Universidad de las Ciencias Informáticas

Facultad 5

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

Título: Componente de software para Realidad Aumentada, basado en la detección de características naturales en imágenes.

Autor: Joaquin Alberto Díaz Vázquez

Tutor: Ing. Ernesto de la Cruz Guevara Ramírez

Co-tutor: Ing. Mileydi Moreno Mirabal

"La Habana, Junio del 2014"

“La gente piensa que enfocarse significa decir sí a aquello en lo que te enfocas, pero no es así. Significa decir no a otras cientos de ideas buenas que hay.”

Steve Jobs

Dedicatoria

A mis abuelos, a mi madre y a quien fue más que un padrastro: a ti Norgito.

A mis tías y a toda mi familia, por confiar en mí y darme todo el apoyo del mundo. Este triunfo también es de ellos.

A mis hermanos, para que les sirva de ejemplo y sean mejor que yo.

Agradecimientos

A mi abuela Nancy, por su entrega y dedicación, por saberme guiar por el buen camino, por sus deseos de convertirme en un hombre de bien, por no rendirse nunca y ayudarme a levantar la cabeza cuando más lo necesité.

A mi abuelo Joaquín, por ser un ejemplo de firmeza, por ser el primer padre que conocí.

A mi madre, por ser la heroína que me trajo al mundo, por estar a mi lado en las buenas y en las malas, por su inmenso sacrificio, por saber ser madre y padre cuando más lo necesité en mi vida.

A ti Norgito, saber convertirte en mi padre, en el hombre que siempre quise imitar. Aunque la distancia nos separe, para mí siempre serás mi papá, al que le debo gran parte de lo que soy.

A mis tías Rahirca y Nacita, que siempre me han querido como a un hijo, por consentirme en todo, porque han sido un ejemplo para mí.

A mis amigos, a mis verdaderos amigos, los que me han criticado cuando he fallado, a los que me han dado la mano cuando he caído, a los que han estado en las buenas y en las malas. Gracias

A mis tutores, porque sacrificaron su tiempo y fueron más que mis guías. A ellos les agradezco por ayudarme a convertir en verdad este sueño.

Gracias a todos.

Declaración de autoría

Declaro ser el 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 _____.

Ernesto de la Cruz Guevara

_____

Firma del Tutor

Mileydi Moreno Mirabal

_____

Firma del Co-Tutor

Joaquín Alberto Díaz Vázquez

_____

Firma del Autor

Datos de contacto

Tutor:

Ing. Ernesto de la Cruz Guevara.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: elguevara@uci.cu

Graduado como Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Profesor asistente con 8 años de experiencia en el trabajo con Gráficos por Computadoras, Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro de Vertex Entornos Interactivos 3D de la Facultad 5 de la UCI.

Co-tutor:

Ing. Mileydi Moreno Mirabal.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: mmirabal@uci.cu

Graduada como Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Profesor instructor con 8 años de experiencia en el trabajo con Gráficos por Computadoras, Visión por Computadoras y Realidad Aumentada. Actualmente pertenece al Centro de Vertex Entornos Interactivos 3D de la Facultad 5 de la UCI.

RESUMEN

La presente investigación surge de la necesidad de desarrollar aplicaciones de Realidad Aumentada (RA) basadas en marcadores, que muestren una representación de la información que se va a aumentar mediante imágenes tradicionales y no al estilo de los marcadores blanco y negro utilizados hasta el momento. La idea es mostrar al usuario una relación semántica más descriptiva entre la información que se va a aumentar y su respectivo marcador. Por tal motivo se decide utilizar la técnica de detección y extracción de características naturales en imágenes, invariantes a escala y rotación para calcular las transformaciones geométricas necesarias para introducir los gráficos aumentados en la escena real. El objetivo es desarrollar un componente de software para RA que produzcan resultados robustos, mediante la utilización de dicha técnica. En el documento se describen los conceptos referentes a la visión por computadoras (VC) relacionados con esta investigación, especialmente los algoritmos detectores y descriptores de características en imágenes. El autor realiza un estudio para conocer la evolución de los trabajos e investigaciones referentes al objeto de la investigación. Se propone una solución basada en la detección y descripción de características naturales en imágenes que hace uso alternativo de los algoritmos *SURF*, *SIFT*, *FREAK* y *ORB*, implementados en la biblioteca *OpenCV*. A partir de los resultados de tales algoritmos es que se calculan las transformaciones geométricas de los marcadores. La solución propuesta aparece descrita en el documento, así como su implementación para demostrar su validez y cumplir con el objetivo de la investigación.

PALABRAS CLAVE

Descriptores, detectores, realidad aumentada, reconocimiento de patrones, visión por computadoras.

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

En la última década se ha producido un aumento sin precedentes con respecto a la cantidad de contenidos audiovisuales disponibles, debido principalmente al uso masivo de Internet y a la proliferación de dispositivos multimedia en el ámbito cotidiano, tanto a nivel empresarial como personal. La necesidad de soluciones adecuadas es cada vez más demandada en distintas y tan variadas áreas como Internet, aplicaciones de usuario, TV, bibliotecas digitales, aplicaciones médicas, etc., y se requieren métodos de acceso y gestión de la información para hacerla disponible de una manera más eficiente. Las imágenes son un elemento clave que forma parte del gran flujo de información audiovisual que se maneja hoy en día, por lo que es primordial contar con bases de datos de gran capacidad, no solo para almacenar información; sino también para imágenes, que requieren mucho más espacio y deben conservar sus características.

La imagen digital es un producto del desarrollo de la informática que tiene como antecesor a la fotografía (que toma como punto de partida un objeto del mundo real) y a la pintura (donde la imagen ha sido creada por un artista). Las imágenes se almacenan en archivos de diferentes formatos, estas pueden ser enviadas por correo electrónico e incluso ser impresas. Actualmente se puede apreciar el constante uso de las imágenes por parte de los medios de comunicación, pues el respaldo en ellas para trasmitir ideas y enriquecer los textos planos posibilita una mejor percepción de la realidad.

Las imágenes forman parte indisoluble de las nuevas tecnologías como lo es la Realidad Aumentada (RA), la disminución de los costos de los dispositivos que se utilizan en la visión por computadora ha potenciado el auge de la RA en la última década. La RA es una tecnología que complementa la percepción e interacción con el mundo real y permite al usuario estar en un entorno real, aumentado con información adicional generada por el ordenador (1).

En Cuba, se encuentra la Universidad de la Ciencias Informáticas (UCI), la cual es una de las principales instituciones desarrolladoras de software del país, esta cuenta con el grupo de investigación científica ViViRG que tiene dentro de sus líneas de investigación la Realidad Aumentada, los miembros de esta línea de investigación pertenecen al centro VERTEX Entornos Interactivos 3D de la Facultad 5, en donde se implementa una biblioteca de software para el desarrollo de aplicaciones de *RA*. En esta biblioteca se han implementado dos variantes de la *RA* basadas en detección de marcadores planos en blanco y negro, lo cual ha ofrecido

buenos resultados en cuanto a la calidad de las aplicaciones de *RA* implementadas, muestra de ello son los trabajos realizados en (2), (3) y (4).
Los marcadores son elementos físicos que contienen características especialmente diseñadas para ser reconocidas como subregiones en una imagen o video y para extraer información de transformaciones geométricas a partir de su detección. En la actualidad se utilizan marcadores especialmente diseñados con imágenes que contienen códigos binarios en blanco y negro en su interior para las aplicaciones desarrolladas en *VERTEX*.

La utilización de la técnica desarrollada hasta el comienzo de la presente investigación implica que se tenga que imprimir una serie de marcadores en blanco y negro, en lugar de imágenes que pudieran ofrecer una pista al usuario sobre los contenidos que se van a presentar. Esta situación implica una reducción en el grado de eficiencia en cuanto a la utilización del espacio físico destinado a los marcadores y la información adicional que se quiere mostrar. Una situación ilustrativa puede ser la confección de un catálogo de productos enriquecido con *RA*, donde se muestra, encima de las imágenes que los simbolizan, información virtual adicional que representa el proceso de elaboración o el modo de empleo de tales productos.

En el mundo se han desarrollado variantes alternativas en las que se utilizan las fotografías e ilustraciones presentes en libros y revistas como marcadores de *RA*. En estas variantes los autores se basan en el reconocimiento de elementos o características que pueden identificar y describir una imagen del resto dentro de un conjunto y a la vez extraer información sobre sus transformaciones geométricas. La diversidad de algoritmos de este tipo desarrollados por la comunidad científica ofrece resultados que varían en cuanto a la robustez de la detección debido a factores como el cambio de iluminación, la distorsión física y cambio de perspectiva o transformación de la imagen que se va a detectar (rotación, traslación y escala).

Como consecuencia de todo lo planteado con anterioridad surge la siguiente interrogante: ¿Cómo adquirir información geométrica tridimensional para Realidad Aumentada (RA) a partir de las características naturales de una imagen?, el cual constituye el **problema científico** de esta investigación.

Consecuentemente, el **objeto de estudio** se concentrará en los métodos de visión por computadoras para identificar y describir imágenes.

Se define como **objetivo general** de la investigación: implementar un componente de software de Realidad Aumentada, basado en la detección de características naturales en imágenes.

El **campo de acción** está enmarcado en la detección de características naturales en imágenes para RA.

Para darle cumplimiento al objetivo planteado anteriormente se plantearon las siguientes las tareas de investigación:

1. Establecimiento de los referentes teórico-metodológicos sobre los algoritmos de detección y descripción de características naturales en imágenes para Realidad Aumentada.
2. Estudio comparativo de los diferentes métodos de detección y descripción de características naturales en imágenes para Realidad Aumentada.
3. Selección de los algoritmos de detección de características naturales que se van a incorporar a la biblioteca de *RA* mediante un estudio comparativo.
4. Implementación de un componente de software para la biblioteca de RA, que agrupe las funcionalidades de los algoritmos de detección de características naturales seleccionados.
5. Validación de los resultados obtenidos al introducir el componente de software implementado.

Los métodos científicos de investigación que se tendrán en cuenta para el desarrollo de esta investigación son los siguientes:

Métodos Teóricos

- **Histórico-Lógico:** El mismo fue seleccionado para llevar a cabo un análisis valorativo de la bibliografía existente, así como para conocer las tendencias actuales en cuanto tratamiento de características unívocas en las imágenes empleando herramientas de visión por computador. Se analiza el estado del arte existente respecto a la problemática planteada y se selecciona la bibliografía así como las herramientas y/o bibliotecas más utilizadas para este fin.
- **Analítico-Sintético:** Se utiliza este método científico de investigación para poder descomponer el problema en partes más concretas y llegar a conclusiones más específicas, y luego lograr la correcta integración y comunicación entre las mismas.
- **Inductivo-Deductivo:** Este método permite inducir a partir de situaciones particulares y de un comportamiento a un nivel general. Además es posible deducir a partir, de datos habituales y una conducta específica, lo que puede ser útil en la prueba de los algoritmos seleccionados. También es utilizado para definir patrones comunes de

comportamiento, que pueden ser ajustados en la implementación de nuevos algoritmos. Se evalúa la problemática existente, para determinar aspectos particulares y desarrollar una propuesta de solución.

Métodos Empíricos

- **Entrevista:** La entrevista a especialistas con experiencia en el empleo de las herramientas seleccionadas para el desarrollo de la investigación es una fuerte herramienta para guiar el desempeño la misma.
- **Consulta de fuentes de Información:** Permitió determinar el estado del arte del objeto de investigación, consultando una bibliografía completa y actualizada.

Este documento está estructurado de la siguiente forma:

Capítulo 1: *"Fundamentación teórica"*. Constituye la base teórica de la investigación realizada. Se describen los principales conceptos relacionados que permiten entender cómo se realiza el proceso de detección y descripción de características naturales en imágenes para utilizarlas en aplicaciones de RA, así como los algoritmos más utilizados para ello. Además se describen las herramientas y metodologías utilizadas para el desarrollo del *software*.

Capítulo 2: *"Solución Propuesta". S*e describe la propuesta de solución teniendo en cuenta el marco teórico, la metodología y las herramientas a utilizar para el desarrollo de software. Además, se muestran los documentos generados por la metodología empleada.

Capítulo 3: *"Implementación y validación de la solución propuesta"*. Se exponen los detalles correspondientes a la implementación del sistema. También se describen las pruebas realizadas al sistema y los principales resultados obtenidos.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un estudio del estado del arte referente a la detección y descripción de las características invariantes en las imágenes. Principalmente, se hace énfasis en los elementos necesarios que se deben conocer para la adquisición de las transformaciones geométricas de las imágenes, para así hacer uso de ellas y generar resultados de RA. Además se describe y justifica el uso de la metodología y herramientas seleccionadas por el autor para implementar el componente de RA que quedó plasmado en el objetivo del presente trabajo.

1.2 Conceptos asociados al dominio del problema

Para adentrarse al área de conocimiento que incluye el proceso de detección y extracción de características invariantes en imágenes, es necesario tener conocimiento previo de los conceptos que facilitan un mejor entendimiento de la investigación. En la siguiente sección se abordarán algunas de estas definiciones.

1.2.1 Realidad Aumentada

Debido a que el problema existente está enmarcado en el campo de la Realidad Aumentada (RA), se hace necesario un estudio sobre el tema. La *RA* es una tecnología relativamente nueva, los primeros trabajos de investigación en este campo aparecen en los años 90. Sin embargo el concepto no era nuevo. El primer trabajo de *RA* se le atribuye a *Ivan Sutherland*, que en 1968 construyó un prototipo que consistía en un casco de realidad virtual y un sistema de *tracking* mecánico, que permitían superponer imágenes tridimensionales generadas por ordenador.

Hollywood tampoco se quedó atrás, en los 80 la imaginación de los cineastas ya concebía sistemas de *RA* en películas como "*Terminator"* o "*Robocop"*. La figura 1 ilustra un ejemplo de esto.

Figura 1 Captura de la película Terminator

Una de las primeras definiciones más completas para esta disciplina se realizó en el año 1994 a través del continuo *Realidad-Virtualidad* conocido como el *Continuo de Milgram* (5). Este continuo hace referencia a la combinación entre elementos de la realidad con elementos virtuales, de manera que según el grado de incidencia de unos u otros se pueden definir cuatro espacios: mundo real, realidad aumentada, virtualidad aumentada y mundo virtual. En la figura 2 se representa un esquema de esta clasificación. Se conoce como *Realidad Mezclada (RM)* al conjunto de *RA* y *Virtualidad Aumentada (VA)* y la diferencia entre estas dos tecnologías radica en la cantidad de contenido virtual frente a la cantidad de contenido procedente del mundo real.

