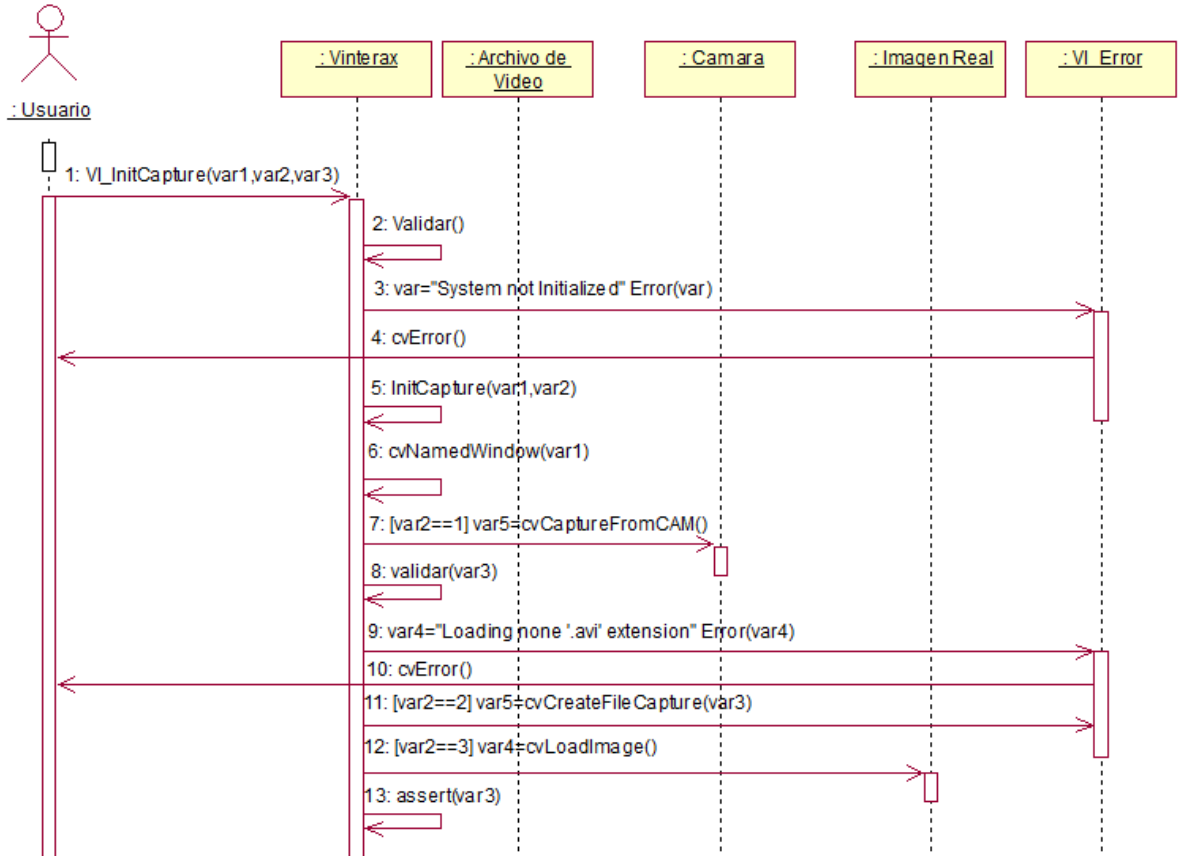

Figura 15: Diagrama de secuencia del diseño. Inicializar Captura.

La figura 14 muestra la secuencia del caso de uso Inicializar Captura donde se habilita al sistema para posteriormente poder reproducir un video desde la fuente especificada en una ventana que el mismo sistema crea. El sistema muestra mensajes de error en los casos en que el sistema no esté previamente inicializado, o se brinde una dirección errónea del archivo fuente o la cámara no esté disponible. También se lanzan excepciones para la entrada de archivos de formatos no aceptables.

