Universidad de las Ciencias Informáticas

Facultad 10

Sistema de reconocimiento de objetos dentro de imágenes.

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

Autor:

Yosbel Santana Castillo

Tutor:

Ing. Siovel Rodríguez Morales

Ciudad de la Habana

Mayo de 2009

NO BASTA SABER, SE DEBE TAMBIÉN APLICAR. NO ES SUFICIENTE QUERER, SE DEBE TAMBIÉN HACER.

JOHANN WOLFGANG GOETHE

Declaración de Autoría:

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Yosbel Santana Castillo **Siovel Rodríguez Morales**

_____ _____

Firma del Autor Firma del Tutor

Opinión del Tutor del Trabajo de Diploma:

Título: Sistema de reconocimiento de objetos dentro de imágenes.

Autores: Yosbel Santana Castillo.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de:

Agradecimientos:

A mis abuelos, aunque algunos ya no están, sin ellos no pudiera estar hoy aquí,

A mis padres, por el magnífico ejemplo que siempre me han dado y por ayudarme a realizar mis sueños, por ser mi guía e inspiración,

A mi hermano, por representar tanto en mi vida y ser un gran amigo,

A mi novia, por su ayuda incondicional y por enseñarme a ver la vida desde un punto de vista diferente,

A Yeya, Manuel y mis primos, por estar siempre a mi lado,

A mi tutor Siovel, por la gran ayuda que me ha brindado en el desarrollo de este trabajo, que no es sólo mío, es de él también, además porque más que mi tutor es mi amigo, a quien admiro, respeto y siempre le estaré agradecido,

A mis amigos, los de aquí y los de Aguacate,

Al Comandante en Jefe, por la magnífica idea de crear esta universidad tan especial,

A todo el que me ayudó.

Dedicatoria:

A mis abuelos, este triunfo también es de ellos,

A mis padres y mi hermano por su apoyo incondicional en cada momento y por confiar siempre en mí,

A mi novia, por su ayuda y por ser fuente de inspiración.

Resumen:

En la Universidad de las Ciencias Informáticas se está implementando, desde el año 2005 y a petición de la Oficina de Seguridad para las Redes Informáticas, un sistema de filtrado que pretende, de forma flexible, regular los contenidos inadecuados que provienen de Internet. Este sistema de filtrado tiene como componente principal una Base de Datos de URLs Categorizadas, mediante la cual se toma la decisión de autorizar o denegar el acceso a los recursos solicitados. Dado que es inviable la categorización de las URLs de forma manual, se necesita para esto un proceso automático, por lo que surge el Motor de Clasificación Inteligente de Contenidos. Este motor es el encargado de clasificar de forma automática las páginas Web y, para poder dar un criterio general de las mismas, es necesario la categorización tanto de su texto como de sus imágenes. Las imágenes constituyen un componente esencial en las páginas Web. Estas son usadas fundamentalmente para hacer más atractivos los contenidos, sin embargo, también contribuyen a la existencia de información dañina como la pornografía, el racismo, la violencia y otros. En este trabajo se propone un módulo para el reconocimiento de objetos en imágenes, con vistas a su aplicación en el Motor de Clasificación Inteligente de Contenidos. A la solución elaborada se le realizaron una serie de pruebas, las cuales arrojaron muy buenos resultados, ya que la fiabilidad con que reconoce es elevada y el tiempo que demora el procesado es bueno, constituyendo así una acertada solución al problema existente.

Palabras claves: Reconocimiento de patrones, *keypoint,* vecino más cercano.

Índice de contenidos

Introducción

En la actualidad Internet constituye la mayor fuente de información a nivel mundial. Su contenido se considera equivalente a tres millones de veces la cantidad de libros escritos en la historia [1]. Su tamaño en bytes se encuentra en el orden de los hexabytes (2^{60} bytes) y tiene una proyección de crecimiento de hasta seis veces para el 2010. La cantidad de sitios Web superan ampliamente los 100 millones [2]. En Internet se alberga una enorme cantidad de contenidos científicos, educativos, artísticos, deportivos, culturales y muchos otros que han permitido un gran salto en el desarrollo de la humanidad. El conocimiento que antes parecía inaccesible para muchos, ahora se tiene a la distancia de un clic y esto es gracias a la red de redes.

Sin embargo, estas ventajas conllevan también al surgimiento de problemas. En Internet prácticamente cualquier persona puede publicar un sitio Web con un fin determinado, provocando la existencia de otra parte a la cual se le puede llamar "*la cara oscura de Internet"*, donde se pueden encontrar sitios con contenidos pornográficos (alrededor del 12% del total de sitios Web [3]), pedófilos, xenofóbicos, terroristas, extremistas (la cantidad de sitios extremistas e ilegales creció en el 2005 en un 42,4% aproximadamente [4]), de drogas (más de 1400 foros de opinión en España acerca de cómo cultivarla y elaborarla de forma casera [5]). Lo más alarmante resulta el crecimiento exagerado de estos contenidos, lo cual significa un gran peligro para la sociedad. Por otra parte, se estima que menos del 10% de la información dentro de la Web está clasificada o bajo un nivel de veracidad.

Un elemento esencial dentro de las páginas Web lo constituye, sin lugar a dudas, las imágenes. Éstas son utilizadas, entre otras cosas, para hacer más atractivos los contenidos o añadir elementos gráficos como flechas de navegación, logotipos de empresas, avatares de usuarios, entre otros. Hoy en día es casi inimaginable un sitio Web sin la presencia de imágenes. Sin embargo, estas contribuyen de forma decisiva a la existencia de contenidos dañinos en la red.

Producto a que Internet se ha convertido en unas de las principales fuentes de consulta de la sociedad ya sea para el estudio, trabajo o para el ocio, 'esta se encuentra expuesta a sufrir directamente daños físicos o psicológicos que la información nociva puede causarle y sobre todo al sector más vulnerable de la población, los menores.

Nuestro país ha tomado consciencia de este problema y es por ello que en la Universidad de las Ciencias Informáticas (UCI) se está desarrollando, desde el 2005 y a petición de la Oficina de Seguridad para las Redes Informáticas (OSRI), un proyecto llamado Filtrado de Paquetes por Contenido (FILPACON). Su funcionamiento general se muestra en el anexo 1, y con el mismo se pretende regular el flujo de materiales inadecuados en nuestra red. El componente fundamental de FILPACON es una Base de Datos de URLs Categorizadas (BDUC) y, dado el gran volumen que posee Internet, llenar ésta de forma manual es una tarea inviable. Para mantener actualizada dicha base de datos se usará el Motor de Clasificación Inteligente de Contenidos (MOCIC), el cual se encuentra en desarrollo y cuya arquitectura se muestra en el anexo 2. Uno de los módulos identificados en este motor es el Módulo de Reconocimientos de Objetos en Imágenes, que es fundamental para la categorización general del documento HTML.

Dada la necesaria presencia de este módulo, surge entonces el siguiente **problema científico:** ¿Cómo reconocer objetos con diferentes formatos y tamaños, de forma tal que aporte un identificador eficaz al motor de categorización de sitios Web de un sistema de filtrado por contenidos de Internet?

La clasificación automática se refiere a un proceso que se ejecuta de forma independiente y para esto es necesario el uso de técnicas de Inteligencia Artificial (IA), por lo que se enmarcó el **objeto de estudio** de este trabajo en el reconocimiento de patrones de las imágenes y como **campo de acción** el reconocimiento de estos patrones para detectar objetos en las imágenes, dado que es una línea de investigación bien definida dentro del objeto de estudio mencionado.

Por lo expuesto anteriormente se **defiende la siguiente idea**: Si se implementa un reconocedor automático de objetos en imágenes MOCIC aumentará su efectividad, siendo así la categorización de los documentos HTML mucho más confiable.

El presente trabajo se plantea como **objetivo general** desarrollar un módulo que permita reconocer objetos de forma automatizada, con la facilidad de ser integrado al motor de categorización de un sistema de filtrado por contenidos. El antes mencionado objetivo general se desglosa en los siguientes **objetivos específicos**:

- Obtener los descriptores más relevantes de las imágenes para el reconocimiento de objetos.

- Elegir una técnica eficiente para el reconocimiento de los objetos.
- Implementar un módulo que sea capaz de reconocer objetos dentro de imágenes.
- Incorporar el módulo al motor de clasificación del sistema de filtrado FILPACON.

Para cumplir con los objetivos planteados se han definido las siguientes **tareas de investigación**:

- Investigar sobre el desarrollo actual de los sistemas de filtrado de contenido de Internet, así como de los motores de clasificación de algunos de estos sistema de filtrado, fundamentalmente los libres y de técnicas de procesamiento de imágenes y reconocimiento visual de patrones.
- Analizar y determinar las características fundamentales a extraer de una imagen para realizar el reconocimiento de objetos.
- Estudiar las diferentes técnicas de reconocimiento de objetos existentes y determinar cuáles serán factibles utilizar.
- Estudiar herramientas libres para acelerar el desarrollo del presente trabajo.
- Estudiar los posibles patrones de diseño a aplicar en la solución.
- Realizar pruebas al sistema para calcular su fiabilidad.
- Realizar pruebas de integración entre el módulo desarrollado y MOCIC.

Métodos científicos

- **Histórico-lógico**: Se utilizó para el estudio del desarrollo de los sistemas de filtrado y los acontecimientos ocurridos en el transcurso de la historia en relación a estos, así como los principios generales de su funcionamiento y desarrollo. La investigación estará basada en los datos históricos fundamentales encontrados y en las necesidades actuales, garantizando que el sistema a desarrollar no constituya un simple razonamiento especulativo.

- **Teórico**: Revisión bibliográfica profunda acerca de los sistemas de filtrado y sobre los fundamentos teóricos del procesamiento de imágenes.
- **Observación**: Se utilizó para la selección de los descriptores de los objetos mediante la observación de las imágenes segmentadas.
- **Modelación**: Se utilizó mediante UML para reflejar la estructura, relaciones internas y características de la solución a través de diagramas.
- **Empírico**: Se evidenció en la puesta a punto del sistema para observar y medir su comportamiento. Se probó el módulo con varios ejemplos con lo que se pudo medir su fiabilidad.

Resultados esperados

- Que el módulo sea capaz de automatizar el proceso de reconocimiento de objetos en imágenes con una fiabilidad de al menos un 80%, y que el mismo pueda ser incorporado a sistemas de filtrado de contenidos.

Estructura del contenido

El presente documento consta de tres capítulos, además de la introducción, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos, anexos y siglario. A continuación se presenta un resumen del contenido de cada capítulo.

Capítulo1. Fundamentación Teórica: En este capítulo se muestra el estado actual del tema relacionado con el campo de acción. En él se analiza la existencia de soluciones a escala nacional e internacional que puedan reutilizarse y se presenta además una breve descripción del objeto de estudio, así como las herramientas fundamentales utilizadas para el desarrollo del software.

Capítulo2. Descripción del sistema: En este capítulo se da una explicación de la teoría y las técnicas utilizadas para la realización del sistema, así como los documentos generados por la metodología empleada.

Capítulo 3. Validación de la solución propuesta: En este capítulo se describen las pruebas realizadas al sistema y los principales resultados obtenidos.

Capitulo1. Fundamentación Teórica

1.1 Introducción

Los sistemas que reconocen objetos en imágenes son de mucha utilidad y, por ejemplo, en la medicina se usan para determinar en una toma de una región del cuerpo una posible afectación o anomalía. Otros usos importantes, y a los cuales va enfocado este trabajo, lo constituyen los motores de clasificación automática de contenidos, los cuales representan una parte muy importante de los sistemas de filtrados por contenidos y en los cuales se encuentran inmersas las aplicaciones capaces de reconocer objetos, aportando así un criterio muy importante para que los sitios Web se puedan clasificar satisfactoriamente.

1.2 Actualización

Un filtro de contenido es uno o más elementos de software que operan juntos para regular (permitir o denegar) el acceso de los usuarios a determinados contenidos que se encuentran en Internet. FILPACON constituye una solución de este tipo y, ante la necesidad de un reconocedor de objetos para el mismo, se precisa el estudio de soluciones similares que puedan existir.

1.2.1 Ámbito internacional

En el marco particular de los sistemas de filtrado, el uso de categorizadores de imágenes para llenar sus bases de datos no está del todo extendido, de hecho existen sistemas de filtrado reconocidos a escala mundial que no analizan las imágenes para dar una conclusión de una página Web analizada [6].

Otros productos sí hacen un análisis tanto del texto como de las imágenes de una página Web para dar un criterio de éstas. Estos categorizadores de textos e imágenes de los productos que lo poseen, han influido en gran medida a la aceptación de los mismos, ya que al utilizar técnicas de IA, se reduce la dependencia de las listas de URLs logrando que el filtro en cuestión se adapte a la naturaleza dinámica de Internet. A continuación se muestra una pequeña lista de algunos sistemas de filtrado que poseen un categorizador de imágenes:

- POESIA[1]
- OPTENET[2]
- PROVENTIA WEB FILTER[3]
- Netsweeper[4]

Con la excepción de POESIA, los filtros mencionados y por ende sus categorizadores, son software privativos teniendo éstos como desventajas las siguientes:

- No son productos separados del filtro.
- Son diseñados como una solución a la medida del propio filtro, imposibilitando el uso de éstos en otro entorno.
- Al no tener acceso al código fuente, es imposible su mantenimiento.
- Alta dependencia de los propietarios.

Image-filterTM es un categorizador de imágenes con una efectividad del 96 % y puede analizar de forma automática el contenido de imágenes y vídeos. Está diseñado para la integración con otros productos, con el objetivo de lograr una solución de filtrado para negocios o para Proveedores de Servicios de Internet (ISPs). Este categorizador, a pesar de tener relevantes ventajas al igual que los mencionados, es un software privativo.