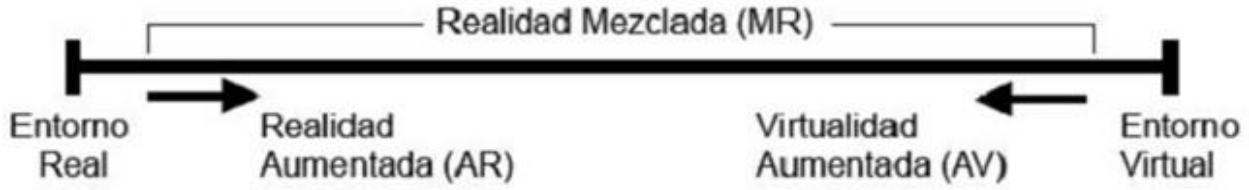

Figura 2. Continuo de Milgram

La definición de RA ha evolucionado a lo largo de los años a medida que se ha profundizado en la investigación. Al principio, las primeras definiciones del concepto estaban directamente vinculadas al dispositivo de visualización, en concreto, a los *HMD (Head Mounted Display)*, de manera que se utilizaba el concepto de realidad aumentada para explicar la capacidad de un dispositivo *HMD* de superponer gráficos virtuales al entorno del usuario.

En 1997 Ronald Azuma (2) da una nueva definición de la *RA*, y plantea que la misma permite al usuario ver el mundo real, con objetos virtuales superpuestos sobre el mundo real, o

compuestos con él. De ahí que la Realidad Aumentada actúe como complemento de la realidad, a diferencia de la Realidad Virtual que reemplaza completamente la realidad. Algo muy importante a señalar dentro de la definición que da *Ronald Azuma* de la *RA* es que esta:

- Combina elementos reales y virtuales.
- Es interactiva y en tiempo real.
- Está registrada en 3D.

Luego de llevar a cabo este análisis del tema relacionado con la *RA* se pudo apreciar como en muchas definiciones dadas, se coincide en que esta tecnología es la definición de la superposición de información virtual sobre entornos reales a partir de una aplicación informática. De esta manera, se puede combinar la visión real en el día a día con información añadida que puede hacer cambiar la manera en que el usuario interpreta la realidad.

Hoy en día esta tecnología empieza a ser más común. Algunos ejemplos de las posibles aplicaciones de la *RA* son:

- **Mantenimiento y montaje:** En este campo la *RA* puede ayudar a un operario que tenga que hacer una tarea de montaje o mantenimiento de alguna máquina mostrando información sobre posiciones de piezas, conexiones de cableado, pares de apriete, etc.
- **Medicina:** Un médico podría utilizar la *RA* para ver el interior del paciente antes de una operación para poder planificarla, o durante ella para localizar algún tejido dañado.
- **Cine:** Las técnicas de vídeo aumentado se usan hoy en día para facilitar el montaje de efectos especiales. Ejemplos de aplicaciones pensadas para este propósito son *RealMiz Matchmover*, *2d3 Boujou*, *The Píxel Farm PFTrack*, *Science-D-Visions 3d Equalizer* y *Andersson Technologies LLC Syntheyes*.
- **Arquitectura:** El vídeo aumentado se puede utilizar para predecir el impacto que tendrá una construcción sobre el paisaje, grabando el terreno sobre el que se construirá y añadiendo un modelo 3D del edificio que se levantará.

Existen varias técnicas de *RA* con las cuales se puede enriquecer la senso-percepción de un usuario con información virtual respecto al mundo que le rodea. Esta información puede ser en forma de sonidos, gráficos, tacto, sabor y olor. En la presente investigación la información que se pretende insertar en el mundo real es información gráfica, la cual se puede hacer a su vez con varias técnicas. En este sentido se han utilizado sensores magnéticos, ultrasonidos, ópticos y mecánicos así como Sistemas de Posicionamiento Global para ambientes exteriores.

En la presente investigación se emplearán sensores ópticos como cámaras web, a través del procesamiento de la imagen que devuelve la cámara, en busca de patrones que puedan ser reconocidos para extraer información geométrica de utilidad para situar los gráficos tridimensionales en la perspectiva correcta del usuario.

1.2.2 La información visual

La información visual de una imagen incluye elementos tales como el color, textura, forma, localización espacial y regiones de interés (3), las cuales constituyen características primitivas o de bajo nivel, en contraste con las de alto nivel, que las describen de manera unívoca. La información visual puede ser tratada desde dos enfoques: global, si se analiza la imagen completa y local, si se aplica a una región de la imagen. Estas características pueden ser tratadas y capturadas por los detectores de puntos clave y descriptores de imágenes.

1.2.3 Detector

Los detectores son algoritmos que se encargan de localizar y guardar las zonas que brindan información unívoca de la misma, estas regiones poseen información distintiva. Es de gran importancia representar estas zonas en imágenes similares, tomadas bajo condiciones distintas(7). Dichas zonas deben ser invariantes ante transformaciones realizadas en la imagen, como por ejemplo: la translación, rotación, cambio de escala y de perspectiva.

1.2.4 Descriptores

Debido a la necesidad de describir el contenido de la información visual, surgen como respuesta los descriptores de imagen. Los descriptores son mecanismos de extracción de características o información visual. Existen dos grupos de ellos: globales y locales. Los globales son los que resumen el contenido de la imagen en un único vector o matriz y los descriptores locales son aquellos que representan la información de la imagen por regiones de interés, por lo que están constituidos por todos los vectores de características de las regiones identificadas en la imagen (7). Estas regiones poseen información distintiva, que suele ser importante representarlas en imágenes similares, tomadas bajo condiciones distintas.

Los descriptores tienen varias formas de representación, entre las tres más comunes se encuentran: los histogramas, descriptores basados en particiones y basados en regiones. Los histogramas son los más ampliamente usados, aunque se crean nuevos enfoques que superan la falta de información espacial de los mismos, que son los basados en particiones, donde la imagen se divide en fragmentos fijos, y por cada una se extrae las propiedades de bajo nivel. El

enfoque basado en regiones es mucho más avanzado, se dirige a la detección de objetos, asemejándose a la forma en que los seres humanos interpretan las imágenes.

Según (7) un histograma representa la frecuencia de aparición de cada una de las intensidades de color presentes en la imagen, mediante la contabilidad de los pixeles que comparten dichos valores de intensidad de color. El histograma está compuesto por diferentes rangos o contenedores que representan un valor o conjuntos de valores de intensidad de color.

1.2.5 Transformación afín

Desde el punto de vista matemático, una transformación afín no es más que cualquier transformación que se pueda expresar en forma de una multiplicación de una matriz, seguido por una adición vectorial. Las transformaciones afines se pueden visualizar de la siguiente manera: cualquier paralelogramo ABCD en un plano se puede asignar a cualquier otro A'B'C'D paralelogramo por una “transformación afín”. Si las áreas de estos paralelogramos son distintas de cero, entonces la transformación afín implícita se define únicamente por tres vértices de los dos paralelogramos (8).

En el campo de visión por computador en lo que al trabajo con imágenes se refiere, se entiende por transformación afín: cualquier tipo de transformación de rotación, escalado o traslación que se produzca en las dos dimensiones espaciales del plano. El elemento básico de toda transformación afín es la *matriz de transformación*, que será la encargada de definir la clase y los parámetros de la transformación a realizar. Así existirá siempre una matriz asociada tanto para el escalado, como para la rotación y la traslación respectivamente (9). Estas trasformaciones pueden representarse por una matriz de dimensiones 2 por 3.

1.2.6 Transformación de perspectiva u homografía

Por otro lado están las transformaciones de perspectiva u homografía. En geometría, una homografía establece una transformación de proyección entre dos planos diferentes (10). Sean P_a y P_b dos puntos pertenecientes a un plano A y B respectivamente, de la siguiente forma:

$$P_a=\begin{bmatrix} X_a \\ Y_a \\ 1 \end{bmatrix} \quad P_b=\begin{bmatrix} X_b \\ Y_b \\ 1 \end{bmatrix}$$

Una homografía se define matemáticamente como una matriz de dimensiones 3 por 3, H_{ab}:

$$H_{ab}=\begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{bmatrix}$$

Esta matriz permite transformar un punto de P_a del plano A a un punto P_b en el plano B, como muestra la ecuación 1, y un punto P_b del plano B a un punto en el plano A, como muestra la ecuación 2, donde $H_{ba}= H_{ab}^{-1}$.

$$P_b= H_{ab} \times P_a \qquad \textit{ecuación 1.}$$

$$P_a= H_{ba} \times P_b \qquad \textit{ecuación 2.}$$

En el contexto de la visión por ordenador, homografía casi siempre se refiere a la asignación entre puntos sobre dos planos de imagen que corresponden a la misma ubicación en un objeto plano en el mundo real. Se puede demostrar que una asignación de este tipo es representable por una sola matriz ortogonal de 3x3 (8).

Las *transformaciones afines* pueden convertir rectángulos en paralelogramos. Pueden deformar la figura, pero deben mantener los lados paralelos. Las *transformaciones de perspectiva* ofrecen más flexibilidad, un cambio de perspectiva puede convertir un rectángulo en un trapezoide. Ya que los paralelogramos son también trapezoides, se puede decir que las *transformaciones afines* son un subconjunto de las *transformaciones de perspectiva*. La *figura 3* muestra ejemplos de diferentes *transformaciones afines* y de *perspectiva*.

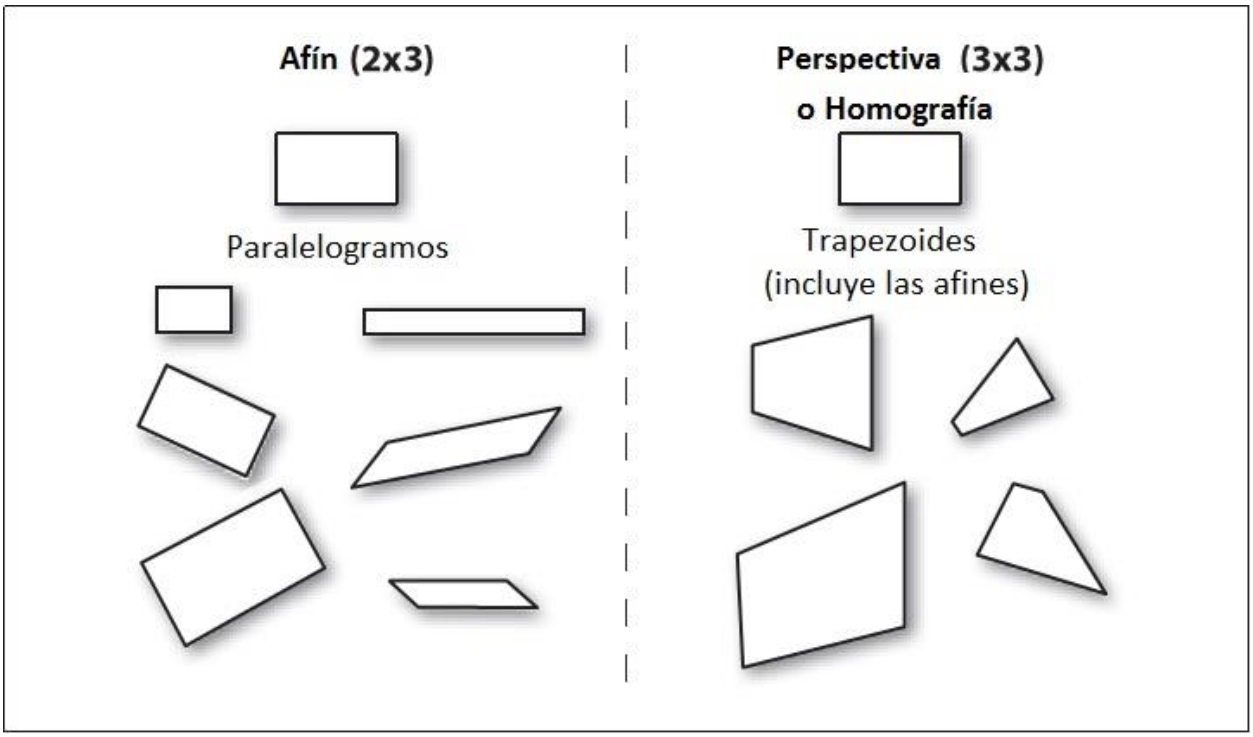

Figura 3. Transformaciones afines y de homografía

1.2.7 Calibración de la cámara

La RA trata de fusionar los objetos del mundo real con un contenido virtual. Para posicionar un objeto 3D en una escena, se necesita conocer su posición con respecto a la cámara que se usará para realizar la obtención de los *fotogramas* de video.

El lente de cada cámara tiene parámetros únicos, como la longitud focal, punto principal y patrón de distorsión del lente. Estos parámetros son conocidos también como parámetros internos o intrínsecos. El proceso de encontrar estos parámetros es llamado calibración de la cámara. El proceso de la calibración de la cámara es sumamente importante para las aplicaciones de RA, ya que mediante este se describe la transformación de perspectiva y la distorsión del lente en la imagen de salida (11). A grandes rasgos hay dos opciones a la hora de calibrar una cámara: *online* y *offline*.

1.2.7.1 Calibración offline

La calibración *offline* es la que se hace antes de comenzar a capturar la secuencia de vídeo que se va a aumentar. La idea es capturar imágenes de un objeto conocido y calcular a partir de éstas los parámetros intrínsecos. Uno de los algoritmos más populares fue propuesto por Tsai en 1987 (12). Este algoritmo utiliza un patrón de calibración compuesto por dos superficies

no coplanares con marcas impresas. La calibración se hace mediante el emparejado de las marcas detectadas en la imagen con sus correspondientes puntos 3D en el espacio. En la figura 4 se ilustra el patrón de calibración propuesto por Tsai

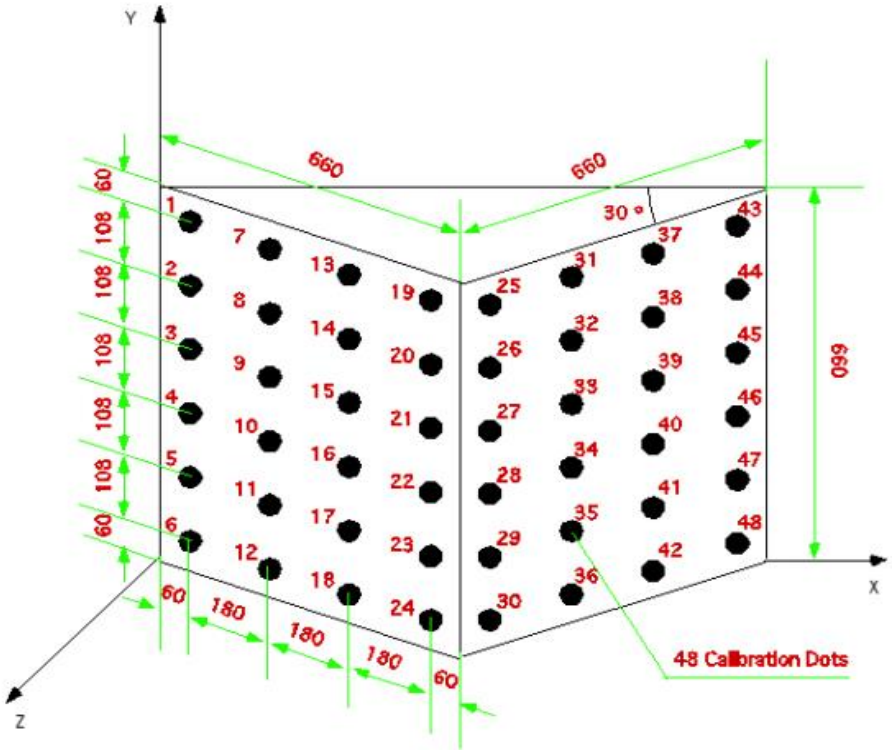

Figura 4. Patrón de calibración de Tsai

Hoy en día el método más usado es el propuesto por Zhang (13). Este método requiere al menos dos vistas de un patrón de calibración plano. Es el algoritmo usado en la biblioteca *OpenCV*, cada vez más extendida en el campo de la visión por computador. Zhang propone una técnica de calibración basada en la observación de una plantilla plana desde varias posiciones. La ventaja de este método de calibración es que permite obtener los parámetros de la cámara fácilmente a partir de una plantilla plana en la cual no es necesario conocer las posiciones de los puntos de interés, ni tampoco las posiciones de la cámara desde donde se han tomado las imágenes. La cámara se puede mover simplemente con la mano. Esto hace que sea una técnica muy flexible (11) .