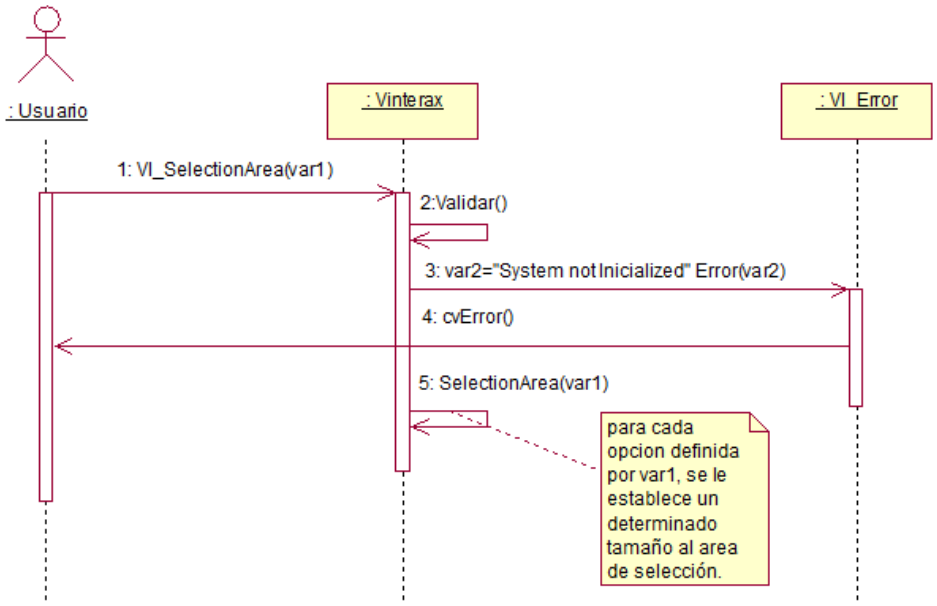

Figura 16: Diagrama de secuencia del diseño: Ajustar Área de Selección.

Figura 15 diagrama de secuencia del diseño para el caso de uso Ajustar Área de Selección. El sistema deberá enviar un mensaje de error en caso de que se trate de invocar esta funcionalidad antes de inicializar el sistema, de no ser así se establece el área de selección solicitada por el usuario.

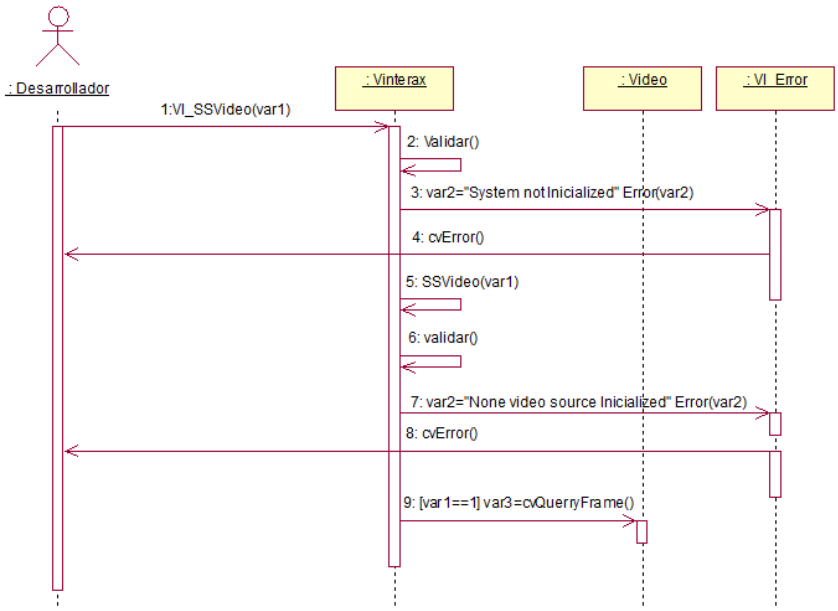

Figura 17: Diagrama de secuencia del diseño: Reproducir Video.

En la figura 16 se muestra el diagrama de secuencia para el caso de uso Reproducir Video, en donde el sistema comienza o detiene la reproducción del video en cuestión en dependencia de la especificación del usuario. Se lanzan mensajes de error en los casos en que la instrucción sea dada sin haber inicializado el sistema o se trate de reproducir un video sin haber inicializado previamente la fuente de captura.