POESIA es un proyecto diseñado para ser aplicado en el sistema operativo GNU/Linux bajo la licencia libre GPL. Dicho proyecto ha supuesto la creación de un filtro de contenidos pornográficos, violentos y xenófobos basado en el filtrado tanto de contenidos textuales como de las imágenes. A pesar de las ventajas mencionadas, este filtro presenta también sus desventajas:

1 http://www.poesia-filter.org
2 http://www.optenet.com/es/webfilter.asp?c=1
3 http://www.virtual.com/pages/partners_iss.aspx
4 http://www.netsweeper.com/Developers/Categorization

- En cuanto al filtrado de imágenes presenta una eficiencia aproximada de un (43 %) [7], la cual es considerada baja.

- Su última versión disponible data del año 2006[5] lo que supone un bajo mantenimiento del software.

1.2.2 Ámbito nacional

En el país FILPACON constituye, hasta el momento, la única solución de su tipo implicando novedad en cada uno de sus componentes. No se tienen referencias de un reconocedor automático de objetos desarrollado por ninguna institución del país, por lo antes expresado, se llega a la conclusión de que a escala nacional no existe una solución a nuestro problema.

1.3 Reconocimiento de patrones

Al estar enmarcado el problema existente en la Inteligencia Artificial (IA), se hace necesario un estudio sobre el tema. Hoy día la IA constituye una solución elegante para diversos problemas de la vida real con apreciables ventajas. Automatizar una tarea usando técnicas de IA hace que aumente la eficiencia de ésta, ya que se realiza las 24 horas del día y se eliminan errores que los humanos cometen por agotamiento físico. En el proceso automatizado con un ordenador de nuestros tiempos, se pueden automatizar tareas complejas dado el avance del hardware y software. En la mayoría de las ocasiones, un ordenador es más eficiente a la hora de realizar la tarea específica para la cual fue concebido, que un humano.

El reconocimiento de patrones es el campo dentro de la IA más cercano al problema presentado, por lo que el presente trabajo enmarca su estudio en dicho campo. Siguiendo la definición de Watanabe[8], un patrón es una entidad a la que se le puede dar un nombre y que está representada por un conjunto de propiedades medidas y las relaciones entre ellas, esto es llamado vector de características. Por ejemplo, un patrón podría ser una imagen de una figura X, y su vector de características serían valores numéricos relacionados con esa figura X como su forma, tamaño y posición. Un sistema de reconocimiento de

5 http://sourceforge.net/projects/poesia

patrones tiene uno de los siguientes objetivos:

- Identificar el patrón como miembro de una clase ya definida[6]
- Asignar el patrón a una clase todavía no definida[7]

El reconocimiento de patrones es ya una ciencia consolidada y cuenta con un flujo bien definido para su uso. El diseño de un sistema de reconocimiento de patrones se lleva a cabo normalmente en cuatro etapas como se ilustra en la figura 1.1

Figura 1.1: Flujo para el reconocimiento de patrones.

Etapa 1. Adquisición: Carga los datos que se analizarán en una variable. En ocasiones estos datos se encuentran de forma analógica y es una responsabilidad de este paso llevarlo a digital.

Etapa 2. Extracción de características: Se extraen los descriptores más relevantes del objeto analizado. Estos descriptores son seleccionados empíricamente o mediante la utilización de heurísticas.

Etapa 3. Toma de decisiones o Agrupamiento: Se entrena, de forma manual[8] o automática[9], dado un conjunto de vectores provenientes de la etapa 2. Este paso se ejecuta solo en la primera iteración; una vez que se ejecute éste, el software quedará listo para la categorización.

Etapa 4. Categorización: Se le asigna una clase al vector resultante de la etapa 2. Este paso se ejecuta siempre a partir de la segunda iteración.

---

6 Clasificación supervisada.
7 Clasificación no supervisada, agrupamiento o clustering.
8 En este caso se necesita de una colección de entrenamiento, es decir, una muestra representativa previamente clasificada por clases.
9 Se refiere a la agrupación no supervisada.

Como se puede percibir, las etapas 3 y 4 se ejecutan en diferentes iteraciones. Para entrenar un software de reconocimiento de patrones se ejecutan las etapas 1, 2 y 3. Una vez entrenado el sistema, éste se puede categorizar ejecutando las etapas 1, 2 y 4.

1.3.1 Reconocimiento de patrones en imágenes

El procesamiento de imágenes es un campo del reconocimiento de patrones, por lo que éste tiene objetivos y características comunes. El flujo definido para el procesamiento de imágenes es similar al del reconocimiento de patrones, con la excepción de que el primero incluye un nuevo paso, la segmentación, la cual se inserta en el segundo paso; quedando ahora el proceso de categorización como se muestra en la figura 1.2.

Figura 1.2: Flujo para el reconocimiento de imágenes.

La segmentación de la imagen es el proceso que subdivide una imagen en regiones homogéneas. Cada región homogénea constituye una parte o un objeto en la escena. En otras palabras, la segmentación de la imagen es definida como la división de regiones que están conectadas entonces, cada píxel en la imagen, adquiere una única etiqueta que indica a la región que pertenece. La segmentación es uno de los más importantes elementos en el reconocimiento de imágenes, principalmente porque en este paso los objetos de interés son extraídos de la imagen [9, p. 131]. Estos objetos conforman los descriptores que formarán el vector de características.

1.3.2 Aplicación del procesamiento de imágenes

El procesamiento de imágenes tiene un gran número de aplicaciones en muchas áreas de la actividad humana, a continuación se mencionan algunas de ellas [10].

Seguridad: Mediante el reconocimiento de patrones biométricos como el rostro, las huellas dactilares, el iris del ojo, entre otros.

Medicina: Mediante el análisis de imágenes, las cuales pueden ser de resonancia magnética, para el diagnóstico de enfermedades.

Industria: Mediante la detección de fallos a través de una imagen de componentes industriales, como por ejemplo, detectar problemas en los filamentos de lámparas incandescentes.

Filtrado: Mediante la categorización de imágenes que se desean filtrar, como por ejemplo, las imágenes pornográficas, símbolos, rostros, objetos entro otros.

1.4 Metodología de desarrollo

Todo desarrollo de software es riesgoso y difícil de controlar y, si no se emplea una metodología que guíe este proceso, los resultados obtenidos son clientes insatisfechos y desarrolladores aún más descontentos. Actualmente a nivel internacional, en dependencia del tiempo de vida y la complejidad del proyecto que se vaya a desarrollar, se proponen diferentes metodologías. 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, es lógico pensar que para lograr ese propósito se necesite tomar en cuenta una de las muchas metodologías para el proceso de desarrollo que existen. Las metodologías, dadas sus características, se enmarcan en dos grandes grupos: los llamados "métodos pesados" y los "métodos 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.

En el presente trabajo se investigó sobre dos metodologías, una de tipo pesada y otra de tipo ágil, la metodología pesada o tradicional que se estudió fue Rational Unified Process (RUP) y la ágil fue Xtreme Programming(XP).

1.4.1 Rational Unified Process (RUP)

Es una de las metodologías más usadas y mayormente probadas a nivel internacional para el desarrollo de software.

Características de RUP [11]

Dirigido por Casos de Uso: Tiene a los Casos de uso como el hilo conductor que orienta las actividades de desarrollo. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él.

Centrado en la arquitectura: Propone la arquitectura de forma similar a la de un edificio. Es necesario tener varios planos con diferentes aspectos para tener una imagen completa del edificio antes que comience su construcción. Aquí entra en juego el término Arquitectura de Software, que abarca las diferentes vistas que se mencionaron anteriormente.

Iterativo e incremental: Propone la descomposición de proyectos grandes en proyectos más pequeños o subproyectos, cada uno de los cuales es una iteración, y cada iteración debe estar controlada y tratar un determinado grupo de casos de uso.

RUP provee un acercamiento a disciplinas para asignar tareas y responsabilidades en un desarrollo organizado. Su objetivo es asegurar la producción de software de alta calidad que conozca la necesidad de sus clientes dentro de un predecible espacio de tiempo y presupuesto. La figura 1.3 ilustra la arquitectura de RUP, la cual tiene dos dimensiones:

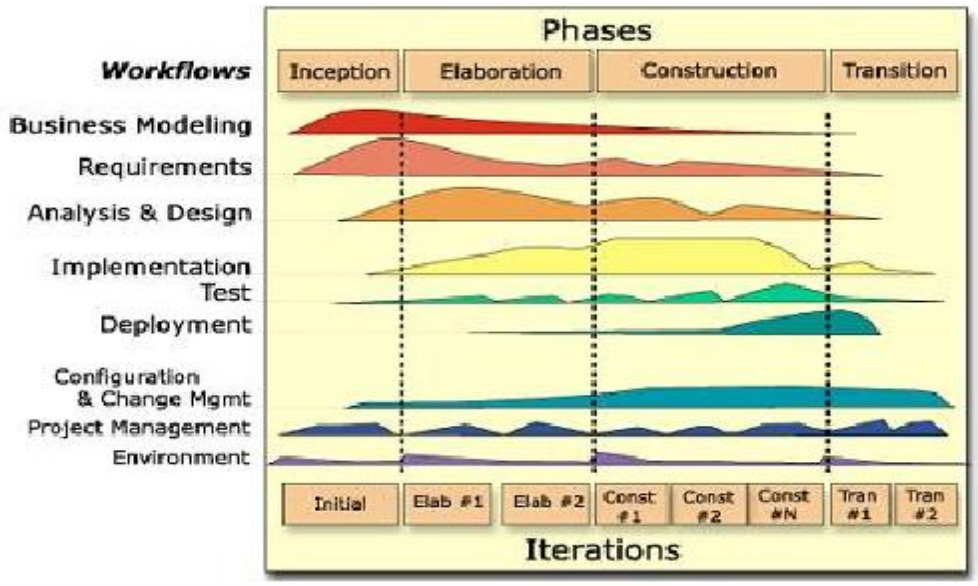


Figura 1.3: Arquitectura de RUP

Fases de desarrollo

Inicio:

1. **Determinar la visión del proyecto.**
2. **Elaboración: Determinar la arquitectura óptima.**
3. **Construcción: Llegar a obtener la capacidad operacional inicial.**
4. **Transmisión: Llegar a obtener el una primera versión del proyecto (*release*).**

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la

evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

Disciplina de Desarrollo

Ingeniería de Negocios: Entendiendo las necesidades del negocio.

Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.

Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.

Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado está presente.

Disciplina de Admitidas

Configuración y administración del cambio: Guardando todas las versiones del proyecto.

Administrando el proyecto: Administrando horarios y recursos.

Ambiente: Administrando el ambiente de desarrollo.

Distribución: Hacer todo lo necesario para la salida del proyecto.

Elementos del RUP

Actividades: Procesos que se llegan a determinar en cada iteración.

Trabajadores: Las personas o entes involucrados en cada proceso.

Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

1.4.2 Xtreme Programing(XP)

XP es una metodología ágil basada en cuatro principios: simplicidad, comunicación, retroalimentación y valor. Además está orientada por pruebas y refactorización, se diseñan e implementan las pruebas antes de programar la funcionalidad y el programador crea sus propios tests (pruebas) de unidad. [9]

Esta metodología se apoya en el trabajo orientado directamente al objetivo, basándose en las relaciones interpersonales, en la velocidad de reacción para la implementación y los cambios que puedan surgir durante el desarrollo del proceso. Por ello se requiere mantener dentro del equipo a un representante "competente" del cliente, quién responderá a todas las preguntas y dudas que surjan del equipo de desarrollo durante el proceso, de forma que no se retrase la toma de decisiones [12]. XP se basa en *UseStories* (historias de usuario), las cuales son escritas por el cliente o su representante dentro del equipo y describen los escenarios claves del funcionamiento del software. A partir de estas historias se generan los *releases* (entregas) entre el equipo y el cliente. Los *releases* son los que permiten definir las iteraciones necesarias para cumplir con los objetivos, de manera que cada resultado de la iteración sea un programa aprobado por el cliente de quien depende la definición de las siguientes iteraciones. XP genera solo cuatro artefactos, de los cuales solo tres son documentados; el último de ellos, las tarjetas CRC son pequeñas tarjetas de cartón que se mantienen en poder del equipo de desarrollo mientras dura el proceso de construcción. [12] Una característica importante de XP es la producción siempre en parejas del código, las cuales van cambiando constantemente para lograr así que todo el equipo sepa y pueda modificar el código generado según las necesidades. Esto logra que los integrantes del equipo de desarrollo aprendan entre sí y compartan dicho código totalmente. En la actualidad XP se proyecta a ser un modelo de desarrollo común, sencillo y adaptable a las características cambiantes y exigentes de empresas y clientes. [12]

Debido fundamentalmente al tiempo que se dispone para la realización del presente software y a la cantidad de desarrolladores del equipo de trabajo, además buscando mantener un proceso homogéneo entre todos los módulos del proyecto (Ver Anexo No 2, Arquitectura de MOCIC), se tomó la decisión de utilizar *Xtreme Programing*(XP) como metodología de desarrollo.

Es válido aclarar que las metodologías no son “camisas de fuerza” para los procesos de desarrollo de software sino guías para éste, por lo que pueden ser adaptadas a las necesidades del equipo de desarrollo. XP solamente genera la documentación básica y absolutamente indispensable, por lo que se decidió además adoptar algunas características de otras metodologías de forma independiente para apoyar la descripción del sistema, como es el diagrama de componentes que se presentan en el capítulo dos perteneciente a RUP.

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 le tiempo de desarrollo.

1.5.1 Lenguaje de programación

Los lenguajes son sistemas de comunicación. Un lenguaje de programación consiste en todos los símbolos, caracteres y reglas de uso que permiten a las personas "comunicarse" con las computadoras. Existen por lo menos varios cientos de lenguajes y dialectos de programación diferentes, cada uno de ellos fueron creado para un fin determinado, por lo que algunos son mejores que otros a la hora de realizar una tarea específica.