En general estos algoritmos son precisos y rápidos, pero no siempre es posible usarlos, ya que en muchas ocasiones sólo se dispone de la secuencia de vídeo a aumentar y no de la cámara que la capturó.

1.2.7.2 Calibración online

Esta calibración es la que se ejecuta sobre las propias imágenes a aumentar. Consiste en encontrar un conjunto de parámetros internos que sean consistentes con la geometría proyectiva subyacente a la secuencia de imágenes. Normalmente no necesitan intervención del usuario, por lo que resultan adecuados para cámaras con lentes varifocales (enfoque y zoom).

Los métodos de calibración online consisten en encontrar una entidad geométrica llamada cónica absoluta, que tiene la propiedad de ser invariante a las transformaciones de rotación y traslación de la cámara. A esta cónica se la conoce como la cónica de calibración, ya que sólo depende de los parámetros internos de la cámara. Dentro de este grupo hay dos alternativas, dependiendo del conocimiento que se tenga de la escena capturada.

La primera de las alternativas aprovecha la existencia de líneas paralelas en la imagen. Hay que recordar que una cámara aplica una transformación de proyección sobre la escena que captura, por lo tanto las proyecciones de líneas paralelas se cruzan en un plano especial llamado el plano en el infinito. A estos puntos de cruce se les conoce como puntos de fuga. Conocer los puntos de fuga en tres direcciones ortogonales permite calcular la cónica absoluta y con ella los parámetros internos de la cámara (14).

La segunda alternativa según (15), se basa en que las imágenes de la cónica absoluta proyectadas en dos fotogramas diferentes de una secuencia de vídeo deben cumplir ciertas restricciones geométricas, llamadas restricciones epipolares. La figura 5 ilustra lo planteado respecto a la restricción de la cónica absoluta.

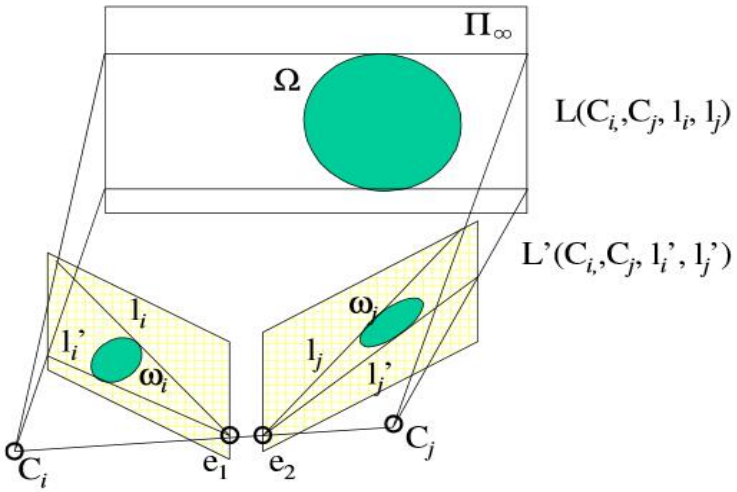

Figura 5. Restricción epipolar en la cónica absoluta

1.2.8 Estimación de la pose

Se conoce como *pose* a la posición que ocupa la cámara respecto a la escena que captura (rotación y traslación). A este conjunto de parámetros se les conoce como parámetros externos o extrínsecos, ya que dependen únicamente de la posición de la cámara respecto a un sistema global de coordenadas (16). Esta definición se ilustra mediante la figura 6.

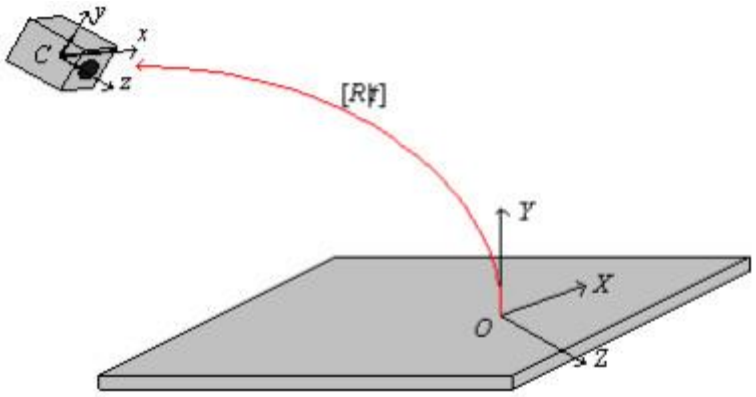

Figura 6. Parámetros extrínsecos

Conociendo los parámetros extrínsecos e intrínsecos de la cámara es posible añadir objetos virtuales a la escena de forma realista. En la figura 7 se puede ver un ejemplo de imagen aumentada, en la que se añade una tetera virtual a una mesa.

Figura 7. Escena de realidad aumentada

En los últimos años se han probado diferentes estrategias para calcular la pose de la cámara, estos van desde sensores inerciales hasta *GPS* o *trackers* magnéticos. Sin embargo la tendencia de la mayoría de autores es usar directamente las imágenes capturadas por la cámara, que es la opción elegida en este trabajo, debido a que es la estrategia utilizada por el grupo de investigación Visualización y Realidad Virtual (VIVIRG).

A la hora de calcular la pose de la cámara se utilizan únicamente las imágenes que ésta captura. Existen básicamente tres opciones, dependiendo del grado de conocimiento que se tenga sobre la escena:

1. Los primeros métodos que se usaron aprovechaban la existencia de objetos conocidos en la imagen, como *marcadores*. La implementación más conocida de este tipo de *tracking* es la biblioteca *ARToolkit* de la universidad de Washington.
2. El segundo grupo se basa en el conocimiento de la geometría de los objetos situados en la escena. Si se tiene en cuenta que la posición de una cámara respecto a un objeto es la inversa de la posición del objeto respecto a la cámara, se pueden utilizar algoritmos de *tracking* de objetos para aplicaciones de realidad aumentada.
3. En el tercer grupo entran los algoritmos de tipo *SFM (Structure From Motion)*. Estos métodos se basan en el efecto *parallax* para calcular el movimiento de la cámara a lo largo de la secuencia. El efecto *parallax* es el movimiento aparente de un objeto respecto al fondo cuando cambia el punto de vista del observador (16). En la figura 8 se aprecia como el punto *y*, cambia notablemente de posición al moverse la cámara.

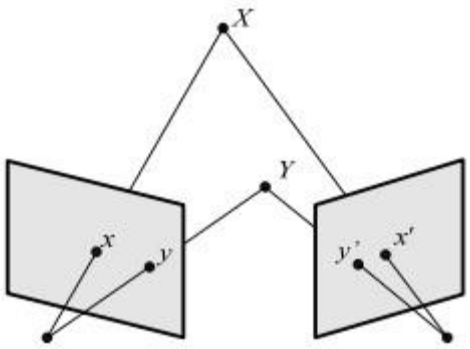

Figura 8. Efecto Parallax

1.2.9 Pasos para el análisis de características en las imágenes

A continuación se muestran los pasos lógicos que se seguirán para el análisis de las características naturales de las imágenes. Los mismos están pensados a partir de la lógica de las funcionalidades que brinda *OpenCV.*

1. Identificación de puntos clave (*keypoints*).
2. Descripción de los *keypoints*.

3. Establecimiento de correspondencias entre descripciones (*matching*).
4. Refinamiento del *matching*.
5. Cálculo de la homografía.

En el primer y segundo paso, la salida del sistema varía de acuerdo a los distintos algoritmos que se empleen, pues para determinadas características de las imágenes, estos algoritmos difieren en sus resultados calculados.

En el tercer punto los distintos algoritmos podrían diferir de la propia función de similitud o de distancia, como también se conoce, que se emplee para determinar los candidatos a posibles resultados. Por lo general estos métodos emplean un mecanismo de umbral para determinar cuáles de los candidatos a resultados realmente lo son. La definición de este umbral es un tema también distintivo de estos algoritmos.

El cuarto paso se obtiene al optimizar el resultado de la etapa de *matching*, pues es normal que durante el paso anterior se obtengan resultados erróneos.

El último paso puede estar aparejado a algún tipo de prioridad o de ordenamiento a la hora de mostrarle al usuario el resultado.

1.3 Algoritmos utilizados para la detección y extracción de características naturales en las imágenes

OpenCV ofrece una variada gama de algoritmos ya implementados para la detección y extracción de características naturales en imágenes, entre los que se destacan:

- ✓ *SURF(Speeded Up Robust Features)*
- ✓ *SIFT(Descriptor Scale Invariant Feature Transform)*
- ✓ *FAST (Features from Accelerated Segment Test)*
- ✓ *MSER (Maximally Stable Extremal Regions)*
- ✓ *BRISK (Binary Robust Invariant Scalable Keypoints)*
- ✓ *BRIEF(Binary Robust Independent Elementary Feature)*
- ✓ *FREAK (Fast Retina Keypoint)*

- ✓ *ORB(Oriented FAST and Rotated BRIEF)*

Luego de un detallado estudio de los métodos anteriormente mencionados se seleccionaron los algoritmos SURF, SIFT, ORB y FREAK, de los cuales se brinda una breve descripción sobre sus aspectos más novedosos por los cuales seleccionaron para el desarrollo de esta investigación. Los criterios que se tuvieron en cuenta para que estos métodos de detección y extracción de características naturales en imágenes formaran parte del presente trabajo son:

- ✓ La robustez y fortaleza ante cambios de iluminación, rotación, escala y perspectiva.
- ✓ El hecho de que algunos de ellos fueran el producto de mejoras de otros algoritmos.
- ✓ El nivel de aceptación por las comunidades científicas ante algunos de ellos.

1.3.1 Descriptor Scale Invariant Feature Transform (SIFT)

Descriptor originalmente desarrollado por Lowe en el 2004 (17) para el reconocimiento de objetos de manera general, propuesto como un algoritmo capaz de detectar puntos característicos estables y además "invariantes frente a diferentes transformaciones como traslación, escala, rotación, iluminación y transformaciones afines" (7) en una imagen. El algoritmo *SIFT* está siendo muy difundido en la actualidad, la prueba es la cantidad de variantes que han surgido hasta llegar a constituir una familia de métodos basados en SIFT.

El proceso de extracción de características de una imagen en este descriptor se define en cuatro fases según Lowe (17):

1. **Detección de Extremos en el Espacio Escala:** tiene como objetivo la detección de los posibles puntos de interés, o sea los puntos candidatos.
2. **Localización de los Puntos de Interés:** se discriminan los puntos candidatos cuya estabilidad se vea afectada, según Lowe "los puntos no firmemente situados sobre los bordes o aquellos con bajo contraste son bastante vulnerables al ruido y por lo tanto no podrán ser detectados bajo pequeños cambios de iluminación o variación del punto de vista de la imagen."
3. **Asignación de la Orientación:** se asigna a cada punto de interés una orientación basada en las propiedades locales de la imagen para garantizar la invarianza respecto a la rotación.

4. **Descriptor del Punto de Interés:** en esta etapa se crea un vector de 128 características por cada uno de los puntos de interés, que contiene una estadística local de las orientaciones del gradiente de la escala de espacio gaussiano.

1.3.2 Speeded Up Robust Features (SURF)

El descriptor *SURF (Speeded-Up Robust Features)* es uno de los más utilizados para la extracción de puntos de interés. Fue planteado por Herbert Bay (18) en el año 2006. *SURF* utiliza la matriz hessiana y su determinante para la localización de los puntos y la determinación de la escala, que produce la reducción del tiempo de computación, esto hace al algoritmo más preciso y rápido. Una vez obtenida la escala mediante la matriz y su determinante, se calcula la orientación del punto de interés, para posteriormente calcular el descriptor *SURF*.

El cálculo de la orientación le otorga al descriptor invarianza ante la rotación. El vector obtenido por este algoritmo es distintivo y al mismo tiempo robusto al ruido, errores y deformaciones geométricas y fotométricas. La longitud del vector tiene una dimensión de 64 valores, lo que supone una reducción de la mitad de la longitud del *SIFT*, además no permite que haya varios puntos invariantes en una misma posición, con distinta escala y/u orientación y utiliza siempre la misma imagen original.

1.3.3 Oriented FAST and Rotated BRIEF (ORB)

ORB se perfila como una alternativa a *SURF* y *SIFT*, teniendo en cuenta el costo computacional, y principalmente que tanto *SURF* como *SIFT* son algoritmos patentados. El mismo fue propuesto por Rublee, y supera la falta de invariancia a la rotación de *BRIEF*. Es una modificación del algoritmo *FAST*. Originalmente *FAST* es sorprendentemente rápido, pero no calcula la orientación ni el tamaño del *keypoint*, a diferencia de *ORB* que si realiza este cálculo, y el tamaño del *keypoint* es fija (19).

ORB calcula una orientación local a través del uso de un centroide intensidad, el cual es una media ponderada de intensidades de los píxeles en la región local, y se asume que no debe ser coincidente con el centro de la función. La orientación es el vector entre la ubicación de características y el centroide. Si bien esto puede parecer inestable, es competitivo con la asignación de orientación única empleada en *SIFT*. El patrón de muestreo empleado en este algoritmo utiliza 256 pares de comparaciones de intensidad en contraste con *BRIEF*, se construye a través de una máquina de aprendizaje, maximizando la varianza del descriptor y minimizando la correlación bajo varios cambios de orientación (20).

1.3.4 Fast Retina Keypoint (FREAK)

En la actualidad, el despliegue de algoritmos de visión en los teléfonos inteligentes y los dispositivos integrados con poca memoria y la complejidad de cálculo persiguen un objetivo: hacer cálculos de los descriptores de una manera más rápida, más compacto, sin dejar de ser resistente a la escala, la rotación y el ruido.