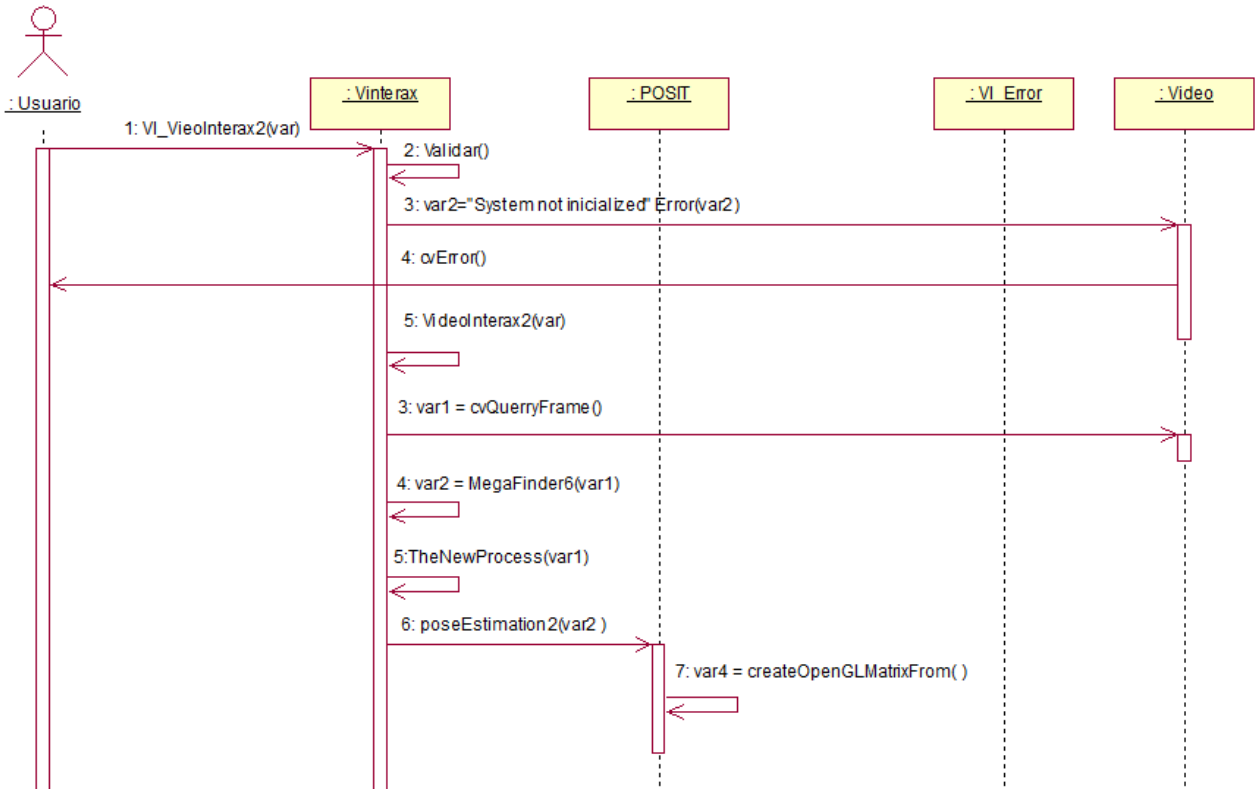

Figura 18: Diagrama de secuencia del diseño para el caso de uso Efectuar *Tracking*.

En la figura 17 aparece el diagrama de secuencia para el caso de uso Efectuar *Tracking*. Para este caso de uso, la secuencia de pasos del sistema comienza con una validación donde se lanza un error en caso de una llamada incorrecta al método responsable del caso de uso, de lo contrario el sistema ejecuta internamente varias funcionalidades.

Primeramente ejecuta una instrucción para obtener el fotograma que se esté reproduciendo en el video, luego esta imagen es usada por otra funcionalidad la cual detecta en la misma los cuatro puntos no coplanarios, haciendo cuatro llamadas (una por cada punto) al procedimiento encargado de encontrar un punto determinado en la porción de la imagen cercana a donde se detectó por última vez.