En la selección del lenguaje de programación se deben tener en cuenta muchos aspectos, para saber a priori si el seleccionado es el adecuado para solucionar el problema dado. Una de las posibles causas del fracaso de un proyecto es una mala selección del lenguaje de programación que se usará. Aspectos muy

importantes a tener en cuenta son:

- Expresividad
- Definición
- Tipos y estructuras de datos
- Facilidades de entrada y salida
- Eficiencia
- Soporte de librerías
- Conocimiento del equipo de desarrollo

Se realizó un estudio sobre tres lenguajes de programación muy usados a nivel mundial, y de esta forma escoger el más adecuado, estos fueron Java, C y C++

Breve historia de Java

Java fue originalmente denominado "Oak". Sus inicios datan de 1991 cuando James Gosling (en Sun Microsystems) encabezó un proyecto cuyo objetivo original era implementar una máquina virtual ampliamente portable y un lenguaje de programación, ambos orientados a dispositivos "embedded" (procesadores incorporados en diversos dispositivos de consumo masivo como VCR's, tostadoras, PDA's, teléfonos móviles, etc.) La sintaxis del lenguaje heredó características de C y C++, explícitamente eliminando aquellas que para muchos programadores (según los diseñadores) resultan excesivamente complejas e inseguras.

Con el auge de Internet, pareció natural aprovechar este lenguaje para desarrollar aplicaciones distribuidas y portables. La primera implementación de Java data de 1995 y pronto los "navegadores web" incorporaron soporte Java para la ejecución de pequeñas aplicaciones interactivas (Applets.) En la actualidad su uso es promovido para el desarrollo de aplicaciones empresariales del lado del servidor,

especialmente a través del estándar J2EE, así como en dispositivos móviles (a través del estándar J2ME.)

Breve historia de C

El lenguaje C es creado entre los años 1972-1973 por un equipo de programadores de los antiguos Laboratorios Bell de AT&T.

Dennis Ritchie diseñó e implementó el primer compilador de lenguaje C en un (¿prehistórico?) computador PDP-11. El lenguaje C se basó en dos lenguajes (prácticamente desaparecidos): "BCPL", escrito por Martin Richards, y "B", escrito por Ken Thompson en 1970 para el primer sistema UNIX en un PDP-7.

El lenguaje C originalmente "oficial" fue el "K&R C". Ese nombre proviene de los nombres de los autores del libro "The C Programming Language" (primera edición), a saber, Brian Kernigham y Dennis Ritchie, el cual fue durante muchos años la "referencia oficial" del lenguaje. Hacia 1988-1989 (luego de varios años de propuestas y acuerdos preliminares), el American National Standards Institute" (ANSI) adoptó una versión mejorada de C, conocida hasta hoy como "ANSI C" o C89. Esta es la versión descrita en la segunda edición de "The C Programming Language -- ANSI C". La versión ANSI C contiene una importante revisión a la sintaxis, especialmente para el uso de funciones (que fue tomada del lenguaje C++), así como la estandarización (parcial) de las librerías del sistema.

Breve historia de C++

El lenguaje C++ está basado en el lenguaje de programación C, el cual a su vez está basado en dos lenguajes muy primitivos (BCPL y B). Sus fundadores fueron Martin Richards (1976) y Ken Thompson (1970). Dos años más tarde de la creación del lenguaje B Dennis Ritchie implementó el diseño de BCPL, B y creó C; el cual se dió a conocer por ser el lenguaje de programación de desarrollo de UNIX.

A principios de la década de los 80, Bjarne Stroustrup[10] empezó a desarrollar C++, que recibiría formalmente su nombre a finales de 1983. En octubre de 1985, apareció la primera divulgación comercial del lenguaje y la primera edición del libro The C++ Programming Language[11].

A continuación se muestra en la tabla 1.1 una comparación teniendo en cuanta las características que nos interesan de los tres lenguajes.

Tabla 1.1 Comparación entre lenguajes de programación

Características	C	C++	Java
Expresividad	Regular	Muy buena	Muy buena
Definición	Regular	Muy buena	Muy buena
Tipos y estructuras de datos	Deficiente	Muy buena	Muy buena
Facilidades de entrada y salida	Buena	Buena	Buena
Eficiencia	Eficiente	Excelente	Buena
Soporte de librerías	Bueno	Muy buena	Muy buena
Conocimiento de los desarrolladores	Regular	Muy bueno	Bueno

10 http://www.research.att.com/~bs
11 Escrito por el propio Bjarne Stroustrup.

Después del estudio anterior y sumándole a esto que la aplicación debe ejecutarse en dos modos, *online y offline*, otro aspecto sumamente importante a tener en cuenta es la rapidez, pues el modo online requiere un procesamiento rápido y de esta forma dar una respuesta en corto tiempo a la petición del usuario. En este aspecto C y C++ son superiores, pero al presentar C++ un mejor soporte de librerías para el procesamiento de imágenes, se decidió profundizar más en sus características:

Alto nivel: Este tipo de lenguaje permite al programador abstraerse de la arquitectura y funcionamiento del hardware.

Orientado a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización del código de una manera más lógica y productiva.

Portabilidad: Un código escrito en C++ puede ser compilado en muchos sistemas operativos[12].

Brevedad: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las “palabras clave”.

Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++ con código producido en otros lenguajes de programación como Ensamblador o el propio C.

Velocidad: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel y a la reducida medida del lenguaje.

Por todo lo antes expuesto, se elige C++ como lenguaje de programación para desarrollar nuestra aplicación.

---

12 Para esto es necesario usar el estándar ANSI para C++. En ocasiones deben hacerse pequeños cambios.

1.5.2 Entorno Integrado de Desarrollo (IDE)

Las ventajas principales de un IDE radican en la automatización de algunas tareas como la gestión de directorios y la compilación asociada al desarrollo de aplicaciones y en poseer algunas funcionalidades que facilitan el desarrollo de software, disminuyendo así el tiempo de construcción. La selección de un IDE para el desarrollo de aplicaciones en C++ utililizando el sistema operativo GNU/Linux, se hace hasta cierto punto compleja, ya que muy pocos de los IDEs existentes completan código, lo cual es una de las características que más agiliza el desarrollo de software. Finalmente se decidió investigar sobre dos que poseen esta propiedad.

NetBeans

NetBeans es un poderoso entorno integrado de desarrollo de fuente abierta, disponible en GNU/Linux y otras plataformas. Este provee a los desarrolladores de software de todo lo necesario para crear aplicaciones de escritorio, empresariales, web y móviles, de nivel profesional y de plataforma cruzada, en los lenguajes Java y C++.

Entre las características principales de NetBeans se encuentran:

- Swing GUI Builder (formerly Project Matisse), para desarrollar rápidamente interfaces gráficas de usuario con soporte de localización y accesibilidad.
- Soporte integrado del control de versiones.
- Soporte para desarrollo colaborativo.
- Editor de código fuente avanzado.
- Soporte del desarrollo de aplicaciones móviles con Java ME.
- Soporte de desarrollo web visual.
- Profiler integrado para optimizar las aplicaciones.

Eclipse

Eclipse es un IDE conocido para el desarrollo de aplicaciones Java. Sin embargo, es un IDE mucho más flexible de lo imaginado, pues gracias a una infinidad de plugins éste permite, entra otras cosas, editar clases visuales de Java, programar aplicaciones J2EE, C/C++ y, en varios lenguajes más, conectarse a bases de datos y escribir consultas SQL, etc.

Características de Eclipse, como IDE de desarrollo:

- Posee una excelente integración con herramientas para el control de versiones (SVN)[13].
- Posee un potente administrador de proyecto.
- Es multi-plataforma.
- Altamente configurable.

Para el desarrollo en C/C++ con Eclipse se utiliza un plugin llamado CDT, el cual tiene las siguientes características:

- Editor de C/C++ (Funcionalidades básicas, resaltamiento de sintaxis, completamiento de código, etc).
- Depurador C/C++ (Usando GDB)
- Lanzador C/C++ (Lanzadores y aplicaciones externas)
- Programa analizador sintáctico
- Herramienta de búsqueda
- Proveedor de asistencia de contenido

13 Control de versiones SubVertion.

- Generador de un fichero llamado *Makefile*[14].

- El Eclipse consume menos recursos que el NetBeans cuando está en ejecución, además el proveedor de asistencia del primero es superior y es más configurable.

Después del estudio anterior se decidió utilizar Eclipse con el plugin CDT, el cual brinda una solución eficiente para el problema planteado. Para su uso se recomienda un ordenador con al menos 512 MB de RAM con un procesador a una velocidad de 2.4 GHz o superior.

1.5.3 Herramienta CASE

Actualmente, desde los inicios de la concepción de un software, se piensa en utilizar alguna herramienta CASE, debido a que la creación de los diagramas de su ciclo de vida a lápiz y papel es un proceso muy engorroso, casi imposible. En el ámbito informático existen muchas herramientas creadas para este fin. Para la realización del presente trabajo se indagó sobre dos de las más reconocidas, éstas son: Visual Paradigm y BoUML.

Visual Paradigm

- Herramienta fácil de utilizar que apoya las últimas notaciones de UML.

- Es software libre, licenciado bajo la GPL.

- Buena integración con Eclipse.

- Existen varias ediciones con prestaciones diferentes.

- Ingeniería directa como inversa.

14 Un fichero de configuración usado por el programa Make definiendo la ubicación del código fuente, como deberá ser compilado y vinculado para crear un programa ejecutable.

- Importa a Rational Rose[15].
- Exportación/importación XML
- Generador de impresos
- Soportada en varios lenguajes de programación.

BoUML

- Es software libre, licenciado bajo la GPL.
- Genera el diagrama de clases de diseño a partir de un código fuente (inversión de código).
- Genera el código fuente a partir del diagrama de clases del diseño (generación de código).
- Genera la documentación del proyecto (en HTML).
- Importa proyectos del Rational Rose.
- Posibilita la realización de los siguientes diagramas:

• Caso de uso • Secuencia • Objeto • Clase • Estado

• Componente • Actividades • Despliegue • Colaboración

Después del estudio realizado, se concluye que la herramienta CASE BoUML sería muy factible utilizarla para un equipo que utilice como metodología de desarrollo RUP, ya que brinda muchas facilidades para generar los diagramas y artefactos generados por ésta. Como la metodología seleccionada para desarrollar este trabajo es XP, entonces se decidió utilizar Visual Paradigm el cual, por sus características, brinda todas las herramientas necesarias para resolver el problema existente. Ésto, unido a la experiencia de los desarrolladores en su utilización, debe conducir al éxito del presente trabajo.

15 Es una de las más poderosas herramientas de modelado visual.

1.5.4 Bibliotecas disponibles

Dado que los distintos productos van ganando en complejidad cada día, se hace necesaria la búsqueda y utilización de herramientas que agilicen el desarrollo de software, como son las bibliotecas, paquetes de clases, frameworks, módulos, etc. C++ es un lenguaje con una gran historia, por lo que cuenta con muchas bibliotecas para la solución de disímiles problemas. Para poder resolver el problema planteado en este trabajo, se necesita una biblioteca capaz de manejar imágenes eficientemente. Mediante el estudio realizado se descubrió la existencia de tres con muy buenas características: LTI-Lib, VXL y OpenCV Library.

A continuación se muestra en la tabla 1.2 una comparación entre dichas librerías.

Tabla 1.2 Comparación entre librerías.

Aspecto	LTI-Lib	VXL	OpenCV Library
Licencia	LGPL	BSD-Compatible	BSD-Compatible
Lenguaje	C++	C++	C/C++
Operación con matrices	+	+	+
Procesamiento básico de imágenes	+	+	+
Procesamiento y análisis avanzado de imágenes	+	+	+/-
Visión 3D	-(solo estimación FM)	+	+/- mayor en las 1.x
Estadísticas de aprendizaje	+	+/-	CVS, en la 1.0
Características únicas	Muy genérico y arquitectura flexible	Gran biblioteca, tiene una larga historia	Incluye detección de rostros, Plataforma específica a través de la optimización de Intel IPP

“+” significa que el tipo de funcionalidad se implementa en más o menos completa.

“-” significa que la funcionalidad no es aplicable o está en estado rudimentario.

“+/-” significa que ésta implementada la mitad de la funcionalidad.

Según la comparación anterior se decide utilizar la librería OpenCV. Sus características son suficientes para resolver el problema presentado en este trabajo, además es un aspecto de mucho peso para su

elección el problema de su licencia que es BSD, por lo que se decide utilizar la misma como base para desarrollar un reconocedor de objetos en imágenes.

1.5.5 Otras herramientas

Se deben mencionar algunas herramientas auxiliares utilizadas para la confección del presente documento que, aunque no formaron parte del desarrollo del software, sí contribuyeron a disminuir el tiempo de desarrollo y aumentar la calidad del trabajo, ellas son:

Gimp: Es un software de edición de imágenes. Es usado en el desarrollo del documento para editar las imágenes generadas y mejorar su estética. Usado en el desarrollo del software para observar el histograma de una imagen, selección de umbral, cortar una región de una imagen, entre otras tareas.

1.6 Objetos a reconocer

Uno de los aspectos más importantes en los sitios Web lo constituyen las imágenes. Estas pueden ayudar al usuario en la navegación, pueden mostrar la ubicación de contenidos o simplemente hacer más amigable la interfaz que se le muestra al usuario. La imágenes están formadas por figuras o trazos que constituyen objetos o símbolos que pueden tener asociado un significado o mensaje determinado.

Se realizó un estudio para determinar cuáles son las categorías más factibles a reconocer debido a su nivel de influencia sobre los usuarios y las consecuencias que éstas pueden traer. También se tuvo en cuenta la cantidad de contenido de esta categoría en Internet y la frecuencia con que se accede a contenido de este tipo. Las categorías identificadas son: nazi, satánica, religiosa y deportiva.

Después de tener ya las categorías definidas, se realizó una investigación para seleccionar los objetos a reconocer en cada una de las categorías. Este estudio se hizo teniendo en cuenta la frecuencia de aparición de éstos en la Web, el significado que tienen asociado, la historia, origen y el uso que se les ha dado.

Se decidió reconocer por la categoría nazi la Esvástica (figura 1.3) y el símbolo del racismo (figura 1.4).

Por la categoría de deportes, el logo de la mlb (figura 1.5) y el de la nba (figura 1.6) y, como representantes de sistemas operativos, el logo de linux (figura 1.7) y el windows (figura 1.7).

nazis:

Figura 1.3 Esvástica

Figura 1.4 Racismo

deportivos:

Figura 1.5 Logo de la mlb

Figura1.6 Logo de la nba

sistemas operativos:

Figura 1.7 Logo de Linux

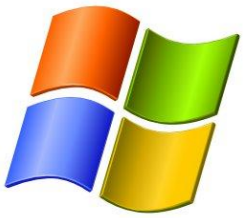


Figura 1.8 Logo de windows

1.7. Conclusiones

En este capítulo se ha cumplido el objetivo general planteado para el mismo, que era adentrarse en el tema de los filtros de contenido y más específicamente en el campo del reconocimiento de objetos. A la vez se ha hecho una valoración de las herramientas, tecnologías y lenguajes propuestos a utilizar para el desarrollo del presente sistema, así como también se escogió la metodología de software que se empleará. Después de culminar este capítulo se llega a las siguientes conclusiones:

- Como lenguaje de programación se escogió C++ por su rapidez y por tener un gran soporte de librerías.
- Como librería se seleccionó Opencv por su licencia y por brindar una mayor cantidad de funciones.
- En cuanto al IDE de desarrollo se decidió Eclipse con el plugin CDT porque consume menos recursos cuando esta en ejecución y posee un mejor proveedor de asistencia.
- Se concluyó utilizar como herramienta CASE el Visual Paradigm ya que brinda una solución factible en consecuencia con la metodología a utilizar.

Capítulo 2. Descripción del sistema

2.1 Introducción

El reconocimiento de objetos desordenados en el mundo real es un aspecto fundamental de muchos problemas de visión por computadora. Las escenas locales de las imágenes requieren características que no se vean afectadas por el desorden o la oclusión parcial. Las características deben ser, al menos parcialmente, invariantes a la iluminación y lo suficientemente distintivas para identificar objetos concretos entre muchas alternativas. La dificultad del problema del reconocimiento de objetos se debe en gran parte a la falta de éxito en la búsqueda de tales características en la imagen. Sin embargo, investigaciones recientes, por ejemplo Schmid y Mohr [13]), han demostrado que la eficacia del reconocimiento puede ser lograda mediante el uso de los descriptores incluidos en la muestra de la imagen local (objeto a reconocer) en un gran número de lugares repetibles en la escena. En este capítulo se describe el sistema elaborado basándose en esta investigación.

2.2 Descripción del sistema

A continuación se explicará todo lo relacionado a la teoría del sistema propuesto, así como la técnica utilizada y además se reflejan algunos estudios realizados anteriormente sobre el tema, los cuales se han tomado como base para el desarrollo de dicho sistema.

2.2.1 Investigación relacionada

La tarea de buscar imágenes coincidentes mediante el uso de un conjunto de puntos de interés locales, se remonta a la labor de Moravec (1981) [14], quien plantea la búsqueda de coincidencias en estéreo utilizando un detector de esquina. El detector de Moravec se ha mejorado por el Condado de Harris y Stephens (1988) [15] para que sean más pequeñas las variaciones de la imagen cerca de los bordes. Harris también demostró su valor para el movimiento eficiente de seguimiento de la estructura 3D (Harris, 1992) [16]. El detector de esquinas de Harris se utiliza ampliamente para muchas otras tareas que

corresponden al tratamiento de imágenes. Si bien estas características de estos detectores se llaman “Detectores de esquina”, no son sólo la selección de las esquinas, sino también cualquier lugar de la imagen que tiene grandes pendientes en todas las direcciones en una escala predeterminada.

Las primeras solicitudes fueron de corto alcance y se centraban en el seguimiento de objetos en movimiento, pero el enfoque posteriormente se amplió a los problemas más difíciles. Zhang (1995) [17] demuestra que es posible hacer coincidir las esquinas de Harris sobre una gran variedad de imágenes mediante el uso de una ventana alrededor de las correspondencias de cada esquina, para seleccionar probables lugares. Al mismo tiempo, enfoque similar fue desarrollado por Torr (1995) [18] para extraer el semejante a objetos rígidos dentro de una imagen en movimiento.

El trabajo de Schmid y Mohr (1997) [19] mostró que buscar características invariantes locales podría ampliarse a los problemas generales de reconocimiento de imágenes en la que una característica se compara con todas sus posibles semejantes en otras imágenes. Harris se utiliza para buscar coincidencias con una ventana de correlación, mediante una rotación invariante de los descriptores locales de la región de la imagen. Esto permitió a las características ser arbitrarias en virtud del cambio de orientación entre las dos imágenes. Además, se demostró que es posible lograr el reconocimiento general en virtud de la oclusión y el desorden coherente mediante la identificación de grupos de características combinadas. El detector de esquina de Harris es muy sensible a los cambios en la escala de imagen, por lo que no proporcionan una buena base para la comparación de imágenes de diferentes tamaños. Trabajos anteriores realizados por el autor (Lowe, 1999) [20] ampliaron la función de enfoque para lograr la escala invariante. En este trabajo también se describe un nuevo descriptor local que tiene más características, mientras que es menos sensible a las distorsiones de la imagen, tales como cambiar el punto de vista 3D. Este trabajo proporciona una corriente más profunda del desarrollo y análisis que el trabajo anterior, mientras que también presenta una serie de mejoras tanto en la estabilidad como en la función de invariancia.

Existe un gran número de investigaciones en la identificación de representaciones que son estables bajo cambio de escala. Uno de los primeros trabajos en este ámbito fue el de Crowley y Parker (1984) [21], que desarrolló una representación donde se identificaron los picos y crestas en el espacio vinculados en una estructura de árbol, la cual podría ser semejante entre imágenes con diferentes escalas. Un trabajo más

reciente sobre la búsqueda de coincidencias fue realizado por Shokoufandeh, Marsic y Dickinson (1999) [22], el cual proporciona una característica adicional y distintiva utilizando coeficientes wavelet. El problema de la identificación de una escala adecuada y coherente para la función de detección, se ha estudiado en profundidad por Lindeberg (1993, 1994) [23][24]. Él describe esto como un problema de escala y selección.

Recientemente, se ha producido una impresionante cantidad de trabajos en el ámbito de la búsqueda de descriptores locales que son invariantes a la plena transformación (Baumberg, 2000[25]; Tuytelaars y Van Gool, 2000[26]; Mikolajczyk y Schmid, 2002[27]; Schaffalitzky y Zisserman, 2002[28], Brown y Lowe, 2002[29]). Esto permite la adecuación a las características invariantes en una superficie plana en virtud de cambios en el gráfico 3D de proyección, en la mayoría de los casos por remuestreo de la imagen en un marco local afín. Sin embargo, ninguno de estos enfoques es todavía plenamente inherente a la invariancia,

Si bien el método que se presenta en este trabajo no es totalmente inherente a las características invariantes, un enfoque diferente que se utiliza en la función relativa del descriptor permite acercarse más a la invariancia. Este enfoque no sólo permite a los descriptores ser semejantes con fiabilidad a través de una gran variedad de distorsión, sino que también hace las funciones más robustas frente a cambios desde el punto de vista 3D de las superficies no planas. Otra ventaja es que es mucho más eficiente la extracción de características y la capacidad de identificar un mayor número de propiedades. Por otra parte, la inherencia a la invariancia es una valiosa propiedad para emparejar superficies planas muy grandes en virtud de los cambios.

Matas (2002)[30] ha demostrado que el máximo extremo estable de las regiones puede producir un gran número de características coincidentes con buena estabilidad. Mikolajczyk (2002) [31] ha desarrollado un nuevo descriptor que utiliza bordes locales, ofreciendo la capacidad de encontrar características estables, incluso cerca de los límites estrechos de formas superpuestas de fondo sobre el desorden. Selinger y Nelson (1998) [32] han mostrado buenos resultados con las características de las agrupaciones basadas en los contornos de la imagen. Del mismo modo, Pope y Lowe (2000) [33] utilizan las características sobre la base del agrupamiento jerárquico de la imagen de contornos, que son particularmente útiles para los objetos que carecen de textura detallada.

La historia de la investigación sobre el reconocimiento visual de patrones contiene un conjunto diverso de las propiedades de la imagen que se puede utilizar como característica en las comparaciones. Carneiro y Jepson (2002) [34] plantean describir la fase donde se obtienen las bases locales que representan la magnitud de las frecuencias espaciales, lo que es conveniente para mejorar la invariancia a la iluminación. Schiele y Crowley (2000) [35] han propuesto el uso de múltiples histogramas resúmenes de la distribución de las mediciones de imagen dentro de las regiones. Este tipo de función puede ser particularmente útil para el reconocimiento de la textura de objetos con formas variables.

2.2.2 Detección de los extremos

La primera etapa de la detección es la búsqueda, en todos los planos de la imagen, de posibles descriptores. Esto se aplica de manera eficiente mediante el uso de una diferencia Gaussiana que se muestra en la figura 2.1, función con el fin de detectar posibles puntos de interés que son invariantes a la escala y la orientación.

Se quieren identificar los lugares en los planos de la imagen que son invariables con respecto a la traslación, la expansión, y la rotación de la misma, y son mínimamente afectadas por el ruido y las pequeñas distorsiones. Lindeberg [24] ha demostrado que con el núcleo de Gauss y sus derivados se llega a un término general que sea invariante al cambio de escala, proporcionando buenos resultados. Para lograr la invariancia de rotación y un alto nivel de eficiencia, se ha optado por seleccionar las localidades más importantes, máximos y mínimos al aplicarse la función de Gauss en la escala espacial. Esto puede ser calculado de manera muy eficiente mediante la construcción de una pirámide de la imagen con remuestreo entre cada nivel. Además, sitúa los puntos clave en las regiones y escalas de variación alto, haciendo que estos lugares sean particularmente estables para la caracterización de la imagen.

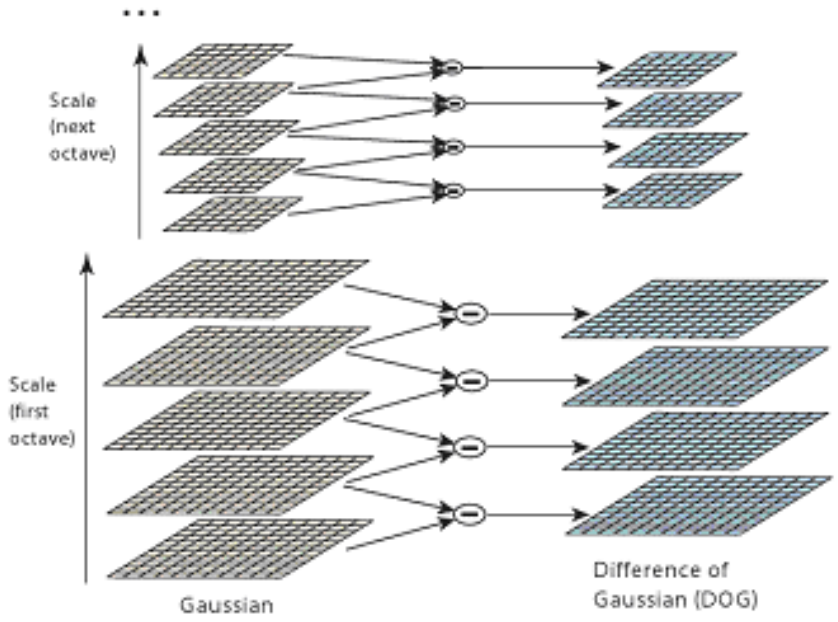


Figura 2.1 Diferencia Gaussiana

El método utilizado es eficiente y estable para detectar y caracterizar los máximos y mínimos de esta función. Como la función gaussiana 2D es separable, su convolución con la imagen de entrada puede ser calculada mediante la aplicación de dos pasos de la 1D gaussiana en función de la horizontal y vertical como se nuestra en la siguiente ecuación:

$$g(x) = \frac{1}{\sqrt{2\Pi\sigma}} \ell^{-x^2/2\sigma^2}$$

Los máximos y mínimos de la función escala-espacio se determinan mediante la comparación de cada píxel en la pirámide con sus vecinos como se muestra en la figura 2.2. En primer lugar, se compara el pixel con sus 8 vecinos en el mismo nivel de la pirámide. Si se trata de un máximo o un mínimo en este nivel, entonces el píxel se calcula en el siguiente nivel más bajo de la pirámide. Si el pixel sigue siendo mayor (o menor) que su equivalente y sus 8 vecinos, entonces la prueba se repite para el nivel anterior. Dado que la mayoría de píxeles serán eliminados dentro de pocas comparaciones, el costo de esta detección es pequeño.

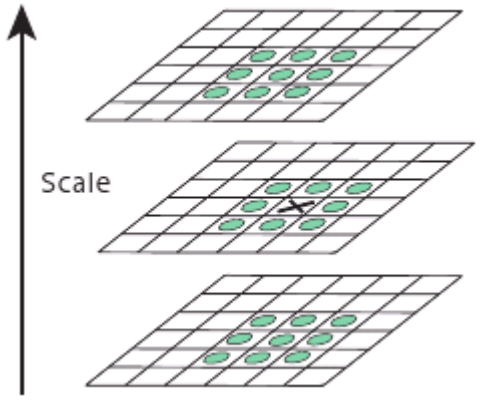


Figura 2.2 Comparación entre píxeles.

2.2.2.1. La estabilidad de las clave

Para caracterizar a la imagen en cada sitio clave se extraen muestras de gradientes y sus orientaciones. En cada pixel, la imagen de gradiente de magnitud, Mij, y la orientación, Rij, se calculan utilizando las diferencias de píxeles como se muestra a continuación:

$$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i;j+1})^2}$$

$$R_{ij} = \tan^2(A_{ij} - A_{i+i,j}, A_{i,j+1} - A_{ij})$$

La robustez a variaciones de la iluminación se ve reforzada por el cambio de magnitudes del gradiente, esto se logra aumentando el valor del umbral del objeto en 0,1 hasta alcanzar un umbral igual o semejante al de la escena o hasta llegar al máximo valor del gradiente. Esto reduce el efecto de un cambio en la dirección de la iluminación.

Para que los descriptores sean tan estables como sea posible a los cambios de iluminación o de contraste, la orientación está determinada por el pico en un histograma de la imagen local de las orientaciones del gradiente. En el histograma se abarca toda la gama de rotaciones, es decir 360 grados.

2.2.3 Localización de las claves o keypoint

En cada uno de los lugares candidatos, mediante un modelo detallado, se determina la localización y escala de estos lugares. Los keypoints son seleccionados en base a las medidas de su estabilidad. Una vez que un keypoint candidato se haya comprobado, mediante una comparación del píxel con sus vecinos, el siguiente paso es realizar un ajuste detallado de los datos de la ubicación y la escala.

En la primera aplicación de este enfoque (Lowe, 1999) [20] se encuentra simplemente en cada keypoints la ubicación y la magnitud de la muestra de un punto central. Sin embargo, Brown ha desarrollado recientemente un método con otro enfoque, sus experimentos mostraron que esto proporciona una mejora sustancial a la adecuación y la estabilidad. Su enfoque utiliza la expansión de Taylor (hasta el término cuadrático) de la escala de la imagen, D (x, y, σ), y cambió de manera que el origen está en el punto de muestreo como se muestra a continuación:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

Donde D y sus derivados son evaluados en el punto de muestreo y X= $(x,y,\sigma)^T$ es el desplazamiento de este punto. La ubicación del extremo, X, se determina tomando la derivada de esta función con respecto a X y se establece en cero, dando:

$$X = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$

Para lograr una buena estabilidad no es suficiente con solo rechazar los keypoints con bajo contraste. Además de esto, se debe comprobar que el keypoint mantenga el mismo valor en su dirección aunque se encuentre en uno de los bordes de la imagen.

En la figura 2.3 se muestran las fases de selección de los keypoint. (a) La imagen de 233x189 píxeles originales. (b) Los primeros 832 lugares en keypoints máximos y mínimos de la función de la diferencia de Gauss. Los keypoints se muestran como los vectores que indican la escala, orientación, y ubicación. (c) Después de la aplicación de un umbral mínimo de contraste, permanecen 729 keypoints. (d) El final los 536 keypoints que permanecen tras ser comparados en los bordes de la imagen.

2.2.4 Orientación de las claves o keypoint

Una o varias orientaciones son asignadas a cada keypoint, basado en las direcciones del gradiente de la imagen. Todas las futuras operaciones son realizadas sobre los datos de la imagen que se han obtenido en relación con la orientación asignada, escala, y ubicación de cada elemento, siendo estas características invariantes a posibles transformaciones. Mediante la asignación de una orientación coherente a cada keypoint, basada en las propiedades de la imagen, se puede lograr la invariancia a la rotación de la imagen

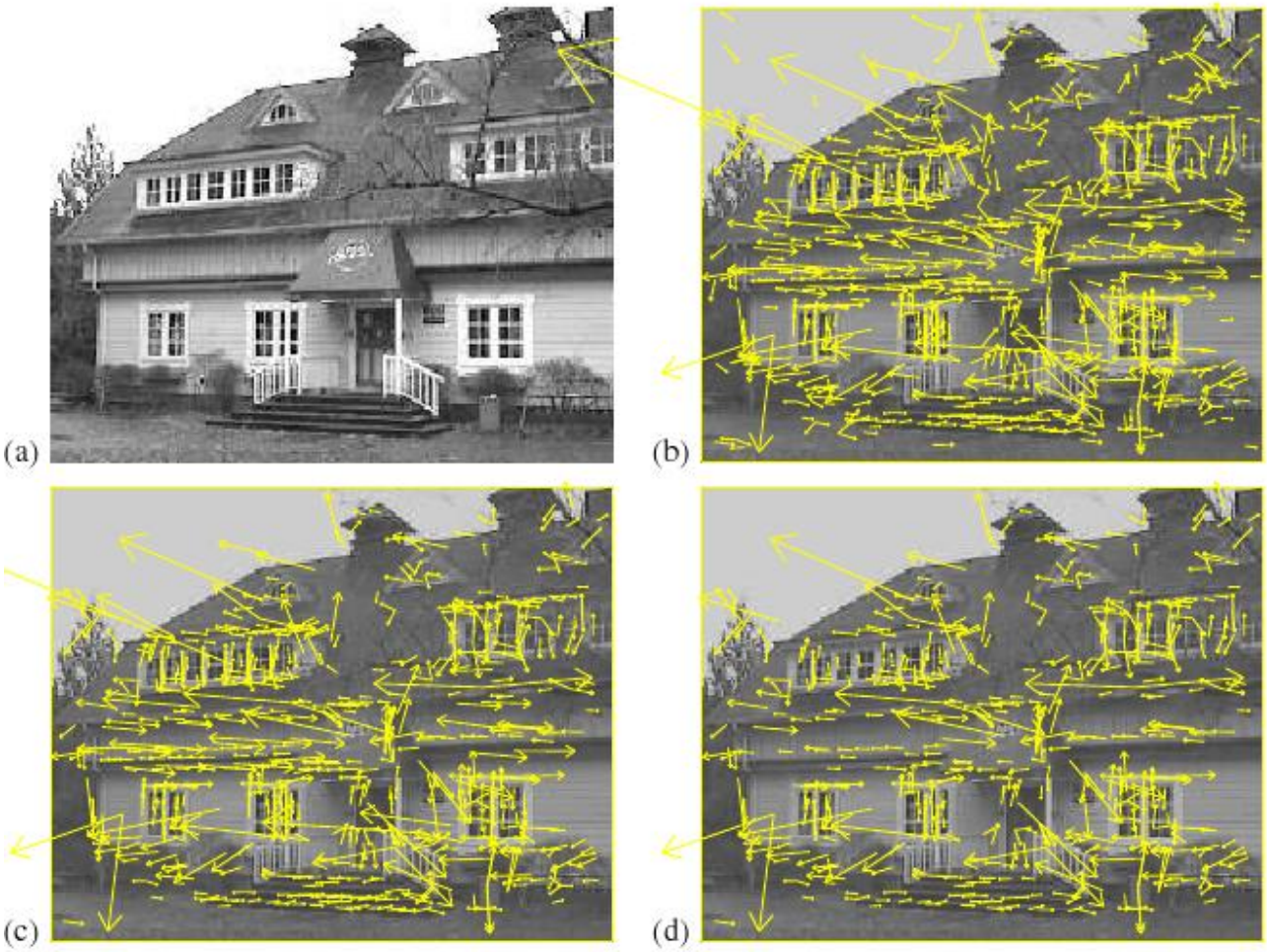


Figura 2.3 Fases de selección de los keypoint

La orientación del keypoint está formada por el gradiente de las orientaciones de los puntos de muestreo en una región alrededor del keypoint. Dicha orientación abarca toda la gama de orientaciones, los 360 grados. Los picos en el histograma corresponden a la orientación dominante de las direcciones locales de gradientes. El pico más alto en el histograma se detecta y, a continuación, cualquier otro local que se encuentra dentro de pico de 80% de la cima más alta también se utiliza para crear un keypoint con esa orientación. Por lo tanto, como es probable que existan múltiples picos de magnitud similar, para estos se crean keypoints en el mismo lugar y la misma escala, pero diferentes orientaciones. Sólo a alrededor del 15% de los puntos se le asignan múltiples orientaciones, pero contribuyen significativamente a la estabilidad de la misma.

2.2.5 Clave o Keypoint descriptor

La imagen de gradientes locales se mide en la región alrededor de cada keypoint como se muestra en la figura 2.4. Estos se transforman en una representación que permite niveles significativos de distorsión y de cambio de la iluminación.

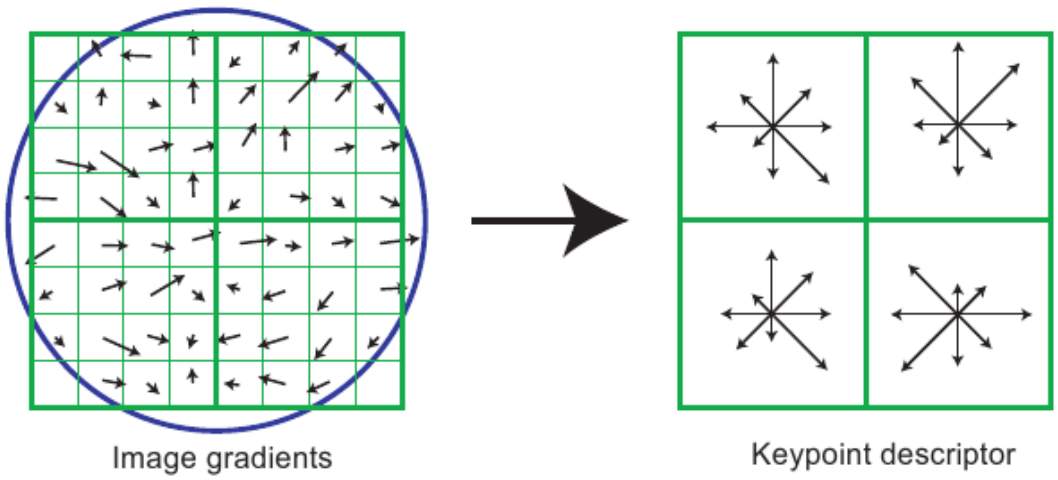


Figura 2.4 Gradientes locales

Un keypoint descriptor es creado por la orientación del gradiente en cada punto de muestra de la imagen en una región alrededor de la ubicación del punto de interés, como se muestra a la izquierda. Estas muestras son entonces acumuladas en el histograma de la orientación que resume el contenido en 4x4 subregiones como se muestra a la derecha, con la longitud de cada flecha correspondiente a la suma de las magnitudes del gradiente en esa dirección de la región.

El keypoint descriptor está formado por un vector que contiene los valores de todas las entradas del histograma de orientación que corresponden a las longitudes de las flechas en el lado derecho de la imagen anterior. La figura 2.4 muestra una matriz de 2x2, mientras que se comprobó que los mejores resultados se logran con una gama de histogramas de 4x4 con 8 componentes de orientación en cada uno. Por lo tanto, en este trabajo se utiliza un vector de 4x4x8 = 128 características del keypoint. El vector se normaliza a la unidad de longitud.

2.2.6 Reconocimiento de objetos

Una forma muy eficiente de buscar las coincidencias mediante los keypoint es a través de la identificación de sus vecinos más cercanos. Este vecino más cercano se define como el keypoint en la escena, con la mínima distancia euclidiana, con el vector del descriptor invariante.

Después de una larga investigación y un estudio minucioso, no se encontraron algoritmos que puedan definir con precisión los vecinos más cercanos de los puntos seleccionados. El vector del keypoint descriptor utilizado en el trabajo tiene una dimensión característica de 128 como se explicó anteriormente, por lo tanto, se ha utilizado un algoritmo que proporciona un resultado bastante aproximado, denominado algoritmo Best-Bin-First (BBF) (Beis y Lowe, 1997)[36], éste devuelve el vecino más cercano, con una elevada probabilidad. Una de las razones por las que el algoritmo BBF funciona especialmente bien para este problema, es que sólo tenemos en cuenta los casos en que el vecino más cercano es inferior a 0,6 veces la distancia más cercana hasta el momento y, por tanto, no es necesario resolver los casos más difíciles en los que muchos vecinos son muy similares a las distancias pero superiores a este valor. El proceso anterior se realiza partiendo de una distancia determinada de forma empírica, basada en pruebas realizadas, es decir, se considera un vecino cercano cuando el valor obtenido es menor que esta distancia.

Para maximizar el rendimiento del reconocimiento de objetos para los objetos pequeños o muy ocluidos, se quiere identificar los mismos con el menor número posible de características semejantes. Se ha determinado que es posible el reconocimiento fiable con tan sólo 3 características, como se muestra en la figura 2.5 donde se reconoce el símbolo buscado en un brazalete. En la figura 2.6 se distingue en una mesa con varios objetos más y en la figura 2.7 se identifica en un avión procedente de una imagen de un vídeo juego.

Figura 2.5 Reconocimiento en brazalete

Figura 2.6 Reconocimiento sobre mesa con diversos objetos

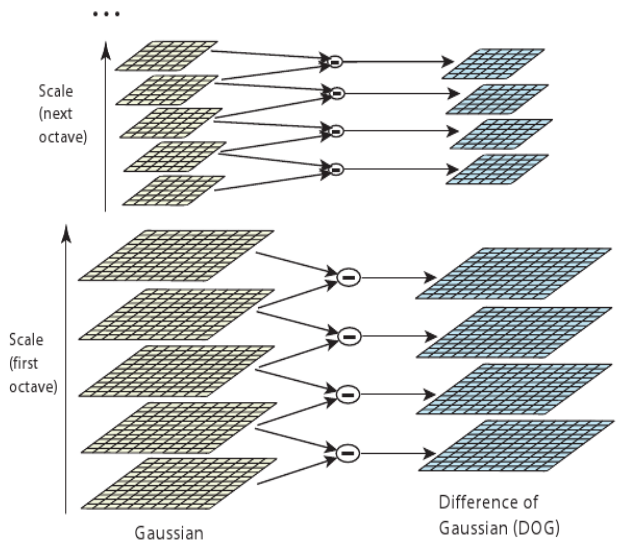
Figura 2.7 Reconocimiento en avión producto de una imagen de un vídeo juego

2.3 Historias de usuario

A continuación se relacionan las historias de usuario y las tareas asociadas a ellas, se presentan con la prioridad que tienen y los usuarios que se encargan de desarrollarlas.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Detección de los extremos
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Programador (Yosbel)	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2 semanas (100% culminado)
Descripción: La primera etapa de la computación es la búsqueda en todos los lugares de las escalas de la imagen. Esto se aplica de manera eficiente mediante el uso de una diferencia de Gaussiana, función con el fin de detectar posibles puntos de interés que son invariantes a la escala y la orientación.	
Observaciones:	

Prototipo de interfase:



Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Estudio sobre la técnica de reconocimiento de objetos mediante la búsqueda de claves o keypoints.	
Tipo de Tarea : Investigativa	**Puntos Estimados:** 1 semana

Fecha Inicio: 05/01/09	**Fecha Fin:** 09/01/09
Programador Responsable: programador (Yosbel)	
Descripción: Se hace una investigación relacionada con estudios anteriores sobre el reconocimiento de objetos basándose en la técnica de claves o keypoints	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Estudio de la biblioteca OpenCV para la extracción de claves o keypoints en la imagen.	
Tipo de Tarea : Investigativa	**Puntos Estimados:** 1 semana
Fecha Inicio: 05/01/09	**Fecha Fin:** 09/01/09
Programador Responsable: programador (Yosbel)	
Descripción: Se hace un estudio de la biblioteca OpenCV para la extracción de claves o keypoints.	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 1
Nombre Tarea: Cálculo del núcleo de Gauss y sus derivados.	

Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 12/01/09	**Fecha Fin:** 16/01/09
Programador Responsable: programador (Yosbel)	
Descripción: Se utiliza para calcular un término general que sea invariante al cambio de escala.	

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 1
Nombre Tarea: Cálculo de la diferencia entre píxeles.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 12/01/09	**Fecha Fin:** 16/01/09
Programador Responsable: programador (Yosbel)	
Descripción: Las diferencias de píxel son eficientes para calcular y proporcionar la suficiente precisión en cuanto a estabilidad.	

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Localización de las claves o keypoint
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Programador (Yosbel)	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2 semanas (100% culminado)
Descripción: En cada uno de los lugares candidatos, un modelo detallado está en condiciones de determinar la localización y escala. Los keypoints son seleccionados en base a las medidas de su estabilidad. Una vez que un candidato keypoint se haya comprobado, mediante una comparación del pixel con sus vecinos, el siguiente paso es realizar un ajuste detallado de los datos de la cercana ubicación, escala y proporción de curvaturas principales.	
Observaciones:	
Prototipo de interfase:	

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Calculo de la expansión de Taylor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 19/01/09	**Fecha Fin:** 23/01/09
Programador Responsable: programador (Yosbel)	
Descripción: Se utiliza para determinar la ubicación de la interpolación máxima.	

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Orientación de las claves o keypoint
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Programador (Yosbel)	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas

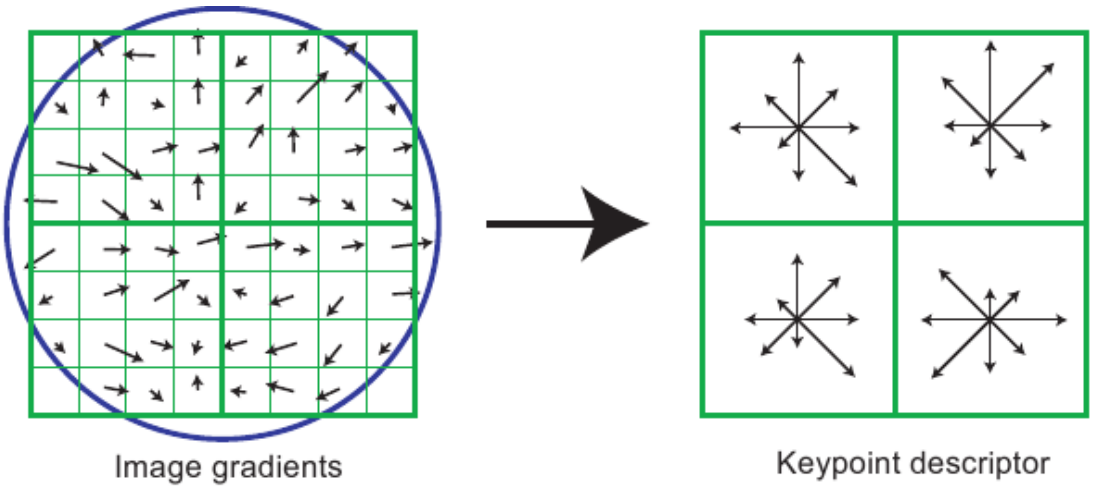
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2 semanas (100% culminado)
Descripción: Una o varias orientaciones son asignadas a cada keypoint basado en las direcciones de gradiente de la imagen. Todas las futuras operaciones son realizadas sobre los datos de la imagen que se ha transformado en relación con la orientación asignada, escala, y ubicación de cada elemento, con la invariancia a estas transformaciones.	
Observaciones:	
Prototipo de interfase:	

Tarea de Ingeniería		
Número Tarea: 1	**Número Historia de Usuario:** 3	
Nombre Tarea: Cálculo de gradiente.		
Tipo de Tarea : Desarrollo		**Puntos Estimados:** 1 semana
Fecha Inicio: 02/02/09		**Fecha Fin:** 06/02/09
Programador Responsable: programador (Yosbel)		

Descripción: Se utiliza para determinar la orientación de los componentes de cada uno de los keypoints.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 3
Nombre Tarea: Cálculo de la diferencia entre píxeles.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 09/02/09	**Fecha Fin:** 13/02/09
Programador Responsable: programador (Yosbel)	
Descripción: Se utiliza para determinar la orientación mejor definida del keypoint.	

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Clave o Keypoint descriptor
Modificación de Historia de Usuario Número: Ninguna	

Usuario: Programador (Yosbel)	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2 semanas (100% culminado)
Descripción: La imagen de gradientes locales se mide en la escala seleccionada en la región alrededor de cada keypoint. Estos se transforman en una representación que permite niveles significativos de distorsión de forma local y el cambio en la iluminación.	
Observaciones:	
Prototipo de interfase:  	

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 4
Nombre Tarea: Sumar vectores.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 16/02/09	**Fecha Fin:** 20/02/09
Programador Responsable: programador (Yosbel)	
Descripción: Se utiliza para determinar la orientación mejor definida de la clave o keypoint.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 4
Nombre Tarea: Normalizar vectores.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 23/02/09	**Fecha Fin:** 27/02/09
Programador Responsable: programador (Yosbel)	

Descripción: Se utiliza para hacer los cálculos con vectores más cercanos a la realidad.

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Reconocimiento de objetos
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Programador (Yosbel)	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 semanas
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2 semanas (100% culminado)
Descripción: Una forma muy eficiente de buscar las coincidencias mediante los keypoint es a través de la identificación de sus vecinos más cercanos. Este vecino más cercano se define como el keypoint con la mínima distancia euclidiana con el vector del descriptor invariante.	
Observaciones:	

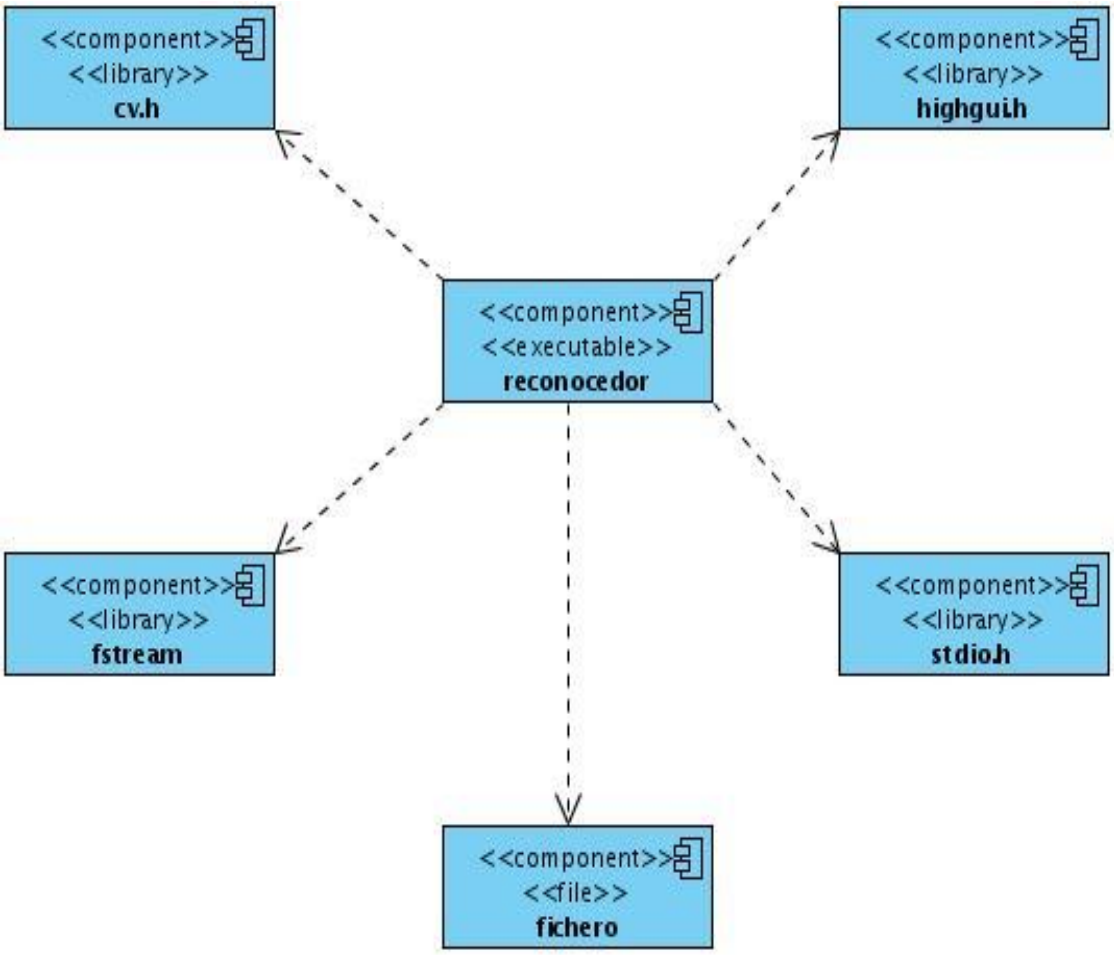
Prototipo de interfase:

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 5
Nombre Tarea: Estudio sobre la técnica de búsqueda de vecinos más cercanos.	
Tipo de Tarea : Investigativa	**Puntos Estimados:** 1 semana
Fecha Inicio: 02/03/09	**Fecha Fin:** 06/03/09
Programador Responsable: programador (Yosbel)	
Descripción: Se hace una investigación relacionada con estudios anteriores sobre búsqueda del vecino más cercano.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 5
Nombre Tarea: Calcular distancia euclidiana.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 09/03/09	**Fecha Fin:** 13/03/09
Programador Responsable: programador (Yosbel)	
Descripción: Se utiliza para identificar el vector más cercano al descriptor invariante.	

2.4 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc.



En la figura 2.8 se presenta el diagrama de componentes para el sistema que se propone.

Componente	Propósito
reconocedor	Es donde se encuentra el código fuente y el encargado de escribir el resultado en el fichero de salida.
cv.h	Este componente contiene las principales funciones de la librería OpenCV.
highgui.h	Este fichero es el encargado del manejo de ventanas y de imágenes del sistema.
fstream	Es la cabecera de la librería utilizada para el manejo de ficheros.
stdio.h	Es la cabecera de la librería utilizada para la entrada y salida estándar del lenguaje utilizado en el sistema.

2.5 Plan de releases

En este paso se define el plan de releases o iteraciones que se muestra en la tabla 2.1 para realizar las entregas intermedias y la entrega final. Tiene como entrada la relación de Historias de Usuario definidas previamente. Para colocar una historia en cada iteración, se tiene en cuenta la prioridad que definió el cliente para dicha Historia. Como resultado de la priorización de Historias se llegó a la siguiente planificación:

Tabla 2.1 Plan de release.

Release	Historias de Usuarios	Tiempo estimado (semanas)
1	1,2	4
2	3,4	4
3	5	2

2.6 Conclusiones

Después de la realización de este capítulo, en el cual se llevó a cabo un estudio sobre la técnica de reconocimiento de objetos basada en la selección de puntos claves se puede concluir lo siguiente:

- La técnica seleccionada tiene poco tiempo de existencia, y sin embargo, se ha venido experimentando un gran avance en su desarrollo.
- Se necesitan profundos conocimiento matemáticos para su uso.
- Se requiere estar probando constantemente lo que se va desarrollando.
- La utilización de éste método puede llevar a alcanzar resultados muy buenos.

Capítulo 3. Validación de la solución propuesta

3.1 Introducción

Para lograr un producto con calidad, es necesario trazarse un plan de pruebas desde el principio y darle seguimiento a los cambios y desarrollar iterativamente. En este capítulo se presentan los casos de pruebas o tests de aceptación que fueron realizados al sistema en cada una de las iteraciones. El cumplimiento de estos casos de pruebas fue el indicador a seguir para avanzar hacia la próxima iteración. Se expone además una relación de las funcionalidades con las que cuenta el sistema hasta la fecha.

3.2 Casos de pruebas

La programación extrema define entre iteración e iteración un conjunto de casos de pruebas o tests de aceptación para poder avanzar a una iteración superior. Durante el desarrollo del sistema de reconocimiento de objetos en imágenes, se diseñó un conjunto de casos de prueba al que fue sometido el sistema para comprobar el funcionamiento de acuerdo a las Historias de Usuario. Se definieron casos de prueba para todas las historias de usuario. A continuación se relacionan las pruebas más significativas realizadas a cada una de las Historias de Usuarios.

Caso de prueba para la historia de usuario: Detección de los extremos

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: *Detección de los extremos*

En esta historia de usuario se intenta probar que se extraigan todos los posibles puntos de interés, así como que estos sean realmente invariantes a las transformaciones.

Además se verificará si los puntos de interés extraídos son estables.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Detección de los extremos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que, dado el objeto a reconocer, se extraen de éste todos los posibles puntos de interés. Se verifica que los puntos extraídos son realmente invariantes a las transformaciones.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que se extraigan correctamente los puntos de interés.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Detección de los extremos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que los puntos de interés extraídos son lo suficientemente estables como para poder garantizar buenos resultados.	

Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.
Resultado Esperado: Que los puntos de interés extraídos sean estables.
Evaluación de la Prueba: Prueba satisfactoria.

Caso de prueba para la historia de usuario: Localización de las claves o keypoint.

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: *Localización de las claves o keypoint*

En esta historia de usuario se intenta probar que los puntos de interés extraídos corresponden a los máximos y los mínimos de la función Gaussiana.

También se pretende comprobar que los puntos de interés extraídos son invariantes a la iluminación al aplicarles un umbral mínimo.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Localización de las claves o keypoint
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que al aplicar un umbral mínimo de contraste, los keypoint seleccionados son invariantes al cambio de iluminación.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	

Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.
Resultado Esperado: Que los puntos de interés extraídos sean invariantes al cambio de iluminación.
Evaluación de la Prueba: Prueba satisfactoria.

Caso de prueba para la historia de usuario: Orientación de las claves o keypoint.

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: *Orientación de las claves o keypoint*

En esta historia de usuario se intenta probar que las orientaciones asignadas a cada descriptor son las correctas.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Orientación de las claves o keypoint
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si las orientaciones asignadas a cada descriptor son las correctas.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que los puntos de keypoint extraídos sean invariantes a la rotación de la imagen, producto de una correcta asignación de sus orientaciones.	

Evaluación de la Prueba: Prueba satisfactoria.

Caso de prueba para la historia de usuario: Clave o Keypoint descriptor.

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: *Clave o Keypoint descriptor*

En esta historia de usuario se intenta probar que el histograma de la orientación es correcto.

Además se verificará que el vector que representa el descriptor sea normalizado correctamente.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Clave o Keypoint descriptor
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el histograma de la orientación es correcto.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el histograma de la orientación sea correcto para así garantizar niveles significativos de distorsión de forma local y el cambio en la iluminación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Clave o Keypoint descriptor
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si la normalización del vector que representa al descriptor es correcta.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el vector que representa al descriptor sea correctamente normalizado.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de prueba para la historia de usuario: Reconocimiento de objetos.

Esta sección cubre el conjunto de pruebas que se le realizaron a la historia de usuario: *Reconocimiento de objetos.*

En esta historia de usuario se intenta probar que la identificación del vecino más cercano de cada keypoint es realizada correctamente, así como verificar que con solo 3 keypoint semejantes basta para concluir que el objeto se encuentra en la imagen.

Además se comprobará que el sistema es capaz de reconocer objetos con cambios es su escala, rotados, con cambios de colores o cambio en su plano de representación.

También se comprobará que se pueden cargar diferentes extensiones de imágenes y que la fiabilidad del sistema es superior al 80%.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que la identificación del vecino más cercano de cada keypoint sea realizada correctamente.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que se logre localizar el vecino más cercano de cada keypoint.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba que con solo identificar 3 keypoint en la escena es suficiente para afirmar la presencia del objeto en la misma.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	

Resultado Esperado: Que identificando solo 3 keypoint sea capaz de reconocer el objeto buscado en la escena.
Evaluación de la Prueba: Prueba satisfactoria.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 3	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el sistema es capaz de reconocer un objeto transformando su escala.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el objeto sea reconocido por el sistema.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 4	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el sistema es capaz de reconocer un objeto rotado.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el objeto sea reconocido por el sistema.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 5	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el sistema es capaz de reconocer un objeto transformando su plano.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el objeto sea reconocido por el sistema.	

Evaluación de la Prueba: Prueba satisfactoria.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 6	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el sistema es capaz de reconocer un objeto al cambiarle sus colores.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que el objeto sea reconocido por el sistema.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 7	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba si el sistema es capaz de cargar imágenes con distintas extensiones.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	

Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.
Resultado Esperado: Que las imágenes sean cargadas sin problemas.
Evaluación de la Prueba: Prueba satisfactoria.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 8	**Nombre Historia de Usuario:** Reconocimiento de objetos
Nombre de la persona que realiza la prueba: Programador (Yosbel)	
Descripción de la Prueba: Se ejecuta la aplicación y se comprueba la efectividad del sistema.	
Condiciones de Ejecución: La aplicación no puede contener errores de ningún tipo en su compilación.	
Entrada / Pasos de ejecución: Se compila la aplicación y se ejecuta.	
Resultado Esperado: Que la efectividad sea superior al 80%.	
Evaluación de la Prueba: Prueba satisfactoria.	

3.3 Resultados obtenidos

En este epígrafe se relacionan los resultados obtenidos durante la elaboración del proyecto por el equipo de desarrollo del sistema reconocedor de objetos en imágenes. Como producto de este trabajo, se

encuentra disponible una versión 0.9 del producto, la cual comenzará a utilizarse en el proyecto MOCIC y se puede esperar mejores resultados en versiones superiores.

3.3.1 Acerca del tiempo de desarrollo

El problema de la necesidad de implementar un reconocedor de objetos en imágenes era una situación problemática que se venía dando desde la concepción misma del Filtro de Paquetes por Contenido (FILPACON), debido a que los existentes en la actualidad son propietarios y el único caso libre que se conoce (POESIA), tiene una muy baja fiabilidad.

Debido a la necesidad de implementar una herramienta capaz de suplir las carencias del proyecto, surge la idea de realizar este sistema. A finales de 2008, se comienza el desarrollo del reconocedor de objetos en imágenes con tareas de investigación. Con un solo programador, que a la vez se convertía en diseñador, analista y encargado de pruebas, se logró realizar un producto atractivo y totalmente funcional. El producto estará sujeto a mejoras ya que se implementarán más funcionalidades para su uso en otros ámbitos.

Para la realización de este producto se estudiaron librerías de procesamiento de imágenes, así como todo lo referente al tratamiento y extracción de patrones en imágenes. El tiempo de desarrollo fue relativamente corto y se cumplieron todas las expectativas que se plantearon durante la planificación inicial.

3.3.2 Acerca de las funcionalidades obtenidas.

Entre algunas de las funcionalidades que presenta el sistema en su primera versión se encuentran las siguientes:

- Permite gestionar imágenes de varios formatos incluidos los más usados actualmente.
- Permite reconocer varios objetos en la misma escena.

- Permite reconocer objetos que se han rotado, que se les ha cambiado la escala, el color o el plano.
- Permite clasificar en qué categoría se encuentra la escena dada.
- Es fácil de utilizar, no se necesitan conocimientos informáticos amplios para su uso.

Al sistema desarrollado se le realizó un conjunto de pruebas para obtener métricas necesarias para su posible utilización como se muestra en la tabla 3.1. Las métricas principales fueron: fiabilidad con que reconoce y tiempo que demora en el procesamiento. Las pruebas para determinar la fiabilidad del sistema se realizaron con 130 imágenes que contenían un total de 150 símbolos y, con esta muestra, la fiabilidad obtenida fue 94,6% como se muestra en la tabla que aparece debajo. Este valor se considera bueno para una primera versión si se compara con otros de su tipo existente. Las pruebas para determinar el tiempo de procesado se llevaron a cabo con 10 objetos y 10 escenas y el tiempo resultante fue 5060ms, es decir, 5,06 segundos.

Tabla 3.1 Resultados

Imágenes procesadas	Correctamente Clasificadas	Incorrectamente Clasificadas	Por ciento
Imágenes con la presencia de los objetos	124	6	95,4%
Imágenes sin la presencia de los objetos	122	8	93,8%
Total de objetos	142	8	94,6%

3.4 Conclusiones

Después del desarrollo de este capítulo, donde se presentaron los casos de pruebas y se analizaron los resultados obtenidos, se llega a las siguientes conclusiones:

- Se obtuvieron los casos de pruebas, los cuales fueron capaces de guiar el desarrollo del sistema hasta obtener los resultados esperados.
- El sistema propuesto tiene una fiabilidad elevada en su reconocimiento, es capaz de categorizar imágenes con alto por ciento de veracidad.
- El tiempo de procesado es bueno, por lo que se considera que se cumplieron los objetivos de este capítulo.

Conclusiones

Con la culminación del presente trabajo de diploma, se cumple con el objetivo trazado en el mismo mediante el desarrollo de un módulo que permite reconocer objetos de forma automatizada, con la facilidad de ser integrado al motor de categorización de un sistema de filtrado por contenidos. Se cumplió además con los siguientes aspectos:

- Se obtuvieron los descriptores más relevantes de las imágenes para el reconocimiento de objetos.
- Se eligió un algoritmo de entrenamiento y clasificación eficiente.
- Se implementó un módulo capaz de reconocer objetos dentro de imágenes con una fiabilidad de 94,6%.
- Se logró la incorporación del módulo al motor de clasificación inteligente de contenido MOCIC, el cual podrá ser usado por el sistema de filtrado FILPACON.

Recomendaciones

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

- Continuar el estudio de las tendencias actuales en el campo del procesamiento de imágenes y específicamente en la vertiente de la extracción de puntos claves.
- Mejorar el método propuesto de selección de puntos claves con el fin de mejorar su fiabilidad.
- Continuar con la optimización del sistema para lograr mejoras en el tiempo de procesamiento.
- Estudiar otras posibles características invariantes en las imágenes que, unidas a los puntos claves, pudieran proporcionar un resultado superior.
- Implementar el reconocedor utilizando momentos de HU y comparar los resultados.

Referencias Bibliográficas

[1] EMC. "Estadísticas que reflejan el tamaño de Internet". [En línea] [Citado e 02 febrero 2009]. Disponible en: <http://www.averlo.com/notas/ciencia/0307/que-grande-internet-actualidad.html>.

[2] Netcraft. "Estadísticas que reflejan el tamaño de Internet". [En línea] [Citado 02 febrero 2009].Disponible en: <http://www.larioja.com/20061102/sociedad/internet-alcanza-hito millones_200611021337.html>.

[3] Review, I.F. "Estadísticas de pornografía en Internet". [En línea] [Citado 02 febrero 2009]Disponible en: <http://internet-filter- review.toptenreviews.com/internet-pornography-statistics.html>.

[4] ISS. Sitios de contenido extremista aumentan en un 42,4% en el 2005. [En línea] [Citada 04 febrero 2009]Disponible en: <http://www.diarioti.com/gate/n.php?id=10082>.

[5] Internet, O.e.d. "Drogas en la red". [En línea] [Citado 04 febrero 2009] Disponible en: <http://www.electronicafacil.net/telefonia/Article6775.html >.

[6] TopTenREVIEWS. Internet Filter Software Review 2008. [En línea] [Citado 04 febrero 2009]Disponible en: <http://internet-filter-review.toptenreviews.com/index.html#anchor>.

[7] POESIA project. Poesia Project. [En línea] [Citado 05 febrero 2009] Disponible en: <http://www.poesia-filter.org>.

[8] S.Watanabe, Pattern Recognition: Human and Mechanical, Wiley, 1985.

[9] Ivar Jacobson et al. El Proceso Unificado de Desarrollo de Software. Addison Wesley Publishing Company, Noviembre de 2000, pp 30-34.

[10] Alexander Vera Tasamá, Andrés F. Ramírez Sánchez, Daniel Moreno Martínez, Milton J. Alzate Silva. Aplicación Didáctica para el Procesamiento de Imágenes Digitales Usando Interfaz Gráfica de Usuario en

MATLAB [En línea] [Citado 05 febrero 2009]. Disponible en <http://www.compelect.com.co/otros/diamatlab/2008/Aplicacion_Didactica_PRIM.pdf>.

[11]. Metodologias RUP y XP - [PROCESOS DE DESAROLLO]. [En linea] [Citado 10 febrero 2009] Disponible en: <http://jackopc.blogspot.com>

[12] Programación extrema. [En línea] [Citado 06 febrero 2009]Disponoble en: <http://www.programacionextrema.org.>.

[13] Schmid, C., and R. Mohr, "Local grayvalue invariants for image retrieval," IEEE PAMI, 19, 5 (1997), pp. 530–534.

[14] Moravec, H. 1981. Rover visual obstacle avoidance. In International Joint Conference on Artificial Intelligence, Vancouver, Canada, pp. 785-790.

[15] Harris, C. and Stephens, M. 1988. A combined corner and edge detector. In Fourth Alvey Vision Conference, Manchester, UK, pp. 147-151.

[16] Harris, C. 1992. Geometry from visual motion. In Active Vision, A. Blake and A. Yuille (Eds.), MIT Press, pp. 263-284.

[17] Zhang, Z., Deriche, R., Faugeras, O., and Luong, Q.T. 1995. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Artificial Intelligence, 78:87-119.

[18] Torr, P. 1995. Motion Segmentation and Outlier Detection, Ph.D. Thesis, Dept. of Engineering Science, University of Oxford, UK.

[19] Schmid, C., and Mohr, R. 1997. Local gray value invariants for image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(5):530-534.

[20] Lowe, D.G. 1999. Object recognition from local scale-invariant features. In International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157.

[21] Crowley, J. L. and Parker, A.C. 1984. A representation for shape based on peaks and ridges in the difference of low-pass transform. IEEE Trans. on Pattern Analysis and Machine Intelligence, 6(2):156-170.

[22] Shokoufandeh, A.,Marsic, I., and Dickinson, S.J. 1999. View-based object recognition using saliency maps. Image and Vision Computing, 17:445-460.

[23] Lindeberg, T. 1993. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. International Journal of Computer Vision, 11(3): 283-318.

[24] Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.

[25] Baumberg, A. 2000. Reliable feature matching across widely separated views. In Conference on Computer Vision and Pattern Recognition, Hilton Head, South Carolina, pp. 774-781.

[26] Tuytelaars, T., and Van Gool, L. 2000. Wide baseline stereo based on local, affinely invariant regions. In British Machine Vision Conference, Bristol, UK, pp. 412-422.

[27] Mikolajczyk, K., and Schmid, C. 2002. An affine invariant interest point detector. In European Conference on Computer Vision (ECCV), Copenhagen, Denmark, pp. 128-142.

[28] Schaffalitzky, F., and Zisserman, A. 2002. Multi-view matching for unordered image sets, or 'How do I organize my holiday snaps?"' In European Conference on Computer Vision, Copenhagen, Denmark, pp. 414-431.

[29] Brown, M. and Lowe, D.G. 2002. Invariant features from interest point groups. In British Machine Vision Conference, Cardiff, Wales, pp. 656-665.

[30] Matas, J., Chum, O., Urban, M., and Pajdla, T. 2002. Robust wide baseline stereo from maximally stable extremal regions. In British Machine Vision Conference, Cardiff, Wales, pp. 384-393.

[31] Mikolajczyk, K. 2002. Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France.

[32] Nelson, R.C., and Selinger, A. 1998. Large-scale tests of a keyed, appearance-based 3-D object recognition system. Vision Research, 38(15):2469-88.

[33] Pope, A.R., and Lowe, D.G. 2000. Probabilistic models of appearance for 3-D object recognition. International Journal of Computer Vision, 40(2):149-167.

[34] Carneiro, G., and Jepson, A.D. 2002. Phase-based local features. In European Conference on Computer Vision (ECCV), Copenhagen, Denmark, pp. 282-296.

[35] Schiele, B., and Crowley, J.L. 2000. Recognition without correspondence using multidimensional receptive field histograms. International Journal of Computer Vision, 36(1):31-50.

[36] Beis, J. and Lowe, D.G. 1997. Shape indexing using approximate nearest-neighbour search in highdimensional spaces. In Conference on Computer Vision and Pattern Recognition, Puerto Rico, pp. 1000-1006.

Bibliografía

- Vice-Rectoría de Formación Departamento de Practica Profesional. Propuesta de Guía para la presentación del Trabajo de Diploma. 2005-2006.

- IDC. The Expanding Digital Universe [En línea] [Citado 15 marzo 2009]. Disponible en: <http://www.emc.com/collateral/analyst-reports/expanding-digital-idc-white-paper.pdf>.

- INFOBAE. Que tan grande es internet [En línea] [Citado 15 marzo 2009]. Disponible en: <http://www.infobae.com/contenidos/305016-100796-0-Qu%C3%A9-tan-grande-es-internet-la-actualidad>.

- TopTenREVIEWS. Internet Filter Software Review 2008 [En línea] [Citado 20 marzo 2009]. Disponible en: <http://internet-filter-review.toptenreviews.com/index.html#anchor>.

- LTU technologies. LTU Technologies: image search and image filter software [En línea] [Citado 20 marzo 2009]. Disponible en: <http://www.ltutech.com/en/technology-and-products.image-filter.html>.

- POESIA project. Poesia Project [En línea] [Citado 20 marzo 2009]. Disponible en: <http://www.poesia-filter.org>.

- Tinku Acharya y Ajoy K. Ray. IMAGE PROCESSING Principles and Applications. John Wiley and & Sons, 2005.

- TIOBE. TIOBE Programming Community Index for June 2008 [En linea] [Citado 25 marzo 2009]. Disponible en: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

- Ivar Jacobson et al. El Proceso Unificado de Desarrollo de Software. Addison Wesley Publishing Company, Noviembre de 2000, pp 30-34.

- Craig Larman. UML y Patrones: Introducción al análisis y diseño orientado a objetos. Addison Wesley Longman.

- dzoom. Formatos de almacenamiento de imágenes digitales [En línea] [Accedida 25 marzo 2009]. Disponible en: <http://www.dzoom.org.es/cont-38-formato-fichero-imagen-fotografia.html>.

Siglario

UCI: Universidad de las Ciencias Informáticas

FILPACON: Filtrado de Paquetes por Contenido

OSRI: Oficina de Seguridad para las Redes Informáticas

BDUC: Base de Datos de URLs Categorizadas

HTML: Hyper Text Marked-up Language

Web: Red

UML: Unified Modeling Language

POESIA: Public Open-source Environment for a Safer Internet Access

OPTENET: Optimal Internet

URL: Uniform Resource Locator

RUP: Rational Unified Process

CASE: Computer Aided Software Engineering

GNU: GNU is Not Unix

Linux: Es el núcleo (kernel) del sistema operativo libre GNU.

GPL: General Public License

IDE: Integrated Development Enviroment

CDT: C/C++ Development Tools

BBF: Best-Bin-First

3D: Tres dimensiones

Glosario

FILPACON: Es un sistema que pretende ser flexible y fiable para regular los contenidos nocivos de Internet.

HTML: Lenguaje de Marcas de Hipertexto. Es el lenguaje de marcado predominante para la construcción de páginas Web.

Web: La traducción literal de esta palabra inglesa es tela de araña, pero en términos informáticos significa mucho más que eso.

UML: Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

POESIA: Es un sistema de filtrado de código abierto bajo la licencia GPL.

OPTENET: Sistema de filtrado con más fuerza en Europa.

URL: Se refiere al texto que identifica a una página Web.

RUP: Metodología de desarrollo de software creada por la Corporación Rational.

GNU: Sistema operativo cuyo nombre es un acrónimo recursivo que significa “GNU No es Unix”.

GPL: Licencia Pública General. Es una licencia creada por la Free Software Foundation (FSF) y orientada principalmente a los términos de distribución, modificación y uso del software.

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

CDT: Es un pluging para Eclipse el cual permite la programación en el lenguaje C/C++

Java: Un lenguaje de programación de alto nivel, orientado a objetos.

Privativo: El software no libre (también llamado software propietario, software privativo, software privado, software con propietario o software de propiedad) se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido.

Subversion: Es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido.

Oclusión: El término se utiliza para describir situaciones donde el objeto a reconocer se encuentra debajo de otro, es decir, no se encuentra en primer plano en la escena.

BBF: Este algoritmo devuelve el vecino más cercano, con una elevada probabilidad.

Invariancia: Es algo que no cambia al aplicarle un conjunto de transformaciones.

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

Keypoint: El término se utiliza para referirse a las claves o puntos distintivos.

Anexos

Anexo No. 1 Funcionamiento General de Filpacon.

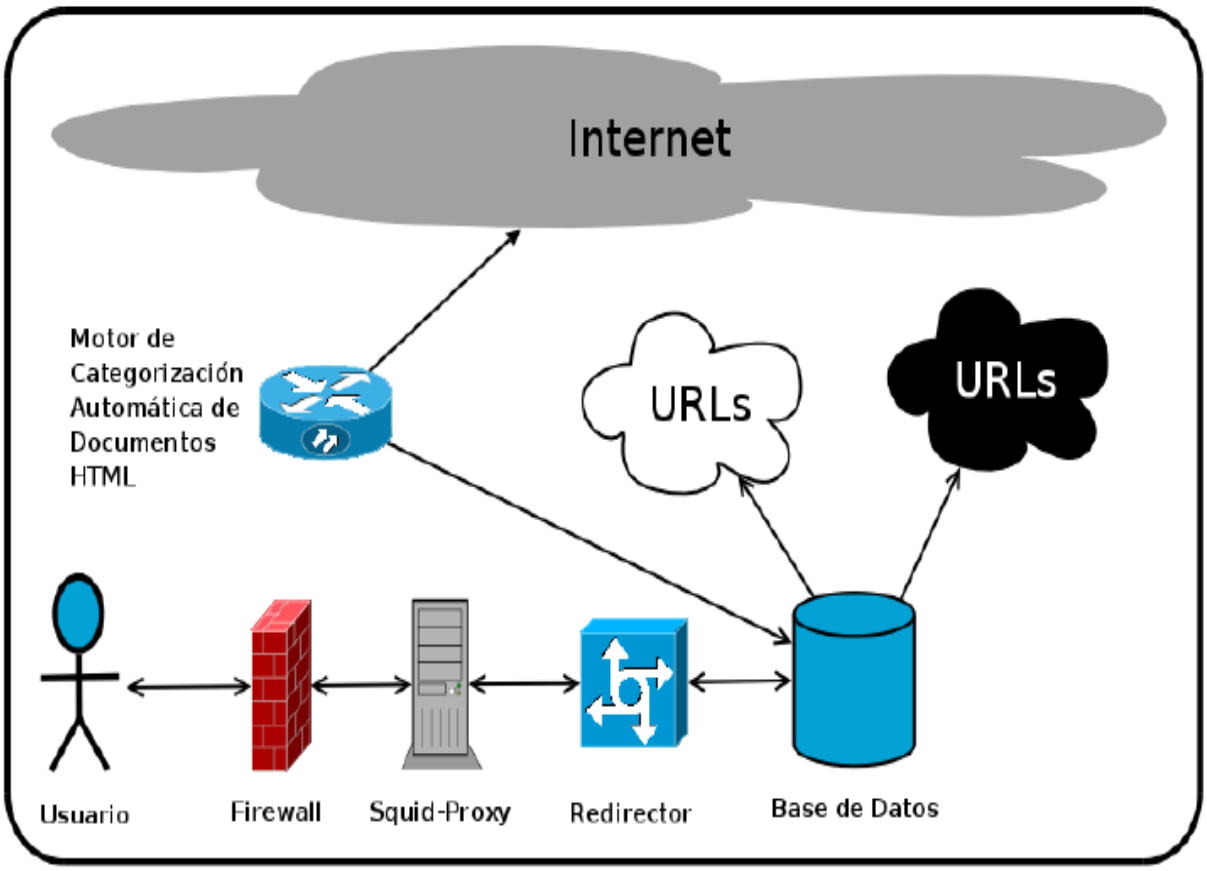


Figura 1. Funcionamiento General de Filpacon

Anexo No. 2 Arquitectura de MOCIC.

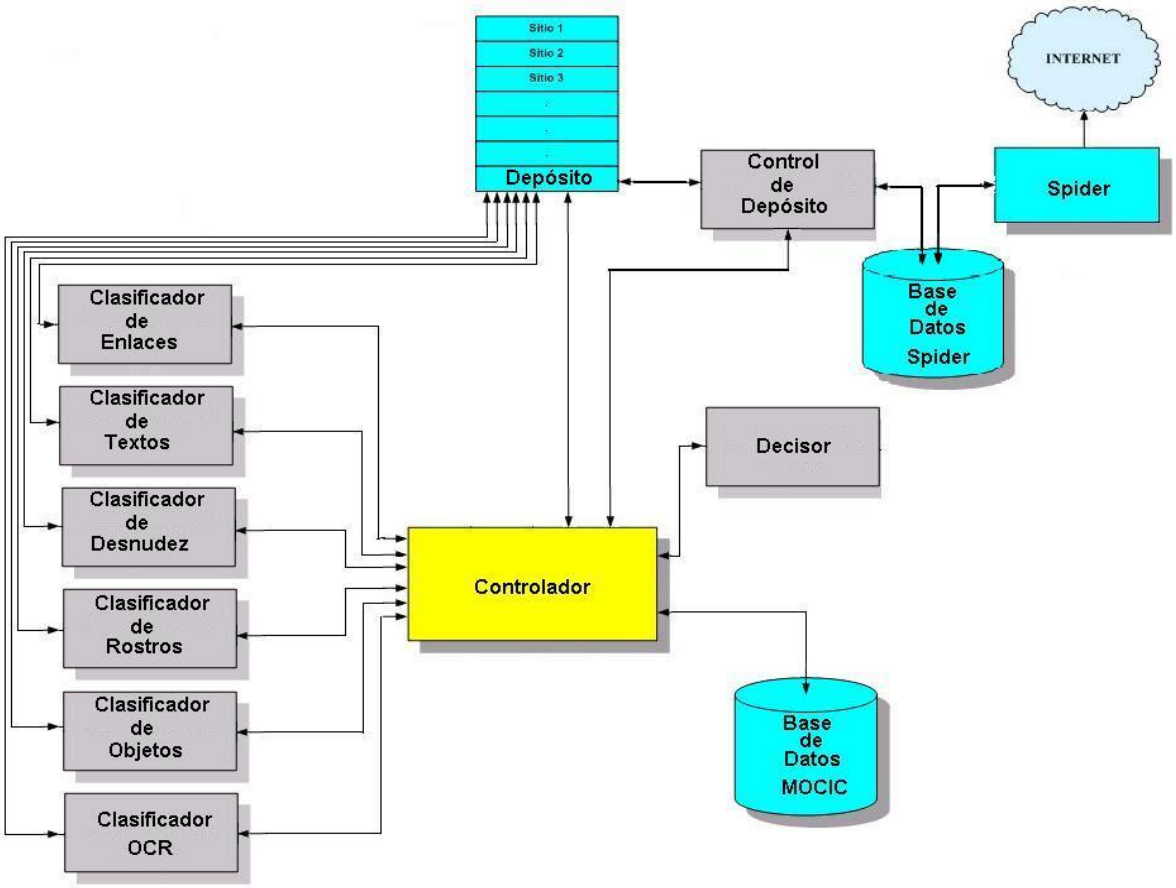


Figura 2. Arquitectura de MOCIC