Universidad de las Ciencias Informáticas

Facultad 7

Título: Video sensor de seguimiento de objetos para el sistema de vigilancia por cámaras

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Rolando Payán Mosqueda

Tutores: Msc. Pedro Medina Riesgo

Lic. Fernando Echemendia Tourt

Ciudad de La Habana, Junio de 2009

"Año del 50 aniversario del triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo 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 30 días del mes de junio del año 2009.

_____

Rolando Payán Mosqueda

Autor

_____ _____

Msc. Pedro Medina Riesgo

Tutor

Ing. Fernando Echemendia Tourt

Tutor

DATOS DE CONTACTO

Tutores:

Msc. Pedro Medina Riesgo:

Graduado de Licenciatura en Ciencias de la Computación en la Universidad de La Habana. Inicialmente formó parte del Grupo de Procesamiento Digital de Imágenes de la Facultad 7. Actualmente se encuentra en prestación de servicios en el Centro de Identidad, trabajando en temas relacionados con la detección y seguimiento de rostros. Es profesor en la Universidad de las Ciencias Informáticas, posee categoría docente de Asistente y categoría científica de Máster en Ciencias. Se encuentra trabajando en su tesis doctoral.

Correo: medina@uci.cu

Ing. Fernando Echemendia Tourt:

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, en el año 2008. Participó en actividades científicas estudiantiles siendo miembro del Grupo de Procesamiento de Imágenes de la facultad 7. Se desempeña actualmente como Jefe de Proyecto en dicho grupo.

Correo: fechemendia@uci.cu

AGRADECIMIENTOS

A mi familia, por su amor.

A mis tutores, por su apoyo.

A la comunidad, por su amistad.

A la Revolución por la oportunidad de servirle.

DEDICATORIA

A mis tutores por apoyarme, ayudarme y guiarme en todo el transcurso de la tesis.

A mis profesores por proporcionarme las herramientas necesarias en mi formación.

A mi familia:

Una casa será fuerte e indestructible cuando esté sostenida por estas cuatro columnas: padre valiente, madre prudente, hijo obediente, hermano complaciente.

Confucio.

A mi madre

Decía José Martí: que Hay un solo niño bello en el mundo y cada madre lo tiene. También se podría decir que, hay una sola madre bella en el mundo y cada hijo la tiene.

Mami te quiero con todo mi corazón gracias por siempre estar conmigo, apoyarme, aconsejarme y ser tan buena, me siento superfeliz de tenerte como madre y ojalá que todas las madres del mundo fueran como tú, te amo.

A mi padre:

A ese hombre, ese hombre que todo nos ha dado que casi de sol a sol a trabajado, que nos a dado todo su amor que es lo más importante. Su hijo le agradece su amor, bondad y su mal genio. Gracias por los consejos que me has dado y que no puedo olvidar, a ti te tengo que agradecer todo esto y mucho más, nunca te podré pagar todo lo que has hecho por mí.

A mi hermano:

Gracias por todos los momentos que hemos compartido, momentos llenos de sentimiento y pensamientos compartidos, sueños y anhelos, secretos, risas y lágrimas, y sobre todo amistad.

A mis tíos, a mis tías, a mi primo, mis abuelos y mis abuelas los quiero mucho.

Y para finalizar quería agradecer a mis amigos de la comunidad, sigan así tal cuales son, apóyense mucho, y recuerden aquel mensaje que ponía el cliente cuando estaba cargando: Take everything in moderation even…… y no le pongo la última palabra porque esta frase se le aplica a todo en la vida.

RESUMEN

En la Facultad 7, el Grupo de Procesamiento de Imágenes y Señales Digitales (GPI), se encuentra desarrollando un Sistema de Video Vigilancia que cuenta con los siguientes módulos: gestión de flujos de video y cámaras, detección de movimiento, detección de formas e identificación de chapas de autos y necesita un módulo que se encargue del seguimiento de objeto en secuencias de video.

El presente trabajo tiene como objetivo implementar un video sensor de Seguimiento de Objetos para el Sistema de Video Vigilancia que desarrolla actualmente el proyecto GPI. Para el desarrollo de estos flujos de trabajo se siguieron los pasos que propone el Rational Unified Process (RUP), como metodología de desarrollo, como herramienta Rational Rose con el lenguaje de modelado UML y el Visual Studio Team System. Como lenguaje de programación se utilizó el C#.NET.

La utilización de los video sensores puede ofrecer una serie de servicios, de valor añadido en el área de la seguridad, o bien ser empleados en el área de la medicina, la industria, el entretenimiento digital, entre otras. El sistema obtenido completa el Sistema de Video Vigilancia desarrollado por GPI y podrá ser utilizado con múltiples fines.

PALABRAS CLAVE

Seguimiento de objetos, video sensor.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los sistemas de video vigilancia tienen cada vez mayor demanda, especialmente, para garantizar la seguridad de interiores y alrededores de edificios. La presencia de numerosas cámaras de seguridad en cualquier entorno urbano es un hecho. Gracias a la evolución tecnológica se ha logrado que instalar cámaras de captura de video no precise altas inversiones económicas y, por tanto, la mayoría de los bancos, estaciones, aeropuertos, tiendas, etc., incorporan en sus instalaciones algún sistema, más o menos complejo, de seguridad basado en video vigilancia.

Desde el punto de vista tecnológico, los sistemas de vigilancia han evolucionado atravesando diferentes etapas en las últimas décadas. La primera generación de sistemas de vigilancia basada en video emplea señales y transmisión analógicas. En la toma de decisiones, siempre es el operador humano el encargado de realizar todas las tareas de análisis de secuencias de video presentadas en varios monitores situados en una sala de control remota, donde las escenas monitorizadas por las distintas cámaras se multiplexan y presentan en un orden periódico y predefinido.

Adicionalmente, la video vigilancia tradicional precisa gran cantidad de espacio de almacenamiento. Todo lo que captura una cámara de seguridad se graba en cintas que, o bien se sobrescriben periódicamente, o bien se guardan en un archivo de video. Este procedimiento limita la duración de video que puede guardarse y hace que el tiempo necesario para su revisión sea elevado, por ejemplo después de haberse producido un delito –asalto en una tienda o robo de un coche– los investigadores pueden buscar el suceso en el video y ver qué ocurrió, pero ya es tarde para evitarlo. Sería necesario un seguimiento continuo, durante las 24 horas del día, del video generado para alertar al personal de seguridad mientras el delito está en progreso, cuando aún existen opciones de evitarlo. (Lipton, May, 2000)

Los sistemas analógicos emplean amplificadores de baja potencia y cables coaxiales para ver de forma remota las imágenes de las cámaras de seguridad. Las cámaras y los monitores están interconectados mediante una red de conmutación y distribución de video, compleja y cara, que direcciona las imágenes de cada cámara hacia un monitor. Esta tecnología tiene demasiadas limitaciones y defectos. El video

analógico sólo se puede distribuir en una red local de cables coaxiales. El costo de enviar una señal analógica a un emplazamiento remoto es prohibitivo.