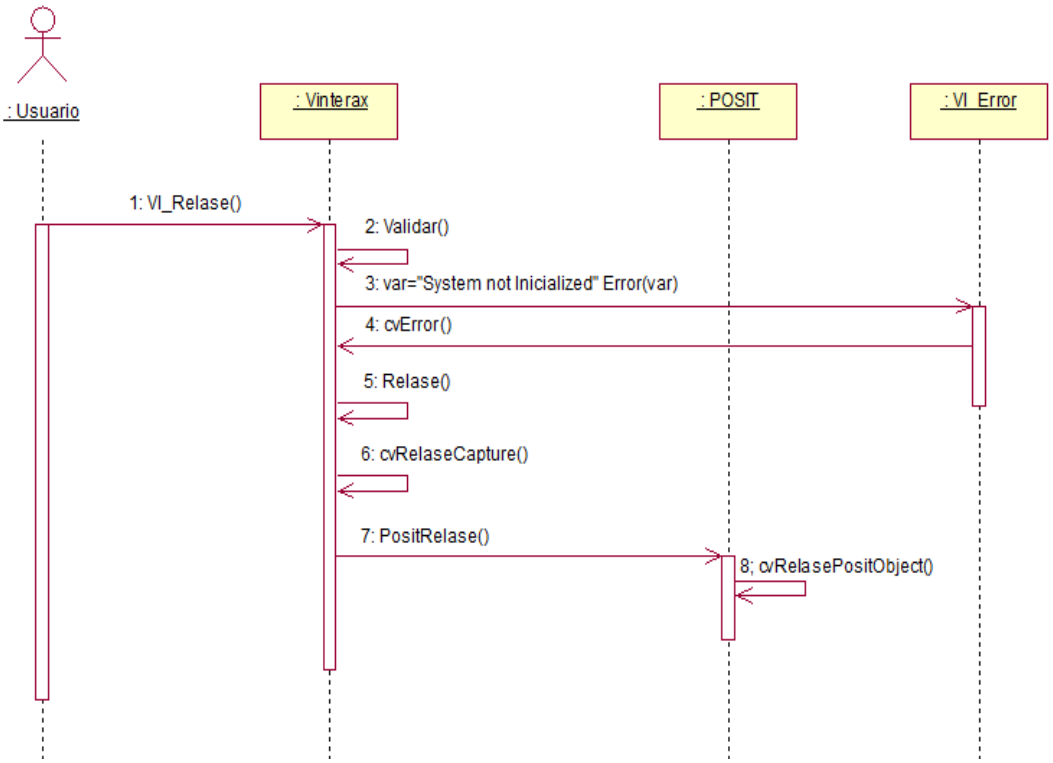

Figura 19: Diagrama de secuencia para el caso de uso Terminar.

La figura 18 muestra el diagrama de secuencia del caso de uso Terminar. Donde las primeras acciones corresponden a la validación de la invocación de dicha funcionalidad, luego continua con la liberación de los recursos usados por el sistema, ya sea un archivo de video o una cámara, también se libera el espacio en memoria para las principales variables.

4.4 Diagrama de Componente.

En la figura tal se presentan los componentes utilizados para la elaboración de la aplicación se agrupados en paquetes. El paquete VInterax es el que engloba los componentes definidos para la elaboración de la solución, en el paquete OpenCV están los componentes específicos de dicha biblioteca

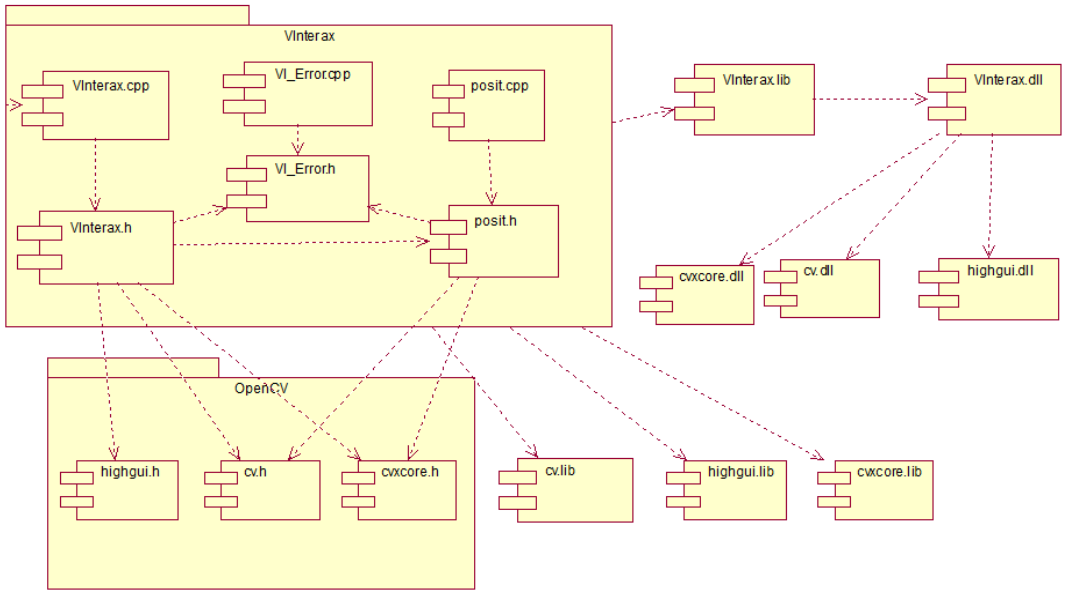

Figura 20: Diagrama de Componente.

4.5 Diagrama de despliegue.

En la figura tal se muestra el diagrama de despliegue. Aunque este es un diagrama sencillo se decidió incluirlo para enfatizar en el uso de una cámara conectada a la computadora para el correcto funcionamiento del producto elaborado, que aunque el mismo puede funcionar además con archivos de imagen y de video, está diseñado principalmente para ser usado con una cámara.

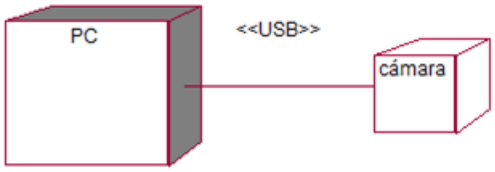

Figura 21: Diagrama de despliegue.