Para abordar mejor las necesidades actuales, se propone en (21) un descriptor de *keypoints* novel inspirado en el sistema visual humano y más precisamente en la retina, acuñado *Fast Retina Keypoint (FREAK).* En muchas cuadrículas de muestreo es posible comparar pares de intensidades de los píxeles, *ORB* utiliza pares aleatorios, *FREAK* sugiere el uso de la cuadrícula de muestreo de la retina, que también es circular, con la peculiaridad de que tiene una mayor densidad de puntos cerca del centro.

Cada punto de muestra necesita ser suavizado para ser menos sensible al ruido. *BRIEF* y *ORB* utilizan el mismo *kernel* para todos los puntos en la región. Para que coincida con el modelo de retina, este algoritmo utiliza *kernels* de diferentes tamaños por cada punto de muestreo. Con el fin de estimar la rotación de los *keypoints*, *FREAK* suma los gradientes locales estimados a través de pares seleccionados, los cuales tienen la característica de poseer campos receptivos simétricos con respecto al centro.

1.4 Metodología de desarrollo

Una metodología es el conjunto de técnicas y procedimientos que permiten conocer los elementos necesarios para definir un proyecto de software, es la base para la edificación de un producto de este tipo. Esencialmente, sirve para aumentar la "calidad" del software y controlar de manera transparente todo el proceso de desarrollo. Si se quiere que un proyecto sea escalable y flexible a los cambios, se recomienda tomar en cuenta una metodología de desarrollo de *software*. Las metodologías, dadas sus características, se enmarcan en dos grandes grupos: las llamadas "metodologías pesadas" y las "metodologías ágiles". La diferencia más notable entre estos dos grupos es que mientras los métodos pesados intentan obtener los resultados apoyándose principalmente en la documentación ordenada, los métodos ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso (22).

1.4.1 Programación Extrema

Durante todo el proceso de desarrollo de software los objetivos fundamentales son que el producto final tenga la calidad requerida y que cumpla con las exigencias del cliente. Para lograr que estos objetivos se realicen es necesario utilizar una metodología de desarrollo de software, para escogerla hay que basarse en las características específicas que tiene cada producto.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Se basa en la comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y agilidad para enfrentar los cambios. *XP* es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Se plantea que el ciclo de vida ideal de *XP* consiste de seis fases: Exploración, Planificación de la Entrega *(Release)*, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto (22).

El ciclo de desarrollo se define en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.

5. Vuelve al paso 1.

Las características esenciales de *XP* son: historias de usuario (HU), roles, procesos y prácticas. Donde las HU constituyen tarjetas para especificar las necesidades del software, son comprensibles y delimitadas para que los programadores puedan implementarlas en poco tiempo. Plantea roles como: programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador, consultor y gestor. El proceso no es más que el ciclo de vida y las prácticas consisten en la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo necesario para que el diseño evolutivo funcione.

La metodología *XP* se caracteriza por realizar entregas pequeñas y frecuentes, producto de dichas pautas, se puede llevar un control de las tareas y se obtienen resultados en muy poco tiempo. Además, no exige una amplia documentación del trabajo realizado.

Debido a que el autor dispone de un corto tiempo para la entrega final de la solución, es el único miembro del equipo de desarrollo de la presente investigación; en la que los requisitos corren el riesgo de ser inestables y consecuentemente se realizarán pequeñas entregas para evaluar el avance de la solución, se decide utilizar la metodología *XP* para guiar el proceso de desarrollo de la solución.

1.5 Tecnologías a utilizar

Un elemento importante a la hora de desarrollar un software, lo constituye las herramientas a utilizar, las cuales pueden contribuir de una forma importante a mejorar la efectividad del mismo así como el tiempo de desarrollo.

1.5.1 Biblioteca

Las bibliotecas de visión por computador facilitan el manejo y procesamiento de las imágenes, dentro de estas bibliotecas se encuentran: *The Matrox Image Library (MIL)*, *Khoros, eVision*, *HIPS*, *Exbem, Aphelion*. Sin embargo estas poseen ineficiencias plausibles: sus ciclos de actualización son largos y por ende lentos, algunas carecen de un entorno de desarrollo de alto nivel, dificultando su uso, y las que disponen de ellos, están limitadas por la plataforma de desarrollo y el propio hardware de captura. Por otro lado se encuentran *OpenCV*, *VXL* y *LTI-Lib*. Este último es un producto escrito para *Windows*, siendo la característica clave que impide utilizarlo en este trabajo. *VXL* por su parte está diseñado para ser portable en muchas plataformas, el mismo trabaja con matrices, vectores, descomposiciones y optimizadores, además de poder cargar, guardar y manipular imágenes en numerosos formatos de archivo comunes, incluyendo imágenes muy grandes, sin mencionar la capacidad que posee de trabajar con puntos, curvas y otros objetos elementales de dos o tres dimensiones. Una de sus desventajas fundamentales comparado con *OpenCV* es que no posee un marco de trabajo completo para el desarrollo de aplicaciones relacionadas con la visión por computador, careciendo por ejemplo de algoritmos de clasificación y de detección de contornos (23).

Para el desarrollo de esta investigación se utiliza la biblioteca de *OpenCV* 2.4.4, que posee más de 500 funciones desarrolladas en *C* y *C++* las cuales son utilizadas para desarrollar tanto productos comerciales como no comerciales. La biblioteca actúa bajo la licencia *BSD*, por lo que es libre y de código abierto, siendo uno de los requisitos primordiales para desarrollar el

componente que dará solución al problema descrito anteriormente, además puede ser utilizada tanto en *Windows* como *Linux*. La versión 2.4.4 agrega el módulo *features2d*, que implementa detectores de características y provee herramientas de alto nivel para el cálculo de correspondencias entre imágenes y detección de objetos y texturas. *OpenCV* ha sido diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real.

1.5.2 Lenguaje de programación

Un lenguaje de programación consiste en todos los símbolos, caracteres y reglas de uso que permiten a las personas "comunicarse" con las computadoras (24). Existen numerosos y dialectos de programación diferentes, cada uno de ellos fueron creados para un fin determinado, por lo que algunos son mejores que otros a la hora de realizar una tarea específica.

La utilización de la biblioteca *OpenCV* para el tratamiento de imágenes, motiva a utilizar como lenguaje base *C++*, lo que permite la compatibilidad y fusión de las sentencias netamente de *OpenCV* y el código propio de la aplicación.

En la actualidad, *C++* es un lenguaje versátil, potente, general y uno de los más utilizados, siendo una versión mejorada del lenguaje *C*. El *C++* mantiene las ventajas del *C* en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Incorpora la programación orientada a objetos y da la posibilidad de redefinir los operadores, es decir, permite la sobrecarga de operadores, y de poder crear nuevos tipos que se comporten de manera diferente. Además añade al tratamiento de excepciones y permite trabajar tanto a alto como a bajo nivel (24).

1.5.3 Entorno de desarrollo integrado (IDE)

La utilización de *OpenCV* como biblioteca de visión por computador y de *C++* como el lenguaje base del componente estimula a la utilización de *Qt Creator* como *IDE* de desarrollo. Este *IDE* posee un avanzado editor de código *C++*, lenguaje que utiliza de forma nativa, y capacidad de vincularse fácilmente con la biblioteca *OpenCV*, permitiendo aprovechar la potencia de esta para el procesamiento de imágenes y video. Además, este *IDE* permite el desarrollo rápido de aplicaciones interactivas en entornos de ventanas. Ambos productos son multiplataforma, su uso es abierto y gratuito, otorgándole libertad al programador de ver el código fuente y poder modificarlo, además de utilizarlo sin restricciones de licencia.

1.6 Conclusiones Parciales

En este capítulo se ofreció una introducción a los temas de la detección y descripción de características invariantes de las imágenes, así como la extracción de su respectiva información geométrica para conocer su perspectiva con respecto al espacio 3D. Se dieron a conocer los principales conceptos relacionados con el objetivo de esta investigación y se sentaron las bases para comenzar a desarrollar el componente basado en la detección de las características naturales en imágenes.

CAPÍTULO 2 SOLUCIÓN PROPUESTA

2.1 INTRODUCCIÓN

En el presente capítulo se especifican las características fundamentales del sistema. Se hace referencia a las fases de exploración y planificación que plantea la metodología de desarrollo *XP*. Además se muestran elementos como: los requerimientos del sistema, la arquitectura y los patrones de diseño que se utilizan para la implementación del componente a desarrollar.

2.2 Propuesta de solución

En el presente trabajo se propone desarrollar un componente, que brinde al Centro Vertex la posibilidad de realizar el procesamiento de imágenes basado en el reconocimiento de elementos o características que pueden identificar y describir una imagen del resto dentro de un conjunto, incorporando algoritmos que produzcan resultados de detección de características naturales robustos, que sean invariantes a la rotación, traslación y escalado de la imagen.

A continuación, en la *figura 9* se ilustra cómo quedó definida la estructura funcional del componente y en la *tabla 1* se muestra una breve descripción de la misma. Dentro del componente se definieron cuatro procesos claves como es el caso de: la detección y extracción de *keypoints*, el cálculo y corrección de correspondencias. Para el desarrollo de estos procesos es necesario utilizar algunos algoritmos y técnicas, que a pesar de que algunos no son los más eficientes en la labor que realizan, permiten probar el correcto funcionamiento del componente que se presenta.

2.2.1 Estructura Funcional del Sistema

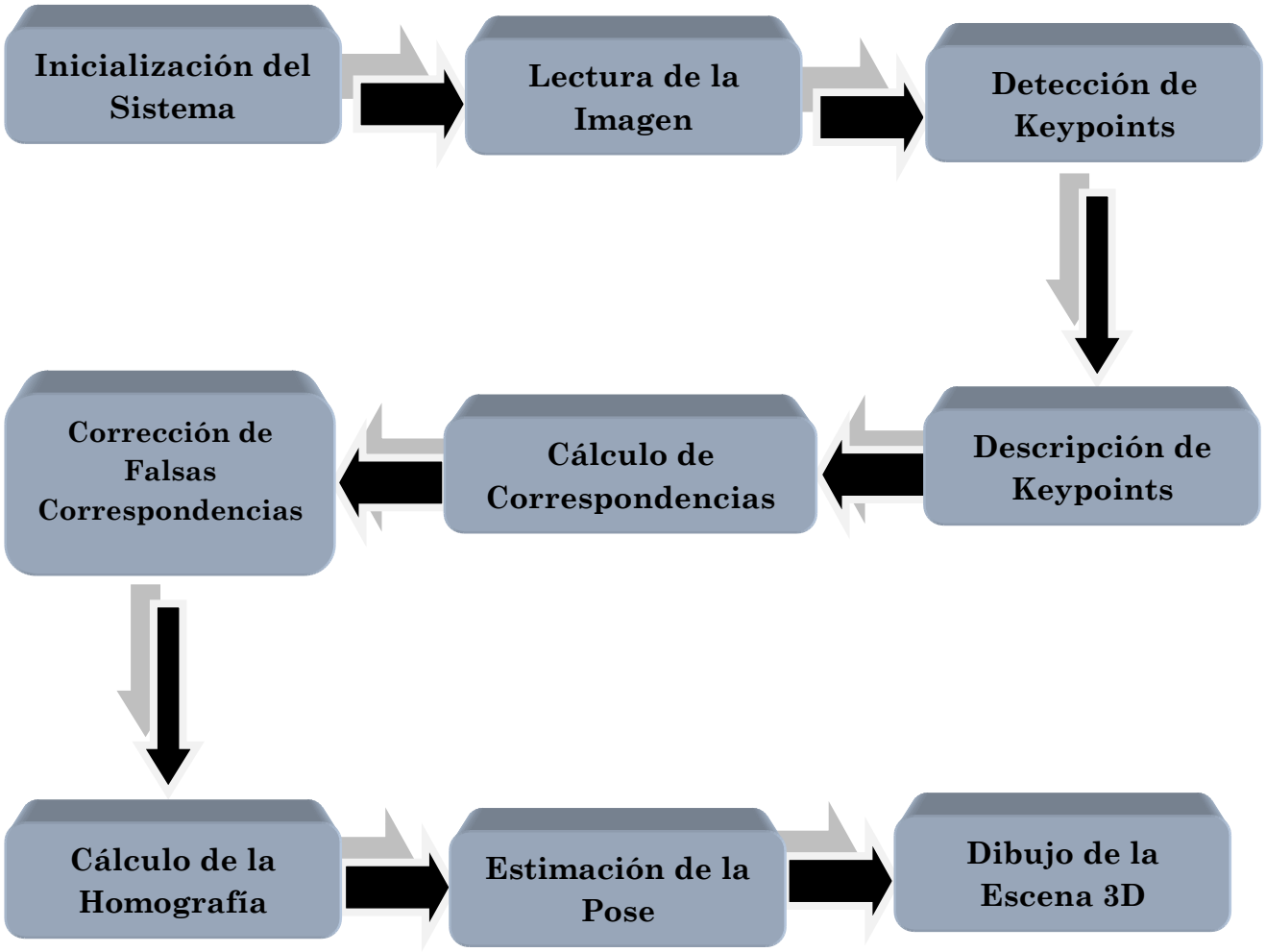

Figura 9. Estructura funcional del Sistema

No	Fase	Descripción
1	Inicialización del Sistema	En esta fase, se prepara el sistema para el análisis de la imagen. Se inicializan los datos para la captura ya sea de un video o de una imagen. Se eligen los algoritmos de detección y extracción de características invariantes en imágenes.
2	Lectura de la Imagen	El sistema lee el fotograma de video o la imagen que se le está mostrando en ese momento. Así como el patrón con el cual comparar.

3	Detección de *Keypoints*	El sistema busca en la imagen y en el patrón las características unívocas de cada uno llamados *keypoints*. El resultado de esta fase dependerá del algoritmo de detección seleccionado en la *fase 1.*
4	Descripción de *Keypoints*	El sistema realiza la descripción de los *keypoints*, tanto como los de la imagen de búsqueda, como los de la imagen patrón. El resultado de esta fase dependerá del algoritmo de descripción seleccionado en la *fase 1.*
5	Cálculo de correspondencias	El sistema establece correspondencias entre *keypoints* de la imagen de búsqueda y los de la imagen patrón, de acuerdo a la descripción que se realizó de los mismos en la fase anterior.
6	Corrección de Falsas Correspondencias	Es normal, que en la fase anterior se cometan errores. Por lo que en esta fase se realiza una limpieza de los mismos, eliminando las falsas correspondencias y rectificando aquellas que no se tuvieron en cuenta en la fase anterior.
7	Cálculo de la Homografía	A partir del cómputo realizado en la fase anterior, se procede a realizar la estimación de la transformación homográfica entre los puntos característicos de la imagen de búsqueda y los de la imagen patrón.
8	Estimación de la Pose	Con los datos obtenidos y procesados se procede a calcular la estimación de la pose, obteniendo la matriz de rotación y el vector de traslación entre la cámara y la imagen.
9	Dibujo de la Escena 3D	Los datos obtenidos en el paso anterior son interpretados por el sistema y usados para dibujar una escena 3D.