La segunda generación de sistemas de vigilancia se basa, principalmente, en métodos de procesamiento y comunicación híbridos analógico-digitales, o completamente digitales. Aprovechan la flexibilidad ofrecida por los primeros algoritmos de procesado de video que permiten centrar la atención del operador humano en un grupo de situaciones de interés. Además, de las facilidades proporcionadas por los primeros métodos de compresión digital para aprovechar el ancho de banda de transmisión.

Actualmente, se está produciendo una migración de los sistemas de video clásicos a los sistemas de tercera generación (C.S. Regazzoni, 2001). Los sistemas de tercera generación aprovecharán el progreso de las redes de ordenadores de bajo coste y alto rendimiento, y las comunicaciones multimedia fijas y móviles. La investigación en este campo trabaja en técnicas distribuidas de procesamiento de video en tiempo real para conseguir sistemas robustos de transmisión de imagen, procesamiento de imagen en color, generación de alarmas basada en eventos. Así como, la reconstrucción de secuencias a partir de modelos, segmentación y análisis en tiempo real de secuencias de imágenes 2D, identificación y seguimiento de múltiples objetos en escenas complejas, reconocimiento de comportamientos humanos, etc.

Se prevé que todos estos trabajos proporcionen a las aplicaciones de vigilancia resultados cada vez más interesantes, gracias a la disponibilidad de una potencia de cálculo muy elevada a unos precios aceptables. El desarrollo de las redes de acceso de banda ancha va a permitir a clientes residenciales utilizar estos sistemas para diferentes aplicaciones. Sin embargo, los sistemas de vigilancia presentan requerimientos específicos que implican una necesidad de investigación dedicada y el desarrollo de nuevas herramientas.

Los sistemas de video inteligentes se desarrollan mediante la implementación de **video sensores**. Un **video sensor** no es más que una herramienta de análisis de video digital que ofrece información significativa proveniente de una secuencia de video. El desarrollo de video sensores tiene su base en el procesamiento digital de la imagen.

En la facultad 7, específicamente en el Grupo de Procesamiento de Imágenes y Señales Digitales, se encuentra desarrollando un Sistema de Video Vigilancia que cuenta con los siguientes módulos: gestión de flujos de video y cámaras, detección de movimiento, detección de formas e identificación de chapas de autos y necesita un módulo que se encargue del seguimiento de objeto en secuencias de video.

Un sistema de vigilancia de tercera generación, además de las necesidades más básicas que puede satisfacer, desarrollando diferentes video sensores de mayor o menor complejidad, se pueden ofrecer una serie de servicios de valor añadido en el área de la seguridad, o bien emplear video sensores en el área de la medicina, la industria, el entretenimiento digital, entre otras.

Del análisis de la problemática expuesta surge el planteamiento del siguiente **problema**: ¿Cómo seguir objetos en secuencias de video digital? Con vistas a dar solución al problema planteado, quedan definidos como **objeto de estudio** las técnicas de procesamiento de videos digitales. A raíz de lo cual, el **campo de acción** se enmarca en video sensores para el seguimiento de objetos en secuencias de imágenes animadas. El **objetivo general de la investigación** es desarrollar un video sensor para el seguimiento de objetos en secuencias de video.

En la presente investigación quedaron definidas una serie de **tareas**:

- Realizar un análisis del estado del arte de los algoritmos de seguimiento de objetos en secuencias de videos.
- Realizar un análisis de las características funcionales de los videos sensores.
- Definir el algoritmo de seguimiento de objetos idóneo para la aplicación.
- Implementar el algoritmo de seguimiento de objetos seleccionado.
- Realizar el análisis y diseño de la aplicación (video sensor).
- Implementar el video sensor.

Para el cumplimiento de las tareas antes mencionadas, se emplearon los siguientes **métodos científicos**:

Métodos Teóricos.

Modelación: Este método se considera de gran importancia para la confección del sistema puesto que para su realización se ha utilizado como metodología RUP, con la cual se hace necesaria la creación de varios artefactos en los diferentes modelos que se construyen, que permiten una amplia reproducción de la realidad.

Analítico – Sintético: Este método ha permitido avanzar en la solución del sistema. Se empleó para conocer las teorías y documentos que ocupan el objetivo de investigación, extrayendo los aspectos más importantes relacionados con el objeto de estudio.

Histórico – Lógico: Ayuda a la comprensión de la evolución en el mundo de los sistemas para la búsqueda y detección de textos en videos a lo largo de la historia.

Métodos Empíricos:

Observación: Este método es de suma importancia puesto que permite observar mediante el registro visual lo que ocurre en la situación real que se analiza.

El trabajo consta de cinco capítulos donde se realiza un estudio crítico y descriptivo de las tecnologías seleccionadas, un estudio comparativo para la selección de los métodos óptimos de técnicas de seguimiento de objetos. Se plantea las características del sistema comenzando por una vista de los procesos del negocio existentes; se diseñará el sistema propuesto y luego se implementará cumpliendo así con los principales flujos del ciclo de desarrollo.

En el Capítulo 1, se realiza la fundamentación teórica que justifica la investigación, se analiza el estado actual del tema a tratar a nivel nacional e internacional así como las nuevas tendencias y las tecnologías usadas en la solución, se fundamenta la metodología usada y se abordan otros temas de interés.

En el Capítulo 2, se detalla la situación problémica, se especifica con detenimiento el problema que se pretende resolver, que constituye la base para la identificación del objeto de automatización. Se realiza la propuesta inicial del sistema, estableciendo y describiendo el negocio en el que se enmarca el sistema, así como los requisitos funcionales y no funcionales. Por último se detallan los casos de uso del sistema identificados, dando un breve resumen de cada uno.

En el capítulo 3, se definen las clases de análisis y diseño del software de los casos de usos del primer ciclo de desarrollo; así como los diagramas de secuencias cumpliendo con las descripciones extendidas de los casos de uso. Se expone la arquitectura del sistema, favoreciendo el cumplimiento de los requisitos no funcionales. Se describen además las clases entidades y las controladoras del flujo de trabajo análisis y diseño.