4.6 Validación de los resultados.

Para la validación de los resultados se realizaron varias pruebas a las principales funcionalidades del componente, con los objetivos de verificar el comportamiento del sistema en cuanto a velocidad de respuesta y estabilidad, para las distintitas condiciones de uso, fuentes de captura de imágenes y opciones de tracking que ofrece el componente. Los resultados de estas pruebas aparecen en una serie de tablas a continuación, donde se relacionan varias características del sistema entre sí, y en distintas situaciones

Para lograr un mayor entendimiento de las tablas a continuación se especifica lo siguiente:

Como variables independientes se tomarán las siguientes:

Fuentes de captura: Estas pueden ser: por webcam, por archivo de video o por archivo de imagen.

Opciones de tracking: las que brinda el componente son: con detección de errores o sin detección de errores.

Condiciones de uso: se refiere a las condiciones de iluminación de la escena en la que esté ubicado el objeto en seguimiento. Otros factores correspondientes a la calidad de la filmación tales como ruido, nitidez y contraste de colores en la imagen son también tomados en cuenta en este aspecto.

Clasificación:

Buena: La iluminación de la escena y la calidad de la filmación brindan imágenes nítidas con muy poco ruido y colores bien diferenciables.

Media: Las imágenes proporcionadas por la captura de video permite una diferenciación de colores leve, las imágenes tiene algo de ruido.

Mala: La iluminación de la escena, el ruido en las imágenes y la poca nitidez hacen que sea difícil a simple vista distinguir los colores en la escena.

Como variables dependientes se tomarán las siguientes:

Velocidad de Respuesta: se refiere al tiempo en que el sistema responde a las solicitudes de cálculo o estimación. Este tipo de solicitudes son constantes o sea se ejecutan para cada fotograma del video.

Clasificación:

Rápido: Para un video reproduciéndose a 24 *frames** por segundo el sistema no deberá reducir la velocidad de reproducción a menos de 20 *frames* por segundo.

Normal: Para un video reproduciéndose a 24 *frames* por segundo el sistema tiende a reducir la velocidad de reproducción en un rango entre los 12 y los 19.

Lento: Para un video reproduciéndose a 24 *frames* por segundo el sistema tiende a reducir la velocidad de reproducción a menos de 12 *frames* por segundo.

Estabilidad: se refiere a la capacidad del sistema para la detección de los marcadores de la cual depende una estimación correcta o errónea de la posición y orientación del objeto en seguimiento, durante todo el proceso de *tracking*.

Clasificación:

Alta: El sistema detecta los colores con facilidad y existe una completa concordancia entre la posición y orientación del objeto en la escena con la del objeto dibujado virtualmente.

Media: El sistema en ocasiones, se ve en la necesidad de utilizar los métodos de corrección de errores. Por momentos la concordancia entre objeto real y el objeto virtual se ve afectada.

Baja: El sistema es inestable, la detección de marcadores brida valores erróneos. La concordancia entre el objeto real y el objeto virtual es imperceptible.

Tablas de validación de resultados para un *tracking* sin detección de errores.

Captura de imágenes mediante webcam.		
Condición de uso	Velocidad de Respuesta	Estabilidad.
Buena	Normal	Alta
Media	Normal	Media.
Mala	Normal	Baja

Tabla 15: Validación de resultados para un *tracking* sin detección de errores. Captura de imágenes mediante webcam.

Captura de imágenes mediante archivo de video.		
Condición de uso	Velocidad de Respuesta	Estabilidad.
Buena	Rápido	Alta
Media	Rápido	Media.
Mala	Rápido	Baja

Tabla 16: Validación de resultados para un *tracking* sin detección de errores. Captura de imágenes mediante archivo de video.

Imagen archivo.		
Condición de uso	Velocidad de Respuesta	Estabilidad.
Buena	Rápido	Alta
Media	Rápido	Alta
Mala	Rápido	Alta.

Tabla 17: Validación de resultados para un *tracking* sin detección de errores. Captura de imágenes mediante imagen de archivo.

Tablas de validación de resultados para un *tracking* con detección de errores.

Captura de imágenes mediante webcam.		
Condición de uso	Velocidad de Respuesta	Estabilidad.
Buena	Media	Alta
Media	Media	Alta
Mala	Baja	Media

Tabla 18: Validación de resultados para un *tracking* con detección de errores. Captura de imágenes mediante webcam.

Captura de imágenes mediante archivo de video.		
Condición de uso	Velocidad de Respuesta	Estabilidad
Buena	Normal	Alta
Media	Normal	Alta
Mala	Lento	Media

Tabla 19: Validación de resultados para un *tracking* con detección de errores. Captura de imágenes mediante archivo de video.

Imagen archivo.		
Condición de uso	Velocidad de Respuesta	Estabilidad.
Buena	Rápido	Alta
Media	Rápido	Alta
Mala	Rápido	Alta.