Tabla 1. Descripción de las Fases de la solución propuesta

2.3 Captura de requisitos

El levantamiento de requerimientos es una etapa esencial en el arranque de todo proyecto de desarrollo de software y debe realizarse efectivamente para poder aumentar las posibilidades de éxito de los proyectos. A continuación se muestran los requisitos funcionales (RF) y los no

funcionales (RFN). Los requisitos funcionales describen los servicios (funciones) que se esperan del sistema (25). Los requisitos no funcionales son restricciones sobre los requisitos funcionales identificados (25).

2.3.1 Requisitos funcionales

RF 1. Leer la imagen de búsqueda y la imagen de patrón a analizar.

RF 2. Convertir la imagen de entrada a escala de grises para su posterior análisis.

RF 3. Aplicar algoritmos de detección y extracción de características de imágenes.

RF 4. Realizar el cálculo de las correspondencias entre los descriptores de la imagen de entrada con los del patrón.

RF 5. Refinar las correspondencias para obtener mejores resultados.

RF 6. Realizar el cálculo de la homografía entre la imagen y el patrón de entrada.

RF 7. Realizar el cómputo necesario para la estimación de la pose.

RF 8. Mostrar una escena 3D.

2.3.2 Requisitos no funcionales

Requisitos de Usabilidad:

1. El componente implementado podrá ser utilizado por cualquier usuario con conocimientos básicos de RA, que esté interesado en la generación de información aumentada basándose en los algoritmos de extracción de características naturales en imágenes.

Requisitos no funcionales de Restricciones en el Diseño e Implementación:

1. Lenguaje de programación *C++.*
2. IDE de desarrollo: *Qt Creator*.
3. Bibliotecas de clases: *Qt*, *OpenCV*.

Hardware:

Mínimo recomendado:

1. Procesador Pentium 4 a 2.7 GHz o superior.

2. Memoria RAM de 1GB.
3. Espacio en disco duro de 512 MB.

2.4 Fase de exploración

En esta fase los clientes seleccionan las historias de usuarios (HU) que desean que estén para la primera entrega. Cada HU describe una de las funcionalidades que el programa tendrá. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, la tecnología y las prácticas a ser utilizadas durante el proyecto. En algunos casos se utiliza un prototipo para probar la nueva tecnología y explorar algunos aspectos de la arquitectura a implementar. La duración de esta fase puede extenderse desde unas pocas semanas a varios meses dependiendo de la adaptación del equipo de desarrollo (26).

2.4.1 Historias de usuario

La metodología XP utiliza las HU para registrar los requerimientos de los clientes y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación. También son utilizadas para poder crear las pruebas de aceptación. Por lo general se necesitan uno o más pruebas de aceptación para verificar que la historia ha sido implementada correctamente (26).

A continuación se expone un ejemplo de descripción de una de las HU con mayor relevancia. El resto de las HU se encuentran plasmadas en el anexo 1.

Historia de Usuario	
Número: 6	**Usuario:** Cliente
Nombre: Cálculo de la Homografía	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2 semanas	**Iteración asignada:** 3
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: El sistema debe ser capaz de realizar el cómputo de la matriz de homografía entre la imagen de búsqueda y la imagen patrón.	
Observaciones: Con el correcto desarrollo de esta funcionalidad, se garantiza una optimización en cuanto al cálculo de correspondencias se refiere. Puesto que	

solamente se tendrán en cuenta aquellas correspondencias que garanticen el correcto cálculo de dicha homografía. Si el resultado de tener en cuenta solamente correspondencias que cumplan con esta condición devuelve un número mínimo de estas, el sistema interpretará que la imagen de búsqueda no guarda relación con la imagen patrón.

Tabla 2. Historia de Usuario 6

2.5 Fase de Planificación

El objetivo de esta fase es fijar la prioridad de cada una de las HU y establecer el contenido de la primera entrega. Los programadores estiman cuanto esfuerzo requiere cada HU y se establece el cronograma. La duración del calendario para la entrega de la primera liberación no suele superar los dos meses. La duración de la fase de planificación en sí no toma más de dos días (26).

2.5.1 Estimación de esfuerzo por historias de usuario

El cliente es el encargado de establecer la prioridad de cada historia de usuario y los programadores tienen la responsabilidad de definir una estimación del esfuerzo necesario para dar cumplimiento a cada una de ellas. Para realizar dicha estimación se toma como medida el punto de estimación, el cual equivale a una semana de trabajo ideal de programación, donde los programadores trabajan el tiempo planeado sin ningún tipo de interrupción.

Para el desarrollo del componente se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, alcanzando los resultados que se muestran en la tabla 3.

Historia de Usuario	Puntos de Estimación
Lectura de imagen	1 semana
Detección de *keypoints*	1 semana
Descripción de *keypoints*	1 semana
Cálculo de correspondencias	1 semana
Refinamiento de correspondencias	2 semana
Cálculo de la *homografía*	2 semana

Estimación de la pose	2 semana
Dibujo de la escena 3D	2 semana

Tabla 3. Estimación de esfuerzo por HU.

2.5.2 Plan de iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que son los clientes quienes deciden qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores (27).

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, el desarrollo del sistema se distribuyó en 4 iteraciones, las cuales se describen a continuación de manera más detallada:

1. **Primera Iteración**

Tuvo como objetivo la implementación de las siguientes HU:

- HU 1. Lectura de imagen
- HU 2. Detección de *keypoints*

Al concluir esta iteración, se preparó al sistema para el desarrollo de la siguiente iteración. Una vez terminada esta iteración se contó con la primera versión de prueba del sistema.

2. **Segunda Iteración**

En esta iteración se llevó a cabo la implementación de las siguientes HU:

- HU 3. Descripción de *keypoints*

- HU 4. Cálculo de correspondencias

De igual manera, al concluir esta iteración, se procedió al desarrollo de la siguiente iteración. El resultado alcanzado en esta iteración nos proporcionó una segunda versión de prueba del sistema.

3. Tercera Iteración

Tiene como objetivo la implementación de las siguientes HU:

- HU 5. Refinamiento de correspondencias.
- HU 6. Cálculo de la homografía.

Al concluir esta iteración se resolvió ya gran parte del problema que fue planteado en el momento que surgió la necesidad de implementar el componente. Es válido señalar que el desarrollo de la HU 6 depende fuertemente del desarrollo de las HU anteriores. Hasta aquí todo es un proceso secuencial, en el que la salida obtenida en cada HU es utilizada en la siguiente. Con el cumplimiento de esta iteración se contó con una tercera versión de prueba del sistema.

Cuarta Iteración

En esta última iteración se llevó a cabo la implementación de las siguientes HU:

- HU 7. Estimación de la pose.
- HU 8. Escena 3D.

Estas HU son integradas con el resultado de las iteraciones antes descritas y como beneficio de esta integración se obtuvo el componente de software de RA, basado en la detección de características naturales en imágenes.

2.5.3 Plan de duración de las iteraciones

No Iteración	Historia de Usuario	Duración Total de Iteraciones
Iteración 1	Lectura de imagen Detección de *keypoints*	2 semanas
Iteración 2	Descripción de *keypoints* Cálculo de correspondencias	2 semana

Iteración 3	Refinamiento de correspondencias. Cálculo de la homografía	4 semana
Iteración 4	Estimación de la pose Escena 3D	4 semana

Tabla 4. Plan de duración de iteraciones

2.5.4 Plan de entrega

Una vez definido el plan de iteraciones, se pasa a elaborar el plan de entrega. Este tiene como principal objetivo dejar plasmado el orden en que se realizaran las diferentes entregas intermedias y la entrega final del componente.

No Iteración	Duración	Fecha Inicio	Fecha Final
Iteración 1	2 semanas	11/2/2014	25/2/2014
Iteración 2	2 semanas	26/2/2014	12/3/2014
Iteración 3	4 semanas	13/3/2014	10/4/2014
Iteración 4	4 semanas	11/4/2014	9/5/2014

Tabla 5. Plan de entrega

2.6 Diseño de la solución propuesta

2.6.1 Arquitectura de Software

El estilo arquitectónico que se le aplicó al sistema fue el conocido estilo *en capas*, específicamente en *tres capas* el cual pertenece a la familia de estilos de *llamada y retorno* (28). Este presenta múltiples ventajas, entre las cuales se pueden mencionar las siguientes:

- ✓ Soporta un diseño basado en niveles de abstracción crecientes, lo que permite a los implementadores la partición de un problema complejo en una secuencia de pasos.
- ✓ Admite naturalmente optimizaciones y refinamientos.
- ✓ Proporciona una amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la

medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

A continuación se muestra el diseño arquitectónico del sistema y una breve descripción del mismo. En este se puede apreciar la forma en que se establece la comunicación entre las diferentes capas, en este caso la arquitectura que se definió fue en tres capas.

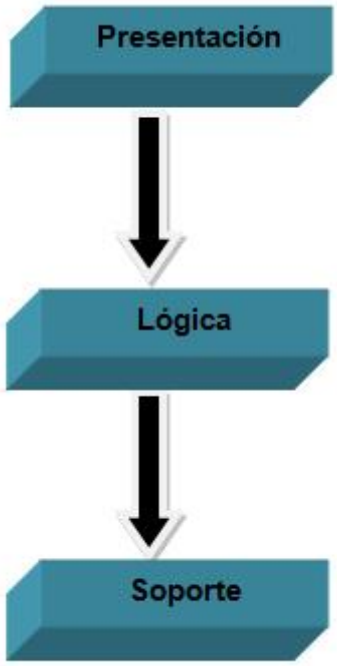

Figura 10 Diseño arquitectónico del sistema.

- ✓ **Presentación:** En esta capa se ubican las interfaces del sistema, las cuales se encargan de efectuar la interacción con el usuario, validan los datos que este introduce y solicitan a la capa de Lógica el correspondiente procesamiento de estos.
- ✓ **Lógica:** En esta capa se encuentran aquellas clases encargadas de efectuar la lógica del cómputo de los datos introducidos por el usuario al sistema. Para que la capa Lógica pueda realizar exitosamente los cómputos necesarios, esta se apoya en la capa de Soporte.
- ✓ **Soporte:** En esta capa se encuentran los elementos necesarios para que la capa de Lógica desarrolle correctamente todas sus funciones. Esta capa enmarca elementos como bibliotecas *(OpenCv) y* datos constantes (parámetros intrínsecos de la cámara, GeometryTipe)

2.6.2 Patrones de diseño

Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y las ventajas e inconvenientes de su uso (29).

Para el diseño de las clases se utilizó el conjunto de patrones de diseño *GRASP* y *GOF*, que intervienen en la asignación de responsabilidades, estructura y comportamiento respectivamente. A continuación se indicarán aquellos que fueron aplicados.

2.6.3 Patrones GRASP

El *Patrón Experto* se utilizó más que cualquier otro al asignar responsabilidades; pues es un principio básico que suele utilizarse en el diseño orientado a objetos. Dio origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican al concepto que representa, por lo que ofrece una analogía con el mundo real (30). Permitió asignarles las responsabilidades necesarias a desarrollar a cuyas clases contienen la información para realizarlas. En este caso cada clase se diseñó según su función en el proceso.

También se hace uso del *Patrón Creador,* que tiene como objetivo guiar la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador encargado de producir el objeto en dependencia del contexto actual (30).

Otro patrón que se pone de manifiesto es el *Bajo Acoplamiento*, que trata de establecer la menor dependencia posible entre las clases, de modo que sea más adaptable a cambios futuros. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, ya que presentan los siguientes problemas:

- Los cambios de las clases relacionadas ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

No puede considerarse en forma independiente de otros patrones como Experto o Alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades (30). El hecho de contar con un bajo acoplamiento asegura *alta cohesión* donde no toda la responsabilidad recae en una sola clase.

2.6.4 Patrones GOF

De los patrones GOF se hace uso del *Singleton,* el cual garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (29). El empleo de este patrón presenta las siguientes características o ventajas:

1. Acceso controlado a la única instancia. Puesto que la clase *singleton* encapsula su única instancia, puede tener un control estricto sobre cómo y cuándo acceden a ella los clientes.
2. Permite el refinamiento de operaciones y la representación. Ya que se puede crear una subclase de la clase *singleton*, y es fácil configurar una aplicación con una instancia de esta clase extendida.
3. Más flexible que las operaciones de clase. Otro forma de empaquetar la funcionalidad de un singleton es usar operaciones de clase (es decir, funciones miembro estáticas en C++). Para esta técnica de C++ se dificulta cambiar un diseño para permitir más de una instancia de una clase. Además, las funciones miembro estáticas en C++ nunca son virtuales, por lo que las subclases no las pueden redefinir polimórficamente.

Otro patrón GOF que se empleó fue el *Facade*, el cual tiene como propósito brindar una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar (29). Como consecuencias de aplicar este patrón se destacan:

1. Oculta a los clientes los componentes del subsistema.
2. Promueve un débil acoplamiento entre el subsistema y sus clientes.
3. No impide que las aplicaciones usen las clases del subsistema en caso de que sea necesario.

2.6.5 Tarjetas CRC

A continuación se muestran las tarjetas CRC, las cuales son fichas, una por cada clase, en las que se escriben brevemente las responsabilidades de cada una e informa además su relación

respectiva con otras clases, las cuales garantizan poder llevar a cabo dichas responsabilidades.

Tarjeta CRC	
Clase: Administrator	
Responsabilidad	**Colaboración**
Es la clase controladora de la aplicación, la misma se encarga de: • Capturar la imagen patrón. • Identificar la vía por la cual el usuario realizará el análisis de las imágenes (Video, Cámara o Imágenes existentes en el disco duro de la pc). • Capturar la imagen de búsqueda. • Capturar los algoritmos que se aplicarán en el proceso de detección y extracción de características en imágenes.	Analyzer AugmentedReality

Tabla 6.Tarjeta CRC de la clase Administrator.

Tarjeta CRC	
Clase: Analyzer	
Responsabilidad	**Colaboración**
Es la clase encargada de: • Recibir los datos enviados por Administrador y maneja el procesamiento de los mismos. • Devolver los datos ya procesados a la interfaz.	Compute GeometryTipe

Tabla 7.Tarjeta CRC de la clase Analyzer.

Tarjeta CRC	
Clase: Compute	
Responsabilidad	**Colaboración**
Esta clase es responsable de: • Realizar el proceso de detección y extracción de las características naturales en las imágenes. • Buscar las correspondencias entre la imagen de búsqueda y la imagen patrón. • Realizar los cálculos necesarios para la obtención de la homografía de la imagen patrón respecto a la imagen de búsqueda.	Pattern

Tabla 8.Tarjeta CRC de la clase Compute.

Tarjeta CRC	
Clase: GeometryTipe	
Responsabilidad	**Colaboración**
Esta clase se encarga de: • Contener estructuras de datos, las cuales serán de utilidad para la generación de la Realidad Aumentada.	AugmentedReality

Tabla 9.Tarjeta CRC de la clase GeometryTipe.