En el capítulo 4, se presentan los modelos definidos en RUP como diagrama de componentes, objetivo primordial de la fase implementación, en los cuales se muestra la descripción física de la topología del sistema y la estructura de las unidades de hardware y el software que se ejecuta en cada unidad como la disposición de las partes integrantes de la aplicación y las dependencias entre los distintos módulos de la aplicación.

En el capítulo 5, se explica en detalle los algoritmos empleados para la realización del video sensor de seguimiento de objetos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza un análisis detallado del estado del arte de los distintos sistemas especializados en el seguimiento de objetos en secuencia de videos. Así como, las distintas técnicas de programación que existen a nivel internacional, nacional y en la universidad. De las tendencias, técnicas, tecnologías, metodologías relacionadas con dichas técnicas y plataformas de desarrollo que la soportan.

1.1 *Conceptos básicos relacionados con el dominio del problema.*

Seguimiento de video: Es el proceso de localizar y seguir un objeto en movimiento (o varios más) en el tiempo utilizando una cámara o un flujo de video.

Video sensor: no es más que una herramienta de análisis de video digital que ofrece información significativa proveniente de una secuencia de video. Vale señalar que el desarrollo de video sensores tiene su base en el procesamiento digital de la imagen.

1.2 Sistemas automatizados existentes vinculados al campo de acción.

1.2.1 SwisTrack

SwisTrack es una poderosa herramienta para el seguimiento de robots, humanos, animales y objetos usando una cámara o un video grabado como entrada. Esta usa la librería OpenCV de Intel para un rápido procesamiento de las imágenes y contiene interfaces para USB, FireWire y cámaras GifE, así como para procesar ficheros AVI.

La arquitectura de SwisTrack es muy flexible y por consiguiente permite al usuario seguir objetos en muchas situaciones. Sus componentes están juntos y configurados. Cada componente ejecuta un paso del procesamiento, el cual puede visualizarse en tiempo real. SwisTrack ya viene con una serie de componentes, pero con tareas especializadas y los programadores son libres de implementar sus propios componentes. La información de posición y trayectoria puede ser mandada por vía TCP/IP en formato

NMEA 0183. Tales datos pueden fácilmente se guardadas para pos procesamiento, o usado en tiempo real.

SwisTrack ha sido desarrollado principalmente por Distributed Intelligent Systems and Algorithms Laboratory (DISAL) y LPM Vision Group a EPFL en Lausanne, Suiza.

1.2.2 ArcSoft

Líder mundial desarrollando software para multimedia

ArcSoft Object Tracking Engine automáticamente sigue objetos mientras se toman fotos con un dispositivo de captura digital. Esta avanzada tecnología ayuda a usuarios novatos a capturar sus fotos con una correcta exposición, foco y mejora, especialmente para niños y mascotas.

1.2.3 RoboRealm

Vision for machines

RoboRealm® es una aplicación de software con una poderosa vista robótica, procesamiento de imagen, y tareas de visión de robótica. Usando una interfaz amigable, de apuntar y cliquear, con análisis complejo de imagen, el control del robot se torna fácil.

1.2.4 ImageJ

ImageJ es el programa de procesamiento de imagen más rápido del mundo. Puede aplicar un filtro a una imagen de 2048x2048 en 0.1 segundos. Esto es 40 millones de pixeles por segundos!

ImageJ tiene desarrollado un módulo que hace un seguimiento de objetos con las siguientes características:

Object Tracker

Autor:	Wayne Rasband (wsr at nih.gov)
Fecha:	21/6/2000
Requerimientos:	ImageJ 1.17y o mayor
Limitaciones:	Solo sigue 2 objetos.
Código:	Tracker_.java
Instalación:	Copiar el Tracker_.class de la carpeta plugins y reiniciar ImageJ.
Descripción:	Este plugin usa el analizador de partículas para seguir uno o dos objetos a través de una pila. Para cada frame, este muestra el numero de objetos, las coordenadas de los primeros dos objetos, y la distancia entre el primero y el Segundo objeto.

1.3 Principales herramientas y tecnologías utilizadas.

Para alcanzar un nivel técnico en el desarrollo de un software, acorde con el desarrollo actual en la automatización de la información, se hace indispensable el estudio detallado de las tecnologías a utilizar, precisando de cada una, las ventajas y desventajas que representan para alcanzar y mantener el nivel técnico esperado. A continuación, se describe las principales tecnologías, conceptos, herramientas y propuestas para el desarrollo de la solución tratada en el presente sistema.

1.3.1 Lenguaje de Programación

Visual C#.NET 2.0

Se seleccionó el lenguaje C# como lenguaje de programación pues es puramente orientado a objetos, posee una constante actualización, sobre el cual se pueden implementar diversas tecnologías muy modernas y utilizadas en la actualidad.

El mismo funciona bajo un entorno de recolección automática de la memoria, lo que aumenta la productividad del desarrollador, brinda un marco de ejecución administrada que permite la optimización y

la ejecución segura del código. Se destaca, además, la existencia de tipos genéricos, indexadores, delegados, eventos, clases parciales, estructuras e interfaces, entre otros rasgos.
La principal desventaja de utilizar este lenguaje es que las plataformas de desarrollo en las que se lleva a cabo su utilización son privadas. Problema que se puede evitar adoptando una estrategia de migración hacia la plataforma libre Mono.

1.3.2 Lenguaje de Modelado

UML 2.1

UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Cabe destacar que UML es un lenguaje para especificar y no un método.
Es utilizado para definir los componentes y artefactos en un sistema. Además, concentra las mejores prácticas de la comunidad y ha sido adoptado masivamente por la industria. En específico el UML 2.1, que es el que se utilizará en la realización de este sistema, posee mejoras en cuanto a la variedad de diagramas, así como la disposición de nuevos artilugios en los diagramas ya existentes en UML 1.x.

1.3.3 Metodología de Desarrollo

Proceso Unificado de Desarrollo (RUP)

RUP constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Además brinda una forma disciplinada para la asignación de las tareas y responsabilidades (quién hace qué, cuándo y cómo). Sus principales características son:

- Iterativo e incremental: A medida que avanza el proceso de desarrollo se producen versiones incrementales, las cuales se acercan cada vez más al producto terminado.

- Guiado por los casos de uso: Los casos de uso son los que indican como debe actuar el sistema con el usuario final o con otro sistema para conseguir su objetivo.

- Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño lo cual constituye la arquitectura del producto a desarrollar.