Tabla 20: Validación de resultados para un *tracking* con detección de errores. Captura de imágenes mediante imagen de archivo.

La validación de resultados mostró que la solución propuesta brinda, para la mayoría de los casos, una resolución rápida y estable al problema del *tracking* de objetos. Reflejó además, la influencia de las variables independientes sobre las variables dependientes, teniéndose a manera de resumen de las anteriores tablas que:

- Una fuente de captura mediante webcam tiene un efecto reductor en la velocidad de respuesta del sistema (debido al tiempo de captura de imágenes desde un dispositivo USB). Las demás fuentes de captura ofrecen tiempos de respuesta muy inferiores con respecto a la webcam. No obstante, para ninguno de los casos, se deja de cumplir, con los requisitos necesarios para que la respuesta sea considerada “en tiempo real interactivo”.
- La detección de errores tiende a reducir la velocidad del sistema a la vez que le brinda mayor estabilidad, la no detección de errores ofrece alta velocidad de respuesta aunque no garantiza la estabilidad del sistema.
- Las condiciones de uso tienen un peso fundamental en la ejecución del sistema, ya que su variación afecta tanto a la variable velocidad como a la variable estabilidad. Bajo condiciones de uso óptimas (entiéndase como óptimas: buena calidad del video de acuerdo con los criterios mencionados anteriormente), el sistema desarrollado puede llegar a alcanzar una alta velocidad y estabilidad.

Por todo lo antes expuesto, se considera que el componente desarrollado cumple con los requisitos para ser usado en aplicaciones de RM en tiempo real.

CONCLUSIONES

El proceso de realización de esta investigación permitió:

- Seleccionar el algoritmo Posit para resolver el problema de la pose de un objeto, y el *tracking* óptico por marcadores de colores como técnica para darle seguimiento a cuatro puntos en una secuencia de imágenes.
- La implementación de un componente que ofrece las funcionalidades necesarias para la estimación de la posición y orientación de un objeto en una escena del mundo real, integrable a cualquier aplicación de gráfico por computadora.
- La apertura de nuevas posibilidades, a través del componente desarrollado para la creación de aplicaciones de RM, fuera de las que ofrece ARToolkit
- La creación de una aplicación prototipo, la cual combina las funcionalidades de OpenGL con las que ofrece el componente desarrollado, para lograr una interacción en tiempo real de un usuario con un objeto en un ambiente virtual, validándose de esta forma el cumplimiento de los objetivos propuestos.

RECOMENDACIONES

Tras haberse integrado con éxito el componente desarrollado en esta investigación a una aplicación sencilla a manera demostrativa, se recomienda la utilización del mismo en la creación de aplicaciones de RM de mayor envergadura, ya sea en Video Juegos, simulaciones, o *tracking* de cabeza. Se recomienda además:

- Migrar la solución propuesta a la plataforma libre GNU\Linux.
- Realizar investigaciones para estudiar la posibilidad del tracking multi-objetos, a partir de la solución brindada.

REFERENCIAS BIBLIOGRÁFICAS

1. **Pablo, Figueroa.** [Online] abril 1, 2008. http://imagine.uniandes.edu.co/drupal/?q=node/28.

2. **Avila, Diego.** Blogger.com. *realidad--virtual.blogspot.com.* [Online] junio 24, 2008. http://realidad--virtual.blogspot.com/2008/06/que-es-realidad-virtual.html.

3. **Bernabé Parrilla, Eduardo.** Tesis.com.es. *Tesis.com.es.* [Online] abril 29, 2010. http://tesis.com.es/tribunales/marques-acosta-ferran/documentos/analisis-movimiento-basado-algoritmos-flujo-optico-estereoscopia/.

4. **Moreno Mirabal, Mileydi and de la Cruz Guevara, Ernesto.** *Localización de objetos virtuales en el mundo real con técnicas de Realidad Aumentada.* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

5. **Lourakis, Antonis A. Argyros, Manolis I.A.** *Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera.* Heraklion, Crete : Institute of Computer Science, 2006.

6. **Prietoa, Carlos E., et al.** www.buenastareas.com. *www.buenastareas.com.* [Online] octubre 7, 2010. http://www.buenastareas.com/ensayos/Sistema-De-Vision-Artificial-Para-El/867687.html.

7. **Mery, Domingo.** *Vision Artificial.* Santiago de Chile : Universidad de Santiago de Chile, 2002.

8. **M. E. Ragab, K. H. Wong.** *MULTIPLE NONOVERLAPPING CAMERA POSE ESTIMATION.* Hong Kong : Proceedings of 2010 IEEE 17th International Conference on Image Processing, 2010.

9. *Pose Estimation based on Four Coplanar Point Correspondences.* **Zhen Zhang, Qixin Cao, Yang Yang y Charles Lo.** Shanghai : Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009.