Tarjeta CRC	
Clase: Pattern	
Responsabilidad	**Colaboración**
Esta clase se encarga de: • Contener la información referente a la imagen patrón. • Calcular la pose de la cámara respecto a la imagen patrón	GeometryTipe CameraCalibration

Tabla 10.Tarjeta CRC de la clase Pattern.

Tarjeta CRC	
Clase: CameraCalibration	
Responsabilidad	**Colaboración**
Esta clase se encarga de: • Proveer la información necesaria referente a la cámara, específicamente a su calibración.	

Tabla 11.Tarjeta CRC de la clase Compute.

Tarjeta CRC	
Clase: AugmentedReality	
Responsabilidad	**Colaboración**
Esta clase se encarga de: • Generar la Realidad Aumentada	GeometryTipe CameraCalibration

Tabla 12.Tarjeta CRC de la clase Compute.

2.7 Conclusiones Parciales

Al culminar este capítulo, quedaron claros los elementos a tener en cuenta para el desarrollo del componente que da respuesta a esta investigación. Se definieron los requisitos funcionales y no funcionales con los que debe contar el componente. Se planificó el tiempo y las tareas a realizar para el desarrollo de la solución propuesta. Además se expuso la lógica de funcionamiento del componente, lo que permitió garantizar su correcto desarrollo. La implementación de esta lógica permitió demostrar que la solución propuesta resuelve el problema de la extracción de características naturales en imágenes. Se definió una estructura para dicho componente que contiene características y funcionalidades que permiten la combinación de algoritmos de detección y extracción de características en imágenes. Con esto se garantizó la variación del resultado atendiendo a los diferentes cambios que puedan estar presentes a la hora de analizar la imagen.

CAPÍTULO 3 IMPLEMANTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 INTRODUCCIÓN

En el presente capítulo se expondrán los detalles correspondientes a la implementación del sistema. De igual manera se hará alusión a las tareas de ingeniería que rigieron el desarrollo de la solución. Se describirán las pruebas realizadas al sistema y los principales resultados obtenidos, con el objetivo de demostrar la eficacia del sistema propuesto basado en la detección de características naturales en imágenes.

3.2 Estándar de Codificación

Se deben establecer reglas para los programadores y estas deben ser seguidas estrictamente, estas reglas son conocidas como el estándar de codificación. En un equipo de desarrollo los programadores acuerdan un estándar para el código, dejando de lado estilos de programación particulares. Todos deben conocer y seguir el estándar de manera tal que al final el sistema parezca ser programado por una sola persona (26). El empleo de esta buena práctica de desarrollo de software es gran importancia y se traduce en una mayor calidad de desarrollo.

El estándar de codificación conduce a una mayor coherencia entre el código personal y el de los compañeros del equipo, y esto a su vez permite generar un código más fácil de entender que facilita su desarrollo y mantenimiento. Esto reduce el costo total de las aplicaciones a crear.

Para el desarrollo de este componente, se estableció un estándar de codificación propio. A continuación se muestran las pautas que lo conforman:

1. Los nombres de las clases, métodos y variables serán escritos en idioma inglés.
2. Los nombres de las clases y estructuras empezarán con letra inicial mayúscula.
3. En caso de que los nombres de las clases y estructuras sean compuestos, cada palabra comenzará con mayúscula, tal y como propone el estándar *Camel Case*.
4. Los nombres de los métodos y variables se escribirán en minúscula.
5. En caso de que los nombres de los métodos y variables sean compuestos, las siguientes palabras comenzarán con letra mayúscula.
6. Los comentarios de implementación se delimitarán entre /*…*/ o se utilizará la variante //.

7. Los comentarios de documentación se delimitarán por /**…*/.

3.3 Fase de Producción

La fase de producción requiere pruebas extras y chequeos de la ejecución del sistema antes de que sea entregado al cliente. En ésta fase, se pueden encontrar nuevos cambios y se decide si serán incluidos en la entrega actual. Durante esta fase, las iteraciones pueden necesitar ser recortadas de tres semanas a una semana. Después que la primera entrega es producida para el uso del cliente, el proyecto *XP* debe mantener el sistema en producción corriendo mientras que también se estén produciendo nuevas iteraciones (31).

En el capítulo anterior se describieron y distribuyeron por iteraciones las HU a desarrollar, pero esta información por sí sola no brinda suficientes elementos que permitan su análisis y desarrollo. Es por este motivo que para cada iteración se establecen las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. Consecuentemente se descomponen las HU en tareas de programación o ingeniería, las mismas pueden ser escritas en un lenguaje técnico debido a que son para uso estricto de los programadores, donde el usuario no tiene que necesariamente comprenderlas. A continuación se muestran las tareas de ingeniería que quedaron definidas por cada iteración e HU correspondiente.

Tareas de la Iteración 1	
Historia de Usuario	**Tareas**
Lectura de imagen	1. Desarrollar una interfaz para la captura de las imágenes de entrada. 2. Validación de los datos capturados.
Detección de *keypoints*	1. Desarrollo de una interfaz que muestre los algoritmos de detección de *keypoints.* 2. Validación de la selección del algoritmo de detección de características naturales

	seleccionado por el usuario. 3. Implementación de la funcionalidad encargada de realizar el análisis de las características naturales de las imágenes.

Tabla 13.Tareas de Ingeniería de la iteración 1.

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 1
Nombre de la tarea: Desarrollar una interfaz para la captura de las imágenes de entrada.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2 días
Fecha de inicio: 11/2/2014	**Fecha final:** 13/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza el diseño e implementación de la interfaz que servirá para la carga de las imágenes a ser analizadas.	

Tabla 14.Tarea de Ingeniería 1 de la HU 1.

Las restantes tareas de ingeniería se encuentran en el anexo 2.

3.3.1 Pruebas

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

La metodología ágil XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de

aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente. Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto el cliente es la persona adecuada para diseñar las pruebas de aceptación (22).

3.3.2 Pruebas de aceptación

Las pruebas de aceptación fueron creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente especificó uno o diversos escenarios para comprobar que una historia de usuario fue correctamente implementada. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. A continuación se muestran las pruebas más significativas realizadas a cada una de las Historias de Usuario.

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Realizar la lectura de una imagen	
Condiciones de ejecución: El usuario debe haber inicializado el sistema.	
Pasos de Ejecución: 1. El usuario selecciona la imagen patrón a ser analizada. 2. El usuario indica al sistema la imagen de búsqueda a analizar.	
Resultados esperados: 1. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 2. Si el usuario no ha indicado la imagen patrón a analizar, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si se efectuaron correctamente los pasos 1 y 2, el sistema debe ser capaz de cargar las imágenes patrón y de búsqueda correspondientes.	
Evaluación de la prueba: Satisfactoria	

Tabla 15. Caso de prueba HU1_P1

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** 2
Nombre: Realizar la detección de los *keypoints.*	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características.	
Pasos de Ejecución: 1. El usuario selecciona la imagen patrón a ser analizada. 2. El usuario indica al sistema la imagen de búsqueda a analizar.	
Resultados esperados: 1. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 2. Si el usuario no ha indicado la imagen patrón a analizar, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si se efectuaron correctamente los pasos 1 y 2, el sistema debe ser capaz de detectar lo *keypoints* en ambas imágenes. Este resultado puede ser más legible cuando se realice el cálculo y el refinamiento de las correspondencias.	
Evaluación de la prueba: Satisfactoria	

Tabla 16. Caso de prueba HU2_P1

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** 3
Nombre: Realizar la descripción de los *keypoints*.	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características.	
Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 3. El usuario le indica al sistema la imagen patrón. 4. El usuario indica la imagen de búsqueda.	
Resultados esperados: 1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor, el sistema debe lanzar un mensaje indicando el error cometido. 2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error. 4. Si se efectuaron correctamente los pasos 1,2 y 3, el sistema debe ser capaz de describir lo *keypoints* detectados en ambas imágenes. Este resultado puede ser más legible cuando se realice el cálculo y el refinamiento de las correspondencias.	
Evaluación de la prueba: Satisfactoria	

Tabla 17. Caso de prueba HU3_P1

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** 4
Nombre: Realizar el cálculo de correspondencias entre el patrón y la imagen de	

entrada.
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características.
Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 2. El usuario le indica al sistema la imagen patrón. 3. El usuario indica la imagen de búsqueda.
Resultados esperados: 1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor, el sistema debe lanzar un mensaje indicando el error cometido. 2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error. 4. Si se efectuaron correctamente los pasos 1,2 y 3, el sistema debe ser capaz de realizar el cálculo de las correspondencias entre las dos imágenes.
Evaluación de la prueba: Satisfactoria

Tabla 18. Caso de prueba HU4_P1

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** 5
Nombre: Realizar el refinamiento de las correspondencias.	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características.	

Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 2. El usuario le indica al sistema la imagen patrón. 3. El usuario indica la imagen de búsqueda.
Resultados esperados: 1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor, el sistema debe lanzar un mensaje indicando el error cometido. 2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error. 4. Si se efectuaron correctamente los pasos 1,2 y 3, el sistema debe ser capaz de realizar el cálculo de las correspondencias entre las dos imágenes.
Evaluación de la prueba: Satisfactoria

Tabla 19. Caso de prueba HU5_P1

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** 6
Nombre: Realizar el cálculo de la homografía.	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características.	
Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 2. El usuario le indica al sistema la imagen patrón.	

3. El usuario indica la imagen de búsqueda.

Resultados esperados:

1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor, el sistema debe lanzar un mensaje indicando el error cometido.
2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error.
3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error.
4. Si se efectuaron correctamente los pasos 1,2 y 3, el sistema debe ser capaz de realizar el cálculo de la homografía correspondiente entre la imagen de búsqueda y la imagen patrón.

Evaluación de la prueba: Satisfactoria

Tabla 20. Caso de prueba HU6_P1

Caso de Prueba de Aceptación	
Código: HU7_P1	**Historia de Usuario:** 7
Nombre: Realizar el computo de la pose 3D.	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características. Se debe de haber realizado correctamente el cálculo de la homografía.	
Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 2. El usuario le indica al sistema la imagen patrón. 3. El usuario indica la imagen de búsqueda.	

Resultados esperados: 1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor, el sistema debe lanzar un mensaje indicando el error cometido. 2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error. 4. Si se efectuaron correctamente los pasos 1,2 y 3 y el resultado arrojado por la homografía es el correcto, entonces el sistema debe ser capaz de realizar el cálculo de la pose 3D correspondiente entre la cámara y la imagen de búsqueda.
Evaluación de la prueba: Satisfactoria

Tabla 21. Caso de prueba HU7_P1

Caso de Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario:** 8
Nombre: Realizar el montaje de la escena 3D.	
Condiciones de ejecución: El usuario debe haber cargado la imagen patrón y la imagen de búsqueda, además, debe haber elegido un algoritmo de detección y uno de extracción de características. Se debe haber realizado satisfactoriamente el cálculo de la pose de la cámara.	
Pasos de Ejecución: 1. El usuario selecciona los algoritmos de detección y extracción de características. 2. El usuario le indica al sistema la imagen patrón. 3. El usuario indica la imagen de búsqueda.	
Resultados esperados: 1. Si el usuario no ha seleccionado un par de algoritmos detector y descriptor,	

el sistema debe lanzar un mensaje indicando el error cometido. 2. Si el usuario no ha indicado la imagen patrón, el sistema debe mostrarle un mensaje indicándole el correspondiente error. 3. Si el usuario no ha indicado la imagen de búsqueda, el sistema debe lanzar un mensaje, indicándole el correspondiente error. 4. Si se efectuaron correctamente los pasos 1,2 y 3 y si se realizó el cálculo de la pose 3D correspondiente entre la cámara y la imagen de búsqueda, entonces el sistema debe ser capaz de mostrar una escena en 3D sobre la imagen de búsqueda.
Evaluación de la prueba: Satisfactoria

Tabla 22. Caso de prueba HU8_P1

En la revisión de las funcionalidades se realizaron cuatro iteraciones de pruebas, detectándose en la primera de ellas un total de dos no conformidades, en la segunda y tercera se detectaron tres no conformidades en cada una. La mayoría de estas no conformidades fueron errores de validación, los cuales fueron solucionados satisfactoriamente. En la cuarta iteración no se encontraron no conformidades alcanzando resultados satisfactorios.

3.4 Resultados obtenidos

En la figura 11 se muestran dos imágenes, a las cuales se les realizó el proceso de análisis de sus respectivas características naturales para validar la robustez de los resultados obtenidos por el componente implementado. Estas imágenes son estándares, propuestas por Krystian Mikolajczyk en (32) para realizar evaluaciones de los métodos de detección y extracción de las características naturales en imágenes.

Imagen Patrón Imagen de Búsqueda

Figura 11. Imágenes utilizadas de prueba para la detección y extracción de características naturales en imágenes.

Con este componente se logró determinar las correspondencias entre dos imágenes, la figura 12 muestra las correspondencias obtenidas tras aplicar el algoritmo SIFT como detector y extractor de características naturales. Es válido mencionar que el componente implementado brinda la posibilidad de seleccionar métodos alternativos para la detección y extracción de las características invariantes de las imágenes utilizando los métodos SIFT, SURF, ORB y FREAK. Por tal motivo solo se pone de ejemplo en las figuras un solo método o una combinación, porque el usuario puede seleccionar los métodos alternativos para aplicar las mismas pruebas descritas en esta sección y comprobar por sí mismo los resultados

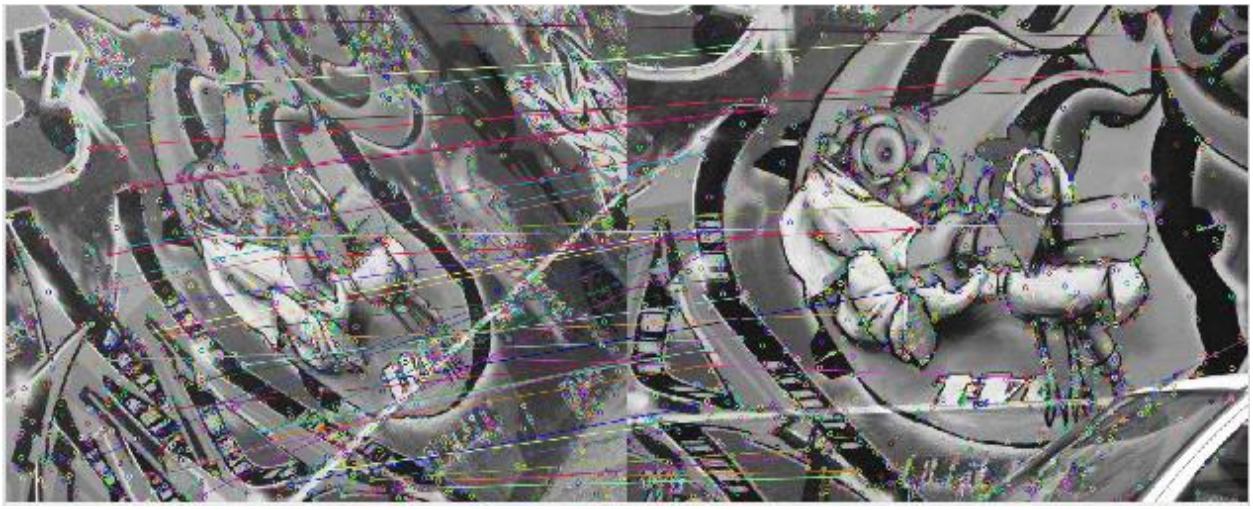

Figura 12. Cálculo de correspondencias con SIFT como algoritmo de detección y extracción.