Es utilizado en la presente tesis por su organización, la misma está compuesta por cuatro fases que son: Inicio, Elaboración, Construcción y Transición, cada una de ellas compuesta de una o varias iteraciones. A su vez está compuesto por nueve Flujos de Trabajo de Ingeniería: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Pruebas y Despliegue, Administración de cambio y configuración, Administración de proyecto y Entorno. En cada una de sus fases se emplean todos los flujos de trabajo pero con diferente énfasis, aunque con el mismo objetivo, obtener artefactos de calidad óptima realizando un esfuerzo mínimo.

1.3.4 Herramienta CASE

Rational Rose

Esta herramienta de diseño guiado por computadora (CASE, por sus siglas en inglés), con la cual se modelarán los flujos de trabajo de la metodología seleccionada. Esta aplicación se encuentra integrada perfectamente a la plataforma .NET y permite hacer uso de técnicas de ingeniería inversa así como de generación automática de documentación. Rational Rose proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente.

1.3.5 Plataforma de Desarrollo

Microsoft Visual Studio 2008

Visual Studio .NET es un conjunto completo de herramientas de desarrollo para la construcción de aplicaciones Web ASP, servicios Web XML, aplicaciones para escritorio, archivos DLL, aplicaciones de consola y aplicaciones móviles.

Ofrece algunas características exclusivas de alta productividad, tales como: diseñadores visuales de Web Forms y Windows Forms, esquemas XML y datos. Un depurador de varios lenguajes que alterna sin problemas entre códigos escritos en lenguajes diferentes. Proporciona una estrecha integración con .NET Framework; así como una ayuda dinámica, que proporciona completamiento contextual continuo mientras se escribe; muestra una lista de tareas, los errores del compilador y las tareas pendientes. Presenta características de diseño de arquitecturas como la integración con Visio y un explorador de servidores para obtener acceso visual a bases de datos, servicios de Windows, contadores de rendimiento y componentes de aplicaciones del lado del servidor.

Visual Studio es actualmente el IDE8 por excelencia para la mayoría de los desarrolladores, plataforma solo comparada en algunos aspectos con el NetBeans y el Eclipse ambos para el lenguaje Java, los antes mencionados no cumplen con todas las características vistas. Todas estas particularidades hacen del Visual Studio la mejor y más práctica herramienta para desarrollar la aplicación.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se detalla la situación problémica, se especifica con detenimiento el problema que se pretende resolver, que constituye la base para la identificación del objeto de automatización. Se realiza la propuesta inicial del sistema, estableciendo y describiendo el negocio en el que se enmarca el sistema, así como los requisitos funcionales y no funcionales. Por último se detallan los casos de uso del sistema identificados, dando un breve resumen de cada uno.

2.1 Problema y situación problémica

Históricamente, las tareas de interpretación de video de alto nivel relacionadas con la vigilancia son llevadas a cabo, en su totalidad, por un operario que tiene que procesar una gran cantidad de información de video que se le muestra en uno o varios monitores.

Una vez detectada una situación de alarma, en la configuración tradicional de un sistema de seguridad, el video grabado debe ser analizado para reconstruir el evento y analizar las causas que lo produjeron. Si la única forma de acceder a la causa de la alarma consiste en visualizar secuencialmente el video grabado, esta operación resulta extremadamente costosa en tiempo y, en consecuencia, en recursos económicos.

A su vez, es bien sabido, que el operario vigila varios escenarios simultáneamente y realiza este trabajo durante varias horas seguidas que pueden llevar a que su atención disminuya y, por tanto, la probabilidad de pasar por alto situaciones peligrosas aumente.

Los problemas de capacidad de almacenamiento, se pueden resolver grabando sólo las partes importantes del video capturado. En aplicaciones de seguridad, esto significaría grabar sólo cuando existe movimiento en la escena o cuando se genera una alarma determinada que previamente fue configurada.

Por tanto, para conseguir sistemas de seguridad más robustos basados en video vigilancia, el operario humano necesita la ayuda de herramientas automáticas de procesado, sistemas capaces de llamar su atención cuando se produce una situación peligrosa o extraña en el entorno que vigila y que permitan recuperar fácilmente la parte de la secuencia de video relativa al evento que representa la razón por la que se le ha alertado.

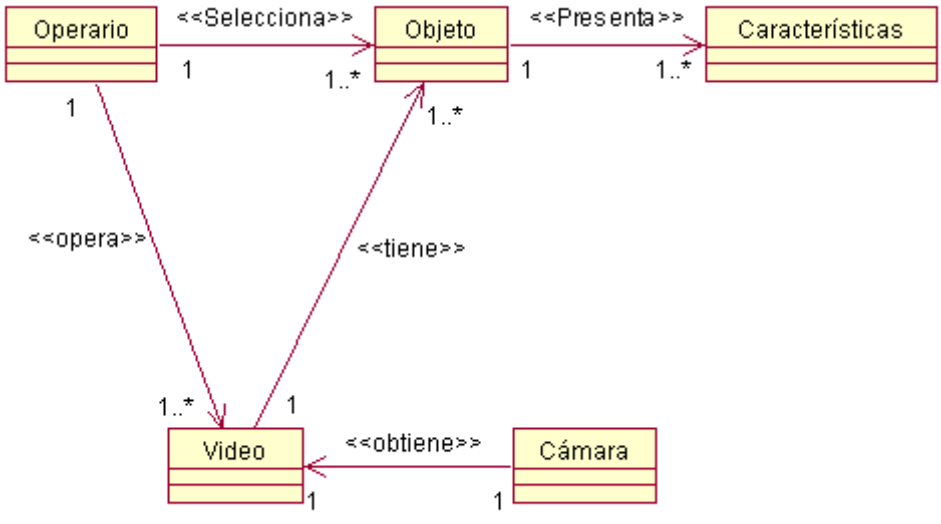
2.2 Objeto de Automatización

Así, se establecen como proceso a automatizar, la localización y seguimiento de objetos en movimiento en el tiempo utilizando una cámara o un flujo de video.

2.2.1 Modelo de Dominio

En el flujo de trabajo actual los procesos no complementan la realización de un modelo de negocio, por lo que se ha decidido crear un modelo de dominio donde se captura y expresa el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema de software.

Figura 1. Modelo de Dominio



Descripción del modelo de dominio: El operario es la persona encargada de seleccionar uno o varios objetos, que pueden ser, una persona, un automóvil, un barco, etc. Este objeto es mostrado en un video obtenido por una cámara de captura. Este objeto presenta varias características que lo hace diferente a los otros objetos, para su posterior seguimiento.

2.3 Especificación de los requisitos de software

Un requisito de software es una condición que debe cumplir un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta. Estos se clasifican en dos grupos fundamentalmente: los requisitos funcionales –comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica– y los requisitos no funcionales –propiedades o cualidades que el producto debe tener.

2.3.1 Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

RF1. Seguir el objeto seleccionado.

2.3.2 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

1. Rendimiento

1.1 Se debe procesar los frames proporcionados y dar seguimiento al objeto en el menor tiempo posible.

2. Portabilidad

2.1 El componente debe poder ser utilizado sobre Windows.

3. Hardware

3.1 Memoria RAM 512 MB

2.4 Definición de los casos de uso

La arquitectura del sistema está condicionada por los casos de uso –los cuales representan la concreción de los requisitos funcionales a la hora de implementar una solución software. Estos proporcionan un medio para que los desarrolladores, usuarios finales y trabajadores lleguen a una comprensión común del sistema propuesto. Son empleados además para la validación de la arquitectura a lo largo del desarrollo.

2.4.1 Definición de los Actores

Actores del sistema	Justificación
	Es el que interactúa con los componentes. Es el que inicializa los casos de usos

2.4.2. Listado de los casos de uso

CU-1	Seguir objeto
Actor	Sistema Tracker
Descripción	El caso de uso se inicia cuando el Sistema Tracker, hace la petición de que se siga un objeto dado previamente.
Referencia	RF1

2.4.3. Diagrama de casos de uso del Sistema

A continuación la figura 2 muestra el diagrama de casos de uso del sistema propuesto. El actor Sistema Tracker es el que se encarga de inicializar el caso de uso Seguir Objeto, y mediante este proceso el sistema comienza a trabajar.

Figura 2. Diagrama de Casos de Uso del Sistema.

2.4.4 Descripción extendida de los Casos de Uso

Nombre del Caso de Uso:	Seguir objeto
Actores:	Sistema Tracker
Propósito:	Dar seguimiento a un objeto.
Resumen:	El sistema recibe el objeto que se desea seguir, graba sus características y devuelve sus posición cada vez que se mueva.
Referencia:	RF1
Tipo:	Crítico
Precondiciones:	Se debe proporcionar el objeto que se desea seguir.
Curso normal de los eventos	
Acciones de los actores	Respuesta del Sistema
1 – Se provee el área y el frame en la cual se encuentra objeto que se desea seguir.	2 - Se extrae las características del objeto.
4 – Se pasan los frames posteriores para seguir el objeto.	5 – Se da la posición del objeto en cuestión.
Postcondiciones:	Se muestran las ocurrencias del proceso.

En el presente capítulo se han analizado los procesos de negocio que serán objeto de automatización. Se describió el sistema propuesto, exponiendo el modelo de dominio del mismo, los requerimientos funcionales y no funcionales, mostrándose además los diagramas de casos de uso que se han modelado. Además, se realizó una descripción detallada de cada caso de uso.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En el capítulo, se definen las clases de análisis y diseño del software de los casos de usos del primer ciclo de desarrollo; así como los diagramas de secuencias cumpliendo con las descripciones extendidas de los casos de uso. Se expone la arquitectura del sistema, favoreciendo el cumplimiento de los requisitos no funcionales. Se describen además las clases entidades y las controladoras del flujo de trabajo análisis y diseño.

3.1 Modelo arquitectónico.

La necesidad de establecer un marco de trabajo estructural básico, o sea, la estructura general del software, ofreciendo una integridad conceptual al sistema, facilitando la comunicación entre los diferentes participantes en el desarrollo y aportando una visión de estructura e interacción de los componentes del sistema; es lo que da vital importancia a la definición de un modelo arquitectónico.

El proyecto desarrollado, es un sistema con una arquitectura centrada en los flujos de datos, basada en el patrón “pipe and filter” (tuberías y filtros); donde los componentes presentes en el sistema responden a decisiones arquitectónicas para hacer cumplir requerimientos funcionales y no funcionales. Estos componentes están asociados a flujos de datos y refinamientos sucesivos. O sea, un conjunto de elementos denominados “filtros” conectados entre si por “tuberías” trasmiten datos desde un componente al siguiente.

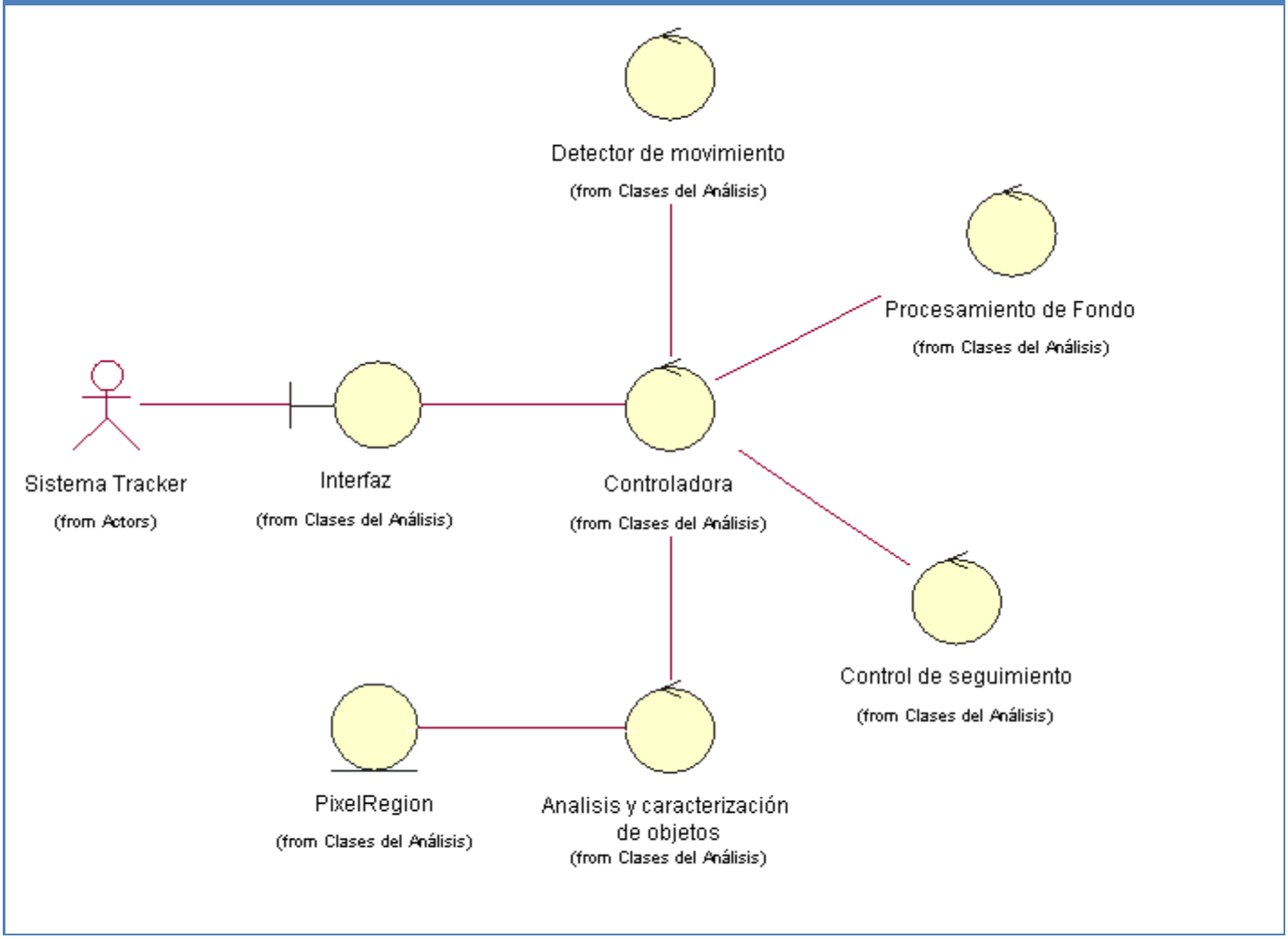
Cada filtro trabaja de manera independiente de los componentes. Se diseñan de tal modo que esperan un conjunto de datos en un determinado formato y obtiene como resultado otros datos de salida en un formato específico. Este patrón de arquitectura es particularmente efectivo a la hora de descomponer el problema en pasos independientes, reutilizar de filtros, facilitar el mantenimiento, independencia entre filtros y ejecución concurrente de filtros.

3.2 Modelo Análisis

Con el propósito de conseguir una comprensión más precisa y una descripción más detallada del sistema, se refinan y estructuran los requisitos obtenidos con anterioridad. Se han realizado los diagramas de clases de análisis, los cuales muestran qué clases participan en las realizaciones de los distintos casos de usos del primer ciclo de desarrollo planteados en capítulo anterior. Estos diagramas se muestran a continuación.

3.2.1 Diagrama de Clases del Análisis

Figura 3. CU-1 Seguir Objeto



En esta figura el actor Sistema Tracker se comunica, a través, una clase Interfaz, con la clase Controladora. La que se encarga de manipular todas las operaciones del sistema, comunicándose para esto, con otras clases controladoras, las cuales son: Detector de Movimiento, Procesamiento de Fondo, Control de seguimiento y Análisis y caracterización de objetos que esta a su vez utiliza una clases Entidad para guardar datos de interés.

3.3 Modelo Diseño

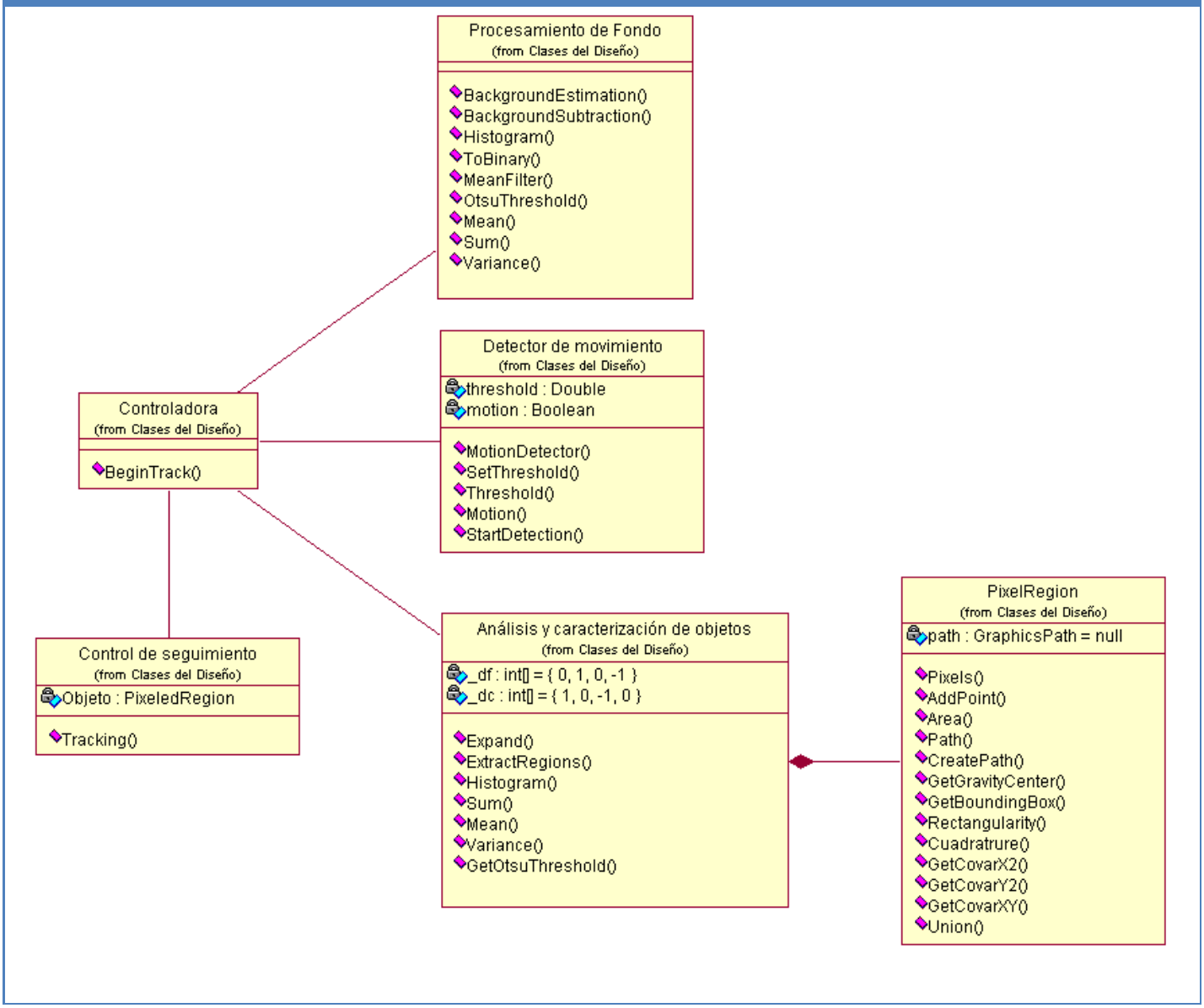
Para la creación de una arquitectura estable y sólida, y de un plano del modelo de implementación; durante esta fase se analiza a profundidad si es posible dar una solución que satisfaga a los requerimientos significativos. En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Básicamente es adaptar el diseño para que se corresponda con el entorno de implementación, diseñando sus funcionalidades.

En esta etapa se generan como principales artefactos los diagramas de clases de diseño, así como los diagramas de interacción, ya sean de secuencia o de colaboración; los cuales se utilizan para modelar los aspectos dinámicos de un sistema utilizando un conjunto de objetos y sus relaciones e incluyendo los mensajes que se pueden enviar entre ellos.

Diagrama de clases del diseño

3.3.1 Diagrama de clases del diseño Seguir objeto

Figura 4. Diagrama de clases de diseño del caso de uso Seguir objeto

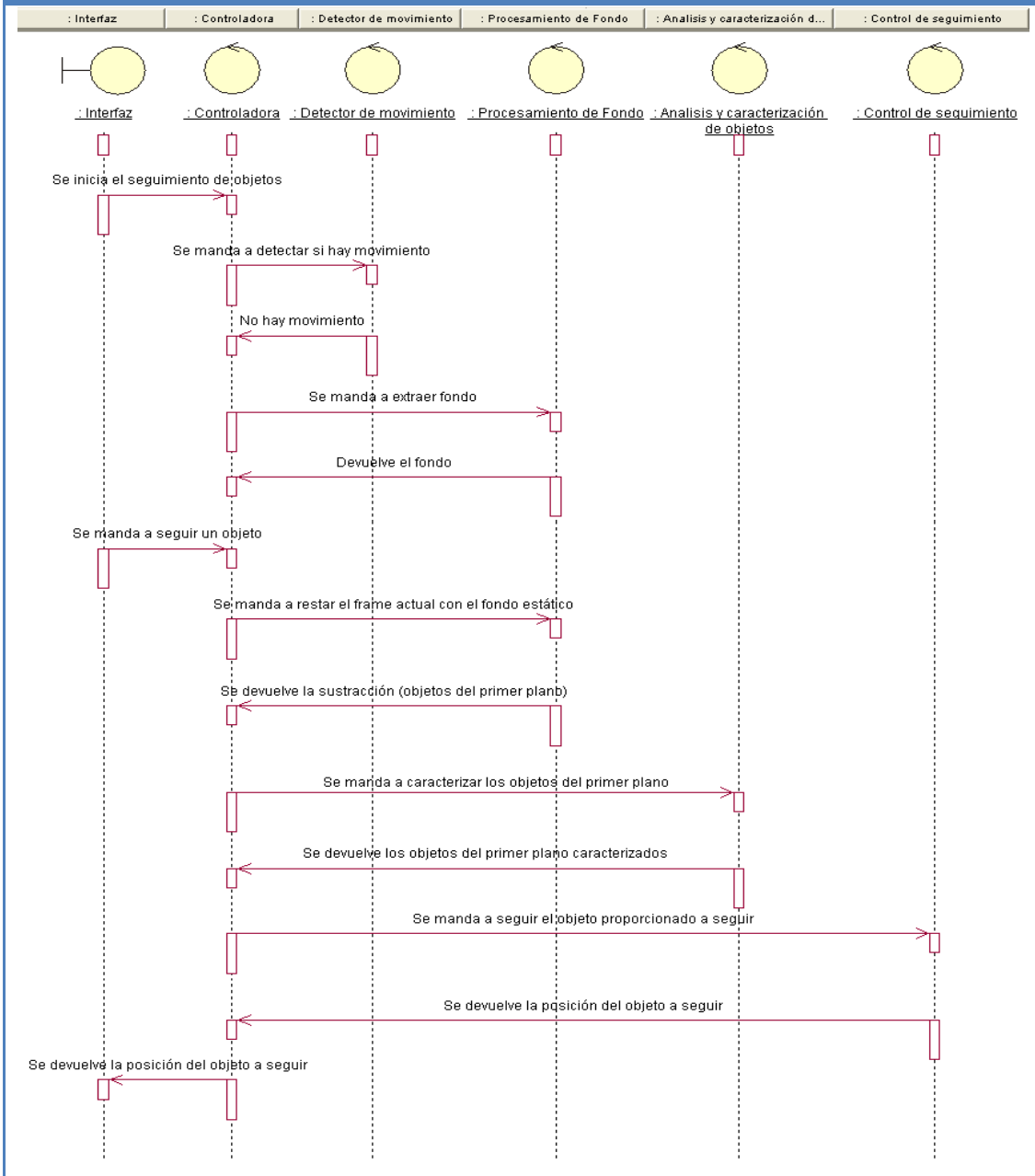


La anterior figura describe las relaciones entre las clases que se definieron en el modelo de análisis pero ya enfocado a un lenguaje de programación, con sus correspondientes variables y métodos.

Diagramas de Secuencia

3.3.2 Diagrama de Secuencia Seguir objeto

Figura 5. Diagrama de Secuencia del Caso de Uso Seguir objeto



El diagrama de secuencia anterior del caso de uso seguir objeto muestra la interacción de un conjunto de estos en una aplicación a través del tiempo. El diagrama inicia cuando en la clase interfaz manda un mensaje de seguir un objeto a la clase controladora. Se detecta si hay movimiento en la escena, si no hay movimiento se envía otro mensaje de la clase Controladora a la clase Detector de Movimiento de mandar a extraer el fondo y a su vez se devuelve este fondo.

Cuando es extraído el fondo, el sistema está listo para seguir un objeto, para hacer esto se manda de la clase controladora a la Clase Procesamiento de Fondo un mensaje de restar el fondo estático al fondo actual. Esta devuelve la sustracción con los objetos en primer plano, la clase controladora manda otro mensaje a la controladora Análisis y Caracterización de objetos para que los analice. Esta manda un mensaje que los objetos ya han sido caracterizados, se manda el mensaje de seguir el objeto, de la controladora a la clase Control de Seguimiento y devuelve esta un mensaje de la posición del objeto a seguir.

3.4 Descripción de las clases

3.4.1 Clase Controladora

Nombre	Controladora
Tipo clase	Controladora
Para cada responsabilidad	
Nombre	`void BeginTrack(Point[] ptos)`
Descripción	Comienza el Seguimiento de Objetos.

3.4.2 Clase Procesamiento de Fondo

Nombre	Procesamiento de Fondo
Tipo clase	Controladora
Para cada responsabilidad	
Nombre	`static unsafe Bitmap BackgroundEstimation(Bitmap[] bmptr)`

Descripción	Hace una estimación del fondo dado un arreglo de frames.
Nombre	`static unsafe Bitmap BackgroundSubtraction(Bitmap back, Bitmap frame)`
Descripción	Resta dos imágenes.
Nombre	`static unsafe int[]  Histogram`
Descripción	Calcula el histograma de una imagen.
Nombre	`static unsafe Bitmap ToBinary(Bitmap subt)`
Descripción	Binariza una imagen, utilizando un umbral, calculado por el método de Otsu.
Nombre	`static unsafe Bitmap MeanFilter(Bitmap subt)`
Descripción	Se le aplica un filtro a la imagen.
Nombre	`static int OtsuThreshold(Bitmap bmp)`
Descripción	Se calcula un umbral de la imagen por el método Otsu.
Nombre	`static float Mean(float[] hist, int pos)`
Descripción	Calculo utilizado para calcular el umbral por el método Otsu.
Nombre	`static float Sum(float[] hist, int pos)`
Descripción	Realiza la suma de un arreglo.
Nombre	`static float Variance(float[] hist, int k, float gmean, float gvarian)`
Descripción	Calcula la varianza utilizada en el método de Otsu.

3.4.1 Clase Detector de Movimiento

Nombre	**Detector de Movimiento**	
Tipo clase	Controladora	
Atributo		Tipo
`threshold`		`double`
`motion`		`bool`
Para cada responsabilidad		
Nombre	`unsafe void SetThreshold(Bitmap bmp1, Bitmap bmp2)`	
Descripción	Cambia el umbral.	

Nombre	`double Threshold()`
Descripción	Devuelve el umbral.
Nombre	`bool Motion()`
Descripción	Dice si hay movimiento o no.
Nombre	`unsafe void StartDetection(Bitmap bmp1, Bitmap bmp2)`
Descripción	Inicia el proceso de detectar movimiento.

3.4.1 Clase Análisis y caracterización de objetos

Nombre	Análisis y caracterización de objetos	
Tipo clase	Controladora	
Atributo		Tipo
`_df`		`static int[]`
`_dc`		`static int[]`
Para cada responsabilidad		
Nombre	`static void Expand(ref PixeledRegion reg, int x, int y, BitmapData odata)`	
Descripción	Método que detecta un objeto en la imagen.	
Nombre	`static List<PixeledRegion> ExtractRegions(Bitmap bmp)`	
Descripción	Método que cada vez que detecta un posible objeto llama al `Expand` para identificar este posible objeto.	
Nombre	`static unsafe int[]  Histogram`	
Descripción	Calcula el histograma de una imagen.	
Nombre	`static float Mean(float[] hist, int pos)`	
Descripción	Calculo utilizado para calcular el umbral por el método Otsu.	
Nombre	`static float Sum(float[] hist, int pos)`	
Descripción	Realiza la suma de un arreglo.	
Nombre	`static float Variance(float[] hist, int k, float gmean, float gvarian)`	
Descripción	Calcula la varianza utilizada en el método de Otsu.	
Nombre	`static int OtsuThreshold(Bitmap bmp)`	

Descripción	Se calcula un umbral de la imagen por el método Otsu.

3.4.1 Clase PixelRegion

Nombre	PixelRegion	
Tipo clase	Entidad	
Atributo		Tipo
path		GraphicsPath
Para cada responsabilidad		
Nombre	`List<Point> Pixels()`	
Descripción	Devuelve una lista de puntos.	
Nombre	`void AddPoint(Point p)`	
Descripción	Adiciona un punto a la lista.	
Nombre	`int Area()`	
Descripción	Obtiene la cantidad de pixeles de el objeto.	
Nombre	`GraphicsPath Path()`	
Descripción	Devuelve el rectángulo que enmarca el objeto.	
Nombre	`void CreatePath()`	
Descripción	Crea el rectángulo que enmarca el objeto.	
Nombre	`Point GetGravityCenter()`	
Descripción	Devuelve el centro de gravedad.	
Nombre	`Rectangle GetBoundingBox()`	
Descripción	Devuelve el BoundingBox del objeto.	
Nombre	`float Rectangularity`	
Descripción	Devuelve la rectangularidad del objeto.	
Nombre	`float Cuadratrure()`	
Descripción	Devuelve el Cuadratura del objeto.	
Nombre	`GetCovarX2()`	

Descripción	Devuelve el Covar de las X del objeto.
Nombre	`GetCovarY2()`
Descripción	Devuelve el Covar de las Y del objeto.
Nombre	`float GetCovarXY()`
Descripción	Devuelve el Covar de las XY del objeto.
Nombre	`void Union(PixeledRegion b)`
Descripción	Une en la lista dos objetos en 1.

3.4.1 Clase Control de Seguimiento

Nombre	Control de Seguimiento
Tipo clase	Controladora
Para cada responsabilidad	
Nombre	`static unsafe Point[] Tracking(PixeledRegion objecto)`
Descripción	Dado el objeto que se desea seguir, da la posición de este en el transcurso del tiempo.

En este capítulo se mostraron los diagramas de clases de análisis y diseño, así como los diagramas de secuencia de cada realización de los casos de uso. Además se han mostrado las relaciones existentes entre las clases entidades y las controladoras y se describieron cada una de ellas.

CAPÍTULO 4: IMPLEMENTACIÓN

A continuación se presentan los modelos definidos en RUP como diagrama de componentes, objetivo primordial de la fase implementación. En los cuales se muestra la descripción física de la topología del sistema, la estructura de las unidades de hardware y el software que se ejecuta en cada unidad. Así como la disposición de las partes integrantes de la aplicación y las dependencias entre los distintos módulos de la aplicación.

4.1 Descripción de los componentes

El sistema cuenta con cuatro ensamblados: BackgroundProcessor.dll, MotionDetector.dll, ObjectDetector.dll y Tracking.dll; que se encargan de distribuir las funcionalidades necesarias para la realización del algoritmo propuesto.

El componente BackgroundProcessor.dll tendrá como objetivo separar el fondo, o background, del primer plano, o foreground, realizar funciones sobre este como binarizarlo y aplicarle filtro y restar este background de otra imagen para obtener los objetos del primer plano.

El componente MotionDetector.dll tiene como propósito, detectar cambio en la escena, es decir, si hubo movimiento en la escena o no.

El ObjectDetector.dll tiene como objetivo definir e identificar regiones que el sistema considerará objetos de primer plano y de estas regiones extraer información que permita caracterizar el objeto al que hacen referencia tales como la cantidad de puntos que conforman el objeto, las coordenadas máximas y mínimas de este objeto, el centro de gravedad de estos objetos, su cuadratura, etc.