10. **Pérez, Carlos and Reinoso, Oscar.** *Seguimiento visual de objetos mediante el control de los parámetros de una cámara motorizada.* Almería : Universidad de Almería, 2003.

11. **Haralick, R. M.** *Performance Characterization in Computer Vision.* Washington : University of Washington C.S. Technical, 1991.

12. **Fischler, M. A., R. C, Bolles.** s.l. : Comm ACM, 1981.

13. *Linear N-Point Camera Pose Determination.* **Long Quan, Zhongdan Lan.** Dubai : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999.

14. **Daniel F. DeMenthon, Larry S. Davis.** *Model-Based Object Pose in25 Lines of Code.* Washington DC : Computer Vision Laboratory. University of Maryland, 1996.

15. **H. Álvarez, D. Borro.** *Cálculo de la Pose de la Cámara ante Oclusiones de un Marcador.* Barcelona : University of Navarra, 2008.

16. **Gary Bradski, Adrian Kaebler.** *Learnig OpenCV.* s.l. : O'Reilly Media, Inc., 2008.

17. *Open Source Computer Vision Library Reference Manual.* s.l. : Intel Corporation, 2001.

18. **Beatriz Martín Guadaño, Ana Melcón Sanjuán, Daniel Tapia Manganero.** *Identificación óptica de la posición y orientación de un Vehículo Aéreo no Tripulado.* Madrid : Universidad Complutense de Madrid, 2009.

19. **M. A. Vicente, C. Fernandez, R. Puerto.** *Extracción de características para el reconicimiento visual.* Elche : Universidad Miguel Hernández, 2004.

20. **Rafael Berenguer Vidal, Rafael Verdú Monedero.** *SISTEMA DE VISIÓN POR COMPUTADOR PARA TRACKING AUTOMÁTICO Y CARACTERIZACIÓN DE OBJETOS EN 3D.* Cartagena : Universidad Politécnica de Cartagena, 2002.

21. *UN MODELO DIFUSO PARA FOCALIZACIÓN EN UN SISTEMA DE VISIÓN ACTIVA.* **Homero Latorre, Juan Luis Castro, José Manuel Benítez, Miguel García-Silvente.** Granada : XII CONGRESO ESPAÑOL SOBRE TECNOLOGÍAS Y LÓGICA FUZZY, 2001.

22. *Sistema Neuronal de Bajo Costo para la Identificación de Colores.* **Jaime Enrique Gómez Santana, Jaime Mauricio Peñaloza Trespalacios, Ing. José de Jesús Rugeles Uribe, Dr. Eduardo Francisco Caicedo Bravo.** Cali-Colombia : X Simposio de Señales, Imágenes y Visión Artificial, 2003.

23. *DETECCIÓN AUTOMÁTICA DEL COLOR DE LA PIEL EN IMÁGENES BIDIMENSIONALES BASADO EN EL ANÁLISIS DE REGIONES.* **Benito, Darío de Miguel.** Madrid : Universidad rey Juan Carlos, 2005.

24. **FERNÁNDEZ, SARA MIRA.** *PIZARRA VIRTUAL BASADA EN REALIDAD AUMENTADA.* MADRID : UNIVERSIDAD PONTIFICIA COMILLAS, 2009.

25. *A new image segmentation method based on HSI color space for biped soccer robot.* **Honger, Q. Zhong y R.** Xiamen : IEEE International Symposium on IT in Medicine and Education., 2008.

26. *Introducción a los sistemas de visión y de colores.* **Loaiza, Humberto.** Cali : Revista Energía y computación. Vol. VIII, Nº 1, Universidad del valle, 1999.

27. *RECONOCIMIENTO DE IMÁGENES A TRAVÉS DE SU CONTENIDO.* **Sáez Peralta, Antonio.** MADRID : UNIVERSIDAD PONTIFICIA COMILLAS, 2009.

28. *Review and analysis of solutions of the three point perspective pose estimation problem.* **Robert M. Haralick, Chung-nan Lee, Karsten Ottenberg, Michael Nolle.** Toronto : International Journal of Computer Vision, 1994.

29. **Diego Aracena Pizarro, Pedro Campos, Clésio Luis Tozzi.** *COMPARACIÓN DE TÉCNICAS DE CALIBRACIÓN DE CÁMARAS DIGITALES.* Mexico : Univ. Tarapacá, 2005.

30. *An Efficient Solution to the Five-Point Relative Pose Problem.* **David, Nister.** s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

31. *Multiple View Geometry in Computer Vision.* **Hartley, Richard I. and Zisserman, A.** s.l. : Cambridge Univ., 2000.

32. *EGT: a Toolbox for Multiple View Geometry and Visual Servoing.* **G.L. Mariottini, D. Prattichizzo.** s.l. : IEEE Robotics and Automation Magazine, 2005.

33. *Pose Determination of 3D Object Based on Four Straight Lines.* **Lijuan, Qin, et al.** Jinan : Fourth International Conference on Natural Computation, 2008.

34. *Markerless pose tracking for augmented reality.* **Yuan, C.** Manchester City : Advances in Visual Computing. Second International Symposium, 2006.

35. *A tracking by detection approach for robust markerless tracking.* **Yuan, C.** Minsk : Industrial Augmented Reality Workshop; at the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2006.

36. webdocs.cs.ualberta.ca. *webdocs.cs.ualberta.ca.* [Online] March 22, 2003. http://webdocs.cs.ualberta.ca/~vis/VR2003tut/.

ANEXOS

- VInterax

- Interacción de una escena virtual con un objeto del mundo real.

Figura 22: Usuario de una aplicación de RM. La cámara capta los movimientos del objeto que sostiene el usuario. El avión dibujado en la escena virtual se mueve según los movimientos del objeto.

- Interacción de una escena virtual con un objeto del mundo real.

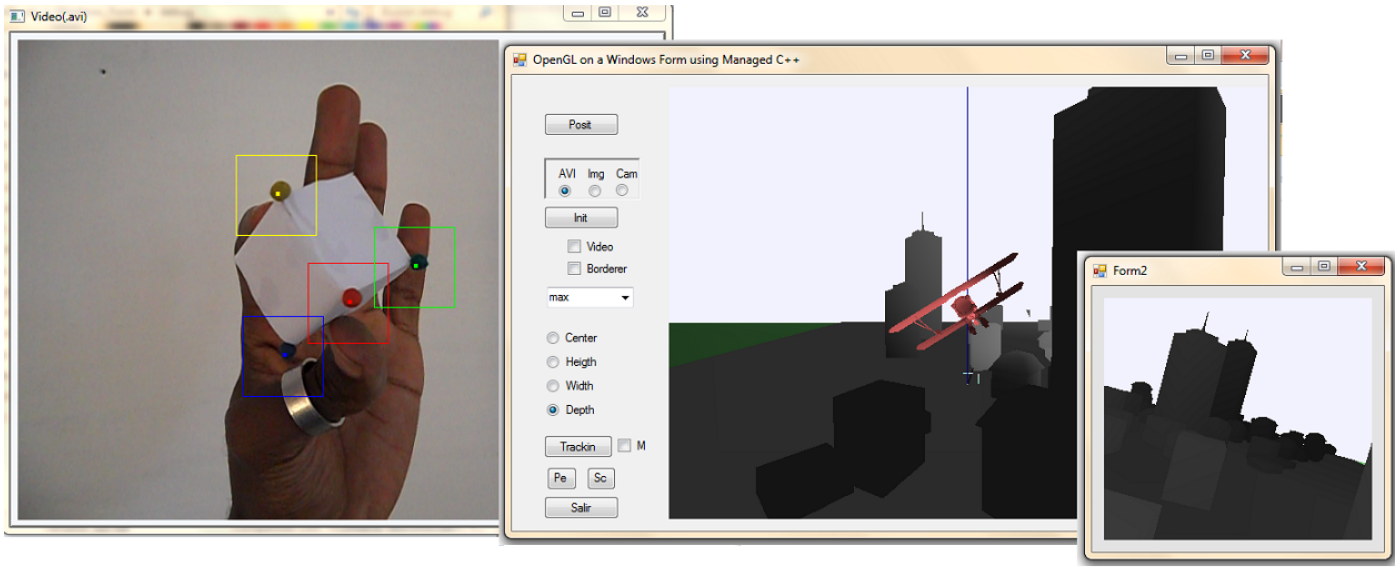

Figura 23: Usuario de una aplicación de RM (Vista de la pantalla del ordenador).

- En la ventana de la izquierda se muestra la reproducción de un video, en donde aparece el objeto al que se le efectúa el *tracking.* En la ventana del centro aparece una escena virtual en donde un objeto en este caso un avión, ejecuta los mismos movimientos del objeto en el video. En la ventana de la derecha aparece una representación de la vista perspectiva desde el avión.

GLOSARIO DE TÉRMINOS.

Frame: Fotograma. Referente a la imagen que se obtiene de un video para cada instante de tiempo.

Matching: Referente a la técnica o proceso de hacer coincidir dos valores

Parametrización: Acción de entrar datos o de especificar nuevos valores, para un algoritmo,

Template: Porción de una imagen, usada para hacerla coincidir en otra imagen

Tracking: Rastreo o seguimiento.