Luego del cálculo de las correspondencias, se procede al cálculo de la homografía con el

objetivo de conocer la transformación geométrica que debe aplicársele a una imagen *A* con respecto a una imagen *B*, de tal forma que se obtenga una imagen *C*, donde *C* es la imagen *A* vista con la misma perspectiva con la que se observa a *B.* Haciendo uso de funcionalidades que brinda *OpenCv,* tales como *findHomography() y warpPerspective()*, se obtuvieron los resultados que se muestran en la figura 13.

Imagen A

Imagen B

Imagen C

Figura 13. Cálculo de homografía con SIFT como algoritmo de detección y extracción

También fue posible brindar al usuario la posibilidad de elegir los algoritmos de detección y extracción de características que este estime conveniente, y de esta manera poder realizar una valoración de los resultados y elegir los algoritmos más robustos.

Pruebas experimentales demostraron que no es posible realizar una combinación entre SIFT como algoritmo de detección y ORB como algoritmo de extracción, pues una combinación de

estos arrojaron resultados de incompatibilidad.

Desde la bibliografía consultada es posible conocer que tanto SURF como SIFT son los métodos que producen resultados más robustos en la detección y extracción de características. Aunque ofrecen resultados robustos, estos están patentados, o sea que para su utilización se tendría que pagar por sus respectivas patentes. Como respuesta a este inconveniente se decidió incorporar al sistema los algoritmos ORB y FREAK, los cuales son libres de patentes.

Del concepto de RA analizado en el primer capítulo se conoce que las aplicaciones de RA poseen restricciones de tiempos de ejecución interactivos y que la información visual introducida debe ser visualmente coherente con el contexto donde se introduce. Para conocer el desempeño de estos métodos con el objetivo conocer si cumplen tales restricciones, se realizó una prueba de combinación de los diferentes métodos, para valorar los resultados ofrecidos, a continuación se muestra una gráfica en función del número de correspondencias dado el par A-B, donde A es el algoritmo detector y B el algoritmo extractor. Es válido aclarar que cuando se realizaron estas pruebas estadísticas, se forzó al sistema para que los algoritmos detectores se limitaran a la búsqueda de una mínima cantidad de *keypoints,* de manera que el resultado de aplicar una homografía con los resultados que estos arrojaban fuera lo más exacta posible tal y como se pudo apreciar en la figura anterior. En esta prueba más correspondencias indican un valor mayor de robustez en el cálculo de la homografía en una etapa posterior.

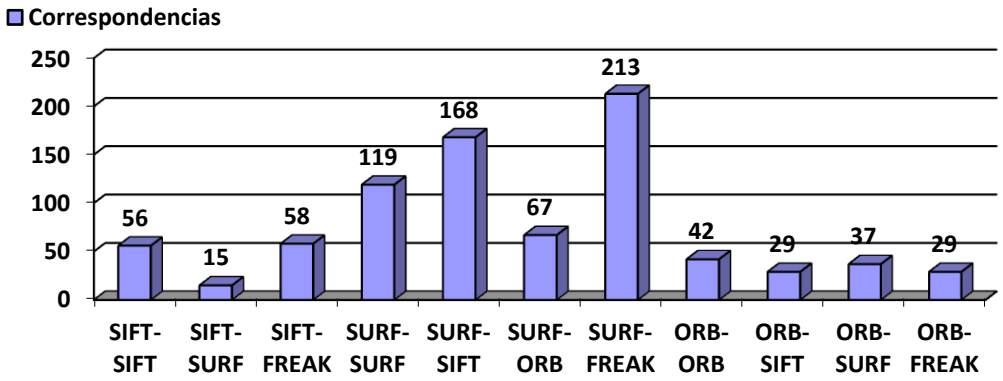

Tabla 23. Cantidad de correspondencias con combinaciones de algoritmos de detección y extracción.

La gráfica anterior mostró que la combinación de algoritmos dada por *SURF-FREAK* arrojó el

mayor número de correspondencias, demostrando que es posible obtener mejores resultados cuando se combinan algoritmos diferentes. Según estos resultados de correspondencias, es indiscutible que tanto *SURF* como *SIFT* son los algoritmos más robustos en la búsqueda de correspondencias, sin embargo, aunque *ORB* ofrece resultados discretos comparados con la cantidad de correspondencias dadas por *SIFT* y *SURF*, es posible alcanzar el objetivo final haciendo uso de este, lo que lo convierte en una buena alternativa frente a estos algoritmos patentados.

Para analizar el rendimiento en términos de recursos computacionales de los métodos implementados, se realizó un cálculo del tiempo de ejecución para los datos de prueba. En la tabla 24 se muestra que las combinaciones de SIFT y SURF aunque arrojaron resultados robustos son los más consumidores de recursos y alcanzan tiempos de ejecución de hasta 21 milisegundos en esta etapa. Este cálculo se puede hacer 34 veces por segundo sin tener en cuenta el cálculo de la homografía y la complejidad de la información visual que se va a introducir lo cual conduce a tiempos de ejecución por debajo de los 15 cuadros por segundo, por tanto no cumplen con el objetivo de la interactividad. Por otro lado ORB y FRAK alcanzan resultados robustos en mucho menos tiempo (2,5 y 0,9 respectivamente) lo cual conduce a resultados satisfactorios.

Para comprobar la coherencia visual de la información aumentada se realizó una prueba con el método SIFT que ofrece los mejores resultados y con el ORB que ofrece también resultados satisfactorios.

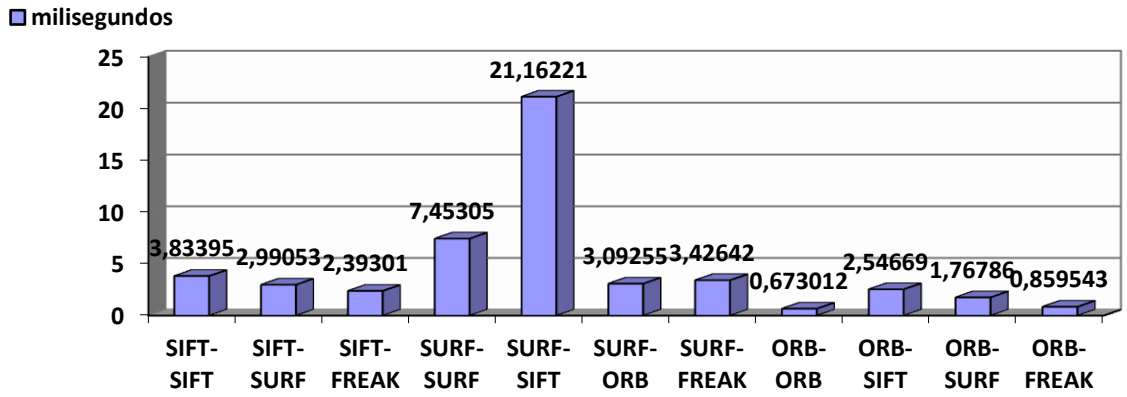

Tabla 24. Tiempos de ejecución

Cómo muestra la tabla 24, los tiempos de ejecución principalmente correspondientes a los pares de algoritmos donde se involucran a SIFT y a SURF, son mayores con respecto a los

arrojados por ORB y FREAK. Esto demuestra que tanto ORB como FREAK constituyen una alternativa a SIFT y SURF.

La figura 14 ilustra los resultados finales para verificar la coherencia visual de la información aumentada, tras aplicar ORB y SIFT como algoritmos de detección y extracción de características naturales en imágenes. Para esta prueba se seleccionó el algoritmo SIFT, ya que este es uno de los que ofrece resultados más robustos en la búsqueda de correspondencias, también se seleccionó el ORB, que ofrece menos correspondencias pero su tiempo de ejecución es inferior al de los demás métodos.

(a) (b)

Figura 14. Coherencia visual de la información aumentada (a) SIFT (b) ORB.

En la figura 14 se puede apreciar que la coherencia visual es de a el alto nivel de competitividad de ORB ante SIFT. La coherencia visual de la información aumentada cuando se utiliza el método ORB es equivalente a la producida por el método SIFT. Este resultado demuestra que el método ORB cumple con las restricciones de las aplicaciones de RA. Por lo tanto se puede decir que ORB es un candidato fuerte para el desarrollo de aplicaciones de RA basadas en la detección de características naturales en imágenes.

3.5 Conclusiones Parciales

Con la finalización de este capítulo se logró definir el estándar de codificación, posibilitando una mayor comprensión del código y facilitando el trabajo a la hora de ser reutilizado. Las pruebas de aceptación permitieron guiar el desarrollo del sistema hasta obtener los resultados esperados. Además se pudo sentar la base para la realización de un análisis de los resultados obtenidos.

CONCLUSIONES GENERALES

La realización de la presente investigación, permitió implementar un componente de software donde se unifica en un mismo marco de trabajo varios algoritmos de detección y extracción de características naturales en imágenes, para adquirir información geométrica tridimensional para Realidad Aumentada. De esta manera se logró dar solución al problema planteado al inicio de la investigación y sobre todo, se pudo vencer el objetivo que se persiguió con este trabajo. Consecuentemente, al finalizar esta investigación, el autor llegó a las siguientes conclusiones:

- Se demostró que los algoritmos SURF y SIFT, son los que ofrecen los resultados más robustos en la búsqueda y correspondencia de características naturales en las imágenes, pero son algoritmos muy consumidores de recursos computacionales que no permiten la tasa de refresco que necesita una aplicación de Realidad Aumentada.
- Se demostró que el uso de los algoritmos ORB y FREAK, a pesar de ofrecer menor cantidad de características y correspondencias, constituyen una alternativa viable al uso de SURF y SIFT, debido a que produce resultados visualmente equivalentes en menor tiempo, cumpliendo así con los tiempos de ejecución interactivos exigidos por las aplicaciones de RA.

RECOMENDACIONES

Después de la investigación realizada y del desarrollo del sistema se recomienda:

- Continuar el estudio de las tendencias actuales en el campo de la Realidad Aumentada y la Visión por Computador.
- Mejorar la solución propuesta, la cual se basa en las características naturales de las imágenes, con el fin de mejorar su fiabilidad.
- Continuar con la optimización del sistema para lograr mejoras en los tiempos de procesamiento.
- Estudiar otras posibles características invariantes en las imágenes que, unidas a los puntos claves, proporcionen un resultado superior.
- Adicionar nuevos algoritmos de detección y extracción de características naturales en imágenes.
- Migrar la solución propuesta a la plataforma libre GNU \Linux.

REFENCIAS BIBLIOGRÁFICAS

1. X. BASOGAIN, M. OLABE, K. ESPINOSA, C. ROUÈCHE and J.C. OLABE. *Realidad Aumentada en la Educación: una tecnología emergente*. 2010.

2. MILEYDI MORENO MIRABAL and ERNESTO DE LA CRUZ GUEVARA RAMÍREZ. *Localización de objetos virtuales en el mundo real con técnicas de Realidad Aumentada.* [online]. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008. [Accessed 25 January 2014]. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1545_08/1/TD_1545_08.pdf

3. YADIRA VERDECIA ÁLVAREZ and LIANNET PEÑA GUERRA. *Herramienta de autor para la creación de contenido de realidad aumentada.* [online]. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2010. [Accessed 25 January 2014]. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03811_10/1/TD_03811_10.pdf

4. JOSÉ ANTONIO LEYVA REGALÓN. *Componente de Interacción Tangible para Realidad Aumentada* [online]. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2011. [Accessed 25 January 2014]. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04662_11/1/TD_04662_11.pdf

5. PAUL MILGRAM and FUMIO KISHINO. *A Taxonomy of Mixed Reality Visual Displays* [online]. December 1994. [Accessed 15 January 2014]. Available from: http://web.cs.wpi.edu/~gogo/hive/papers/Milgram_IEICE_1994.pdf

6. AZUMA, Ronald T. A survey of augmented reality. *Presence* [online]. 1997. [Accessed 2 March 2014]. Available from: http://www.mitpressjournals.org/userimages/ContentEditor/1332945956500/PRES_6-4_Azuma_web.pdf

3. GARCÍA, O. Boullosa. Estudio comparativo de descriptores visuales para la deteccion de escenas cuasi-duplicadas. *Febrero 2011*. [Accessed 2 March 2014]. Available from: http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20110318OscarBoullosa.pdf

4. BRADSKI, Gary R and KAEHLER, Adrian. *Learning OpenCV: computer vision with the OpenCV library*. Sebastopol, CA : O'Reilly, 2008. ISBN 9780596516130 0596516134.

9. RAÚL IGUAL and CARLOS MEDRANO. *Tutorial Opencv*. 23 April 2008.

10. JULIO MOLLEDA MERÉ. Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3d de productos laminados. [online]. July 2008. [Accessed 18 March 2014]. Available from: http://www.tdx.cat/bitstream/handle/10803/11139/UOV0060TJMM.pdf;jsessionid=6032828665ECEBD30351B7DD8D4B8050.tdx2?sequence=1

11. CARLOS RICOLFE VIALA, ANTONIO JOSÉ SÁNCHEZ SALMERÓN and RAÚL SIMARRO FERNÁNDEZ. TÉCNICAS DE CALIBRADO DE CÁMARAS. [online]. 2012.

[Accessed 18 March 2014]. Available from: http://www.ceautomatica.es/old/actividades/jornadas/XXIV/documentos/viar/135.pdf

12. ROGER Y. TSAI. *A versatile camera calibration technique for high accuracy 3D machine vision metrology using off the self TV cameras and lenses.* [online]. 1987. [Accessed 15 March 2014]. Available from: http://www4.cs.umanitoba.ca/~jacky/Robotics/Papers/tsai_calibration.pdf

13. ZHENGYOU ZHANG. *Flexible Camera Calibration by Viewing a Plane from Unknown Orientations* [online]. 1998. [Accessed 15 March 2014]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.7481&rep=rep1&type=pdf

14. DIEGO BORRO YAGÜEZ. ESTIMACIÓN DE LA POSE DE UNA CÁMARA PARA APLICACIONES DE VÍDEO AUMENTADO. [online]. 2010. [Accessed 15 March 2014]. Available from: http://jrsanchez.users.sourceforge.net/pdf/suficiencia.pdf

15. ELSAYED E. HEMAYED. *A survey of camera self-calibration* [online]. 2003. [Accessed 15 March 2014]. Available from: http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CBoQFjAA&url=http%3A%2F%2Fwww.egypteducation.edu.eg%2Fpluginfile.php%2F28441%2Fmod_folder%2Fcontent%2F1%2FSelfCalibration_Hemayed.pdf%3Fforcedownload%3D1&ei=c0ieU7X0LaHNsQT6rIHIBA&usg=AFQjCNEmPa4hAwyL-59H25pu_xV2reg1MA&bvm=bv.68911936,d.cWc&cad=rja

16. JAIRO ROBERTO SÁNCHEZ TAPIA and DIEGO BORRO YAGÜEZ. ESTIMACIÓN DE LA POSE DE UNA CÁMARA PARA APLICACIONES DE VÍDEO AUMENTADO. [online]. 2010. [Accessed 15 March 2014]. Available from: http://jrsanchez.users.sourceforge.net/pdf/suficiencia.pdf

17. DAVID G. LOWE. Distinctive Image Features from Scale-Invariant Keypoints. [online]. 5 January 2004. [Accessed 2 March 2014]. Available from: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

18. BAY, Herbert, TUYTELAARS, Tinne and VAN GOOL, Luc. Surf: Speeded up robust features. In : *Computer Vision–ECCV 2006* [online]. 2006. [Accessed 2 March 2014]. Available from: http://link.springer.com/chapter/10.1007/11744023_32

19. RUBLEE, Ethan, RABAUD, Vincent, KONOLIGE, Kurt and BRADSKI, Gary. ORB: an efficient alternative to SIFT or SURF. In : *Computer Vision (ICCV), 2011 IEEE International Conference on* [online]. IEEE, 2011. p. 2564–2571. [Accessed 16 March 2014]. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126544

20. HEINLY, Jared, DUNN, Enrique and FRAHM, Jan-Michael. Comparative evaluation of binary features. In : *Computer Vision–ECCV 2012* [online]. Springer, 2012. p. 759–773. [Accessed 30 March 2014]. Available from: http://link.springer.com/chapter/10.1007/978-3-642-33709-3_54

21. ALAHI, Alexandre, ORTIZ, Raphael and VANDERGHEYNST, Pierre. Freak: Fast retina keypoint. In : [online]. IEEE, 2012. p. 510–517. [Accessed 30 March 2014]. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6247715

22. CANÓS, José H., LETELIER, Patricio and PENADÉS, Mª Carmen. Metodologías Ágiles en el desarrollo de Software. *Universidad Politécnica de Valencia, Valencia* [online]. 2003. [Accessed 3 March 2014]. Available from: http://noqualityinside.com.ar/nqi/nqifiles/XP_Agil.pdf

23. CARLOS F. PONS CALVO. GRUPO VISIÓN POR COMPUTADOR. [online]. 23 September 2008. [Accessed 3 March 2014]. Available from: http://www.vision.uji.es/~montoliu/docs/pfc/CarlosPons.pdf

24. PAUL BUSTAMANTE, IKER AGUINAGA, MIGUEL AYBAR, LUIS OLAIZOLA and IÑIGO LAZACANO. *Aprenda C++ Básico como si estuviera en primero*. February 2004.

25. MÉNDEZ, Gonzalo. Ingeniería de Requisitos. [online]. 2008. [Accessed 6 April 2014]. Available from: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/03-Requisitos.pdf

26. LUIS CALABRIA and PABLO ORTIZ. *Metodología XP* [online]. 2003. [Accessed 6 April 2014]. Available from: http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologia_xp.pdf

27. CARLOS SÁNCHEZ GONZÁLEZ. *ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras.* [online]. Universidade da Coruña, 2004. [Accessed 6 April 2014]. Available from: http://oness.sourceforge.net/proyecto/html/ch05s02.html

28. CARLOS REYNOSO and NICOLÁS KICILLOF. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft* [online]. March 2004. [Accessed 9 May 2014]. Available from: http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF

29. ERICH GAMMA, RICHARD HELM, RALPH JOHNSON and JOHN VLISSEDES. *Patrones De Diseño*. 2003. ISBN 84-7829-059-1.

30. Patrones de Asignación de Responsabilidades - EcuRed. [online]. November 2011. [Accessed 21 April 2014]. Available from: http://www.ecured.cu/index.php/Patrones_de_Asignaci%C3%B3n_de_Responsabilidades

31. Extreme Programming - EcuRed. [online]. 2011. [Accessed 25 May 2014]. Available from: http://www.ecured.cu/index.php/Extreme_Programming#Producci.C3.B3n

32. KRYSTIAN MIKOLAJCZYK. [Accessed 17 May 2014]. Available from: http://lear.inrialpes.fr/people/mikolajczyk/Database

ANEXOS

ANEXO 1 Historias de Usuario

Historia de Usuario	
Número: 1	**Usuario:** Cliente
Nombre: Lectura de imagen	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1 semana	**Iteración asignada:** 1
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Permite realizar la carga de las imágenes patrón y de búsqueda, las cuales serán analizadas por el sistema.	
Observaciones: En caso de que el usuario no realice correctamente esta operación, el sistema debe ser capaz de mostrar un mensaje notificando dicho error.	

Tabla 25. Historia de Usuario 1

Historia de Usuario	
Número: 2	**Usuario:** Cliente
Nombre: Detección de *keypoints*	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1 semana	**Iteración asignada:** 1
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Dado el algoritmo de detección seleccionado por el usuario, el sistema debe ser capaz de realizar la detección de los correspondientes *keypoints* en las imágenes de entrada.	
Observaciones: En caso de que el usuario no realice correctamente la selección del algoritmo de detección, el sistema debe notificar mediante un mensaje la ocurrencia de dicho error.	

Tabla 26. Historia de Usuario 2

Historia de Usuario	
Número: 3	**Usuario:** Cliente
Nombre: Descripción de *keypoints*	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1 semana	**Iteración asignada:** 2
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Dado el algoritmo de descripción seleccionado por el usuario, el sistema debe ser capaz de realizar la descripción de los *keypoints* obtenidos en la fase anterior.	
Observaciones: En caso de que el usuario no realice correctamente la selección del algoritmo de extracción, el sistema debe notificar mediante un mensaje la ocurrencia de dicho error.	

Tabla 27. Historia de Usuario 3

Historia de Usuario	
Número: 4	**Usuario:** Cliente
Nombre: Cálculo de correspondencias	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1 semana	**Iteración asignada:** 2
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Una vez realizada la descripción de los *keypoints,* el sistema debe ser capaz de realizar el cálculo de correspondencias entre estas.	
Observaciones:	

Tabla 28. Historia de Usuario 4

Historia de Usuario	
Número: 5	**Usuario:** Cliente
Nombre: Refinamiento de correspondencias	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2 semanas	**Iteración asignada:** 3
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Una vez realizado el cálculo de las correspondencias, el sistema debe ser capaz de realizar un refinamiento del resultado obtenido en esta fase anterior.	
Observaciones:	

Tabla 29. Historia de Usuario 5

Historia de Usuario	
Número: 6	**Usuario:** Cliente
Nombre: Cálculo de la homografía	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2 semanas	**Iteración asignada:** 3
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: El sistema debe ser capaz de realizar el cómputo de la matriz de homografía entre la imagen de búsqueda y la imagen patrón.	
Observaciones: Con el correcto desarrollo de esta funcionalidad, se garantiza la una optimización en cuanto al cálculo de correspondencias. Puesto que solamente se tendrán en cuenta aquellas correspondencias que garanticen el correcto cálculo de la homografía. Si el resultado de tener en cuenta solamente correspondencias que cumplan con esta condición devuelve un número mínimo de estas, el sistema interpretará que no la imagen de búsqueda no guarda relación con la imagen patrón.	

Tabla 30. Historia de Usuario 6

Historia de Usuario	
Número: 7	**Usuario:** Cliente
Nombre: Estimación de la pose	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2 semanas	**Iteración asignada:** 4
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: El sistema debe ser capaz de realizar el cómputo de la pose respectiva a la imagen de búsqueda.	
Observaciones:	

Tabla 31. Historia de Usuario 7

Historia de Usuario	
Número: 8	**Usuario:** Cliente
Nombre: Dibujo de la escena 3D	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2 semanas	**Iteración asignada:** 4
Programador responsable: Joaquin Alberto Díaz Vázquez	
Descripción: Los cálculos obtenidos en las fases anteriores servirán para que el sistema sea capaz de mostrar una información aumentada.	
Observaciones:	

Tabla 32. Historia de Usuario 8

ANEXO 2 Tareas de Ingeniería

Tareas de Ingeniería	
No. de tarea: 2	**No. HU:** 1
Nombre de la tarea: Validación de los datos capturados.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 5 días
Fecha de inicio: 13/2/2014	**Fecha final:** 18/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación correspondiente para asegurar que los datos introducidos por el usuario son válidos.	

Tabla 33. Tarea de Ingeniería 2 de la HU 1

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 2
Nombre de la tarea: Desarrollo de una interfaz que muestre los algoritmos de detección de *keypoints*.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 día
Fecha de inicio: 18/2/2014	**Fecha final:** 19/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza el diseño e implementación de la interfaz que permita al usuario elegir el algoritmo de detección de características naturales que desee aplicarle a las imágenes a procesar.	

Tabla 34. Tarea de Ingeniería 1 de la HU 2

Tareas de Ingeniería	
No. de tarea: 2	**No. HU:** 2
Nombre de la tarea: Validación de la selección del algoritmo de detección de características naturales seleccionado por el usuario.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2 días
Fecha de inicio: 19/2/2014	**Fecha final:** 21/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación en la clase correspondiente para asegurar que el usuario ha elegido de forma correcta un algoritmo de detección de características naturales.	

Tabla 35 Tarea de Ingeniería 2 de la HU 2

Tareas de Ingeniería	
No. de tarea: 3	**No. HU:** 2
Nombre de la tarea: Implementación de la funcionalidad encargada de realizar el análisis de las características naturales de las imágenes.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 4 días
Fecha de inicio: 21/2/2014	**Fecha final:** 25/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de detectar las características naturales invariantes presentes en las imágenes dado el algoritmo de detección seleccionado por el usuario.	

Tabla 36. Tarea de Ingeniería 3 de la HU 2

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 3
Nombre de la tarea: Desarrollo de una interfaz que muestre los algoritmos de descripción de *keypoints*.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 día
Fecha de inicio: 26/2/2014	**Fecha final:** 27/2/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza el diseño e implementación de la interfaz que permita al usuario elegir el algoritmo de descripción de características naturales que desee aplicarle a las imágenes a procesar.	

Tabla 37. Tarea de Ingeniería 1 de la HU 3

Tareas de Ingeniería	
No. de tarea: 2	**No. HU:** 3
Nombre de la tarea: Validación de la selección del algoritmo de extracción de características naturales seleccionado por el usuario.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2 día
Fecha de inicio: 27/2/2014	**Fecha final:** 1/3/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza el diseño e implementación de la interfaz que permita al usuario elegir el algoritmo de descripción de características naturales que desee aplicarle a las imágenes a procesar.	

Tabla 38. Tarea de Ingeniería 2 de la HU 3

Tareas de Ingeniería	
No. de tarea: 3	**No. HU:** 3
Nombre de la tarea: Implementación de la funcionalidad encargada de realizar la descripción de las características naturales de las imágenes.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 4 día
Fecha de inicio: 1/3/2014	**Fecha final:** 5/3/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de realizar la descripción las características naturales invariantes presentes en las imágenes dado el algoritmo de extracción de características seleccionado por el usuario.	

Tabla 39. Tarea de Ingeniería 3 de la HU 3

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 4
Nombre de la tarea: Implementación de la funcionalidad encargada de realizar el cálculo de las correspondencias entre las imágenes de entrada.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 7 día
Fecha de inicio: 5/3/2014	**Fecha final:** 12/3/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de realizar el cálculo de las correspondencias existentes entre las características de la imagen de búsqueda y la imagen patrón.	

Tabla 40. Tarea de Ingeniería 1 de la HU 4

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 5
Nombre de la tarea: Realizar un refinamiento del cálculo de las correspondencias.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 14 día
Fecha de inicio: 13/3/2014	**Fecha final:** 27/3/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de realizar un refinamiento de las correspondencias existentes entre la imagen patrón y la imagen de búsqueda. Eliminando falsas correspondencias.	

Tabla 41. Tarea de Ingeniería 1 de la HU 5

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 6
Nombre de la tarea: Realizar el cálculo de la homografía entre la imagen patrón y la imagen de búsqueda.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 14 día
Fecha de inicio: 27/3/2014	**Fecha final:** 10/4/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de realizar el cálculo de la homografía entre la imagen patrón y la imagen de búsqueda. El resultado de esta funcionalidad debe poder utilizarse para mejorar el resultado del refinamiento de las correspondencias, obteniendo así un refinamiento de correspondencias por homografía.	

Tabla 42. Tarea de Ingeniería 1 de la HU 6

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 7
Nombre de la tarea: Realizar el cálculo de la pose de la cámara respecto a la imagen patrón.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 14 día
Fecha de inicio: 11/4/2014	**Fecha final:** 25/4/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de realizar el cálculo de la pose de la cámara respecto a la imagen patrón. Para lograr dicho resultado, se debe utilizar el resultado arrojado por la homografía.	

Tabla 43. Tarea de Ingeniería 1 de la HU 7

Tareas de Ingeniería	
No. de tarea: 1	**No. HU:** 8
Nombre de la tarea: Validar el resultado de los cálculos geométricos obtenidos con una escena de *RA*	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 14 día
Fecha de inicio: 25/4/2014	**Fecha final:** 9/5/2014
Programador responsable: Joaquin Alberto Díaz Vázquez.	
Descripción: Se realiza la implementación de la funcionalidad en la clase correspondiente encargada de crear un resultado de *RA*, basándose en los cálculos geométricos obtenidos en etapas anteriores, validando así la robustez del proceso de extracción de características naturales en imágenes.	

Tabla 44. Tarea de Ingeniería 1 de la HU 8

GLOSARIO DE TÉRMINOS

3D: En todo el contexto se utiliza para hacer referencia a algo que está o pueda ser representado en tres dimensiones.

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

Framework: En el desarrollo de software, un *framework* es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros *softwares* para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GPS: Un sistema GPS (*Global Positioning System*) o Sistema de Posicionamiento Global es un sistema compuesto por un lado por una red de 30 satélites denominada *NAVSTAR*, situados en una órbita a unos 20.000 km. de la Tierra, y por otro lado por unos receptores GPS, que permiten determinar nuestra posición en cualquier lugar del planeta, bajo cualquier condición meteorológica.

HMD: (*Head Mounted Display*) Dispositivo montado en la cabeza. Como su nombre lo indica este es un dispositivo muy parecido a un casco de realidad virtual, pero el usuario en vez de mirar un entorno totalmente generado por la computadora, observa una parte del mundo real con elementos virtuales insertados en la escena que está visualizando.

IDE: Entorno de desarrollo integrado. Herramienta que se usa para facilitar el desarrollo de software.

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

Tracking: Este término se utiliza para denominar el seguimiento de objetos a través de visión por computadoras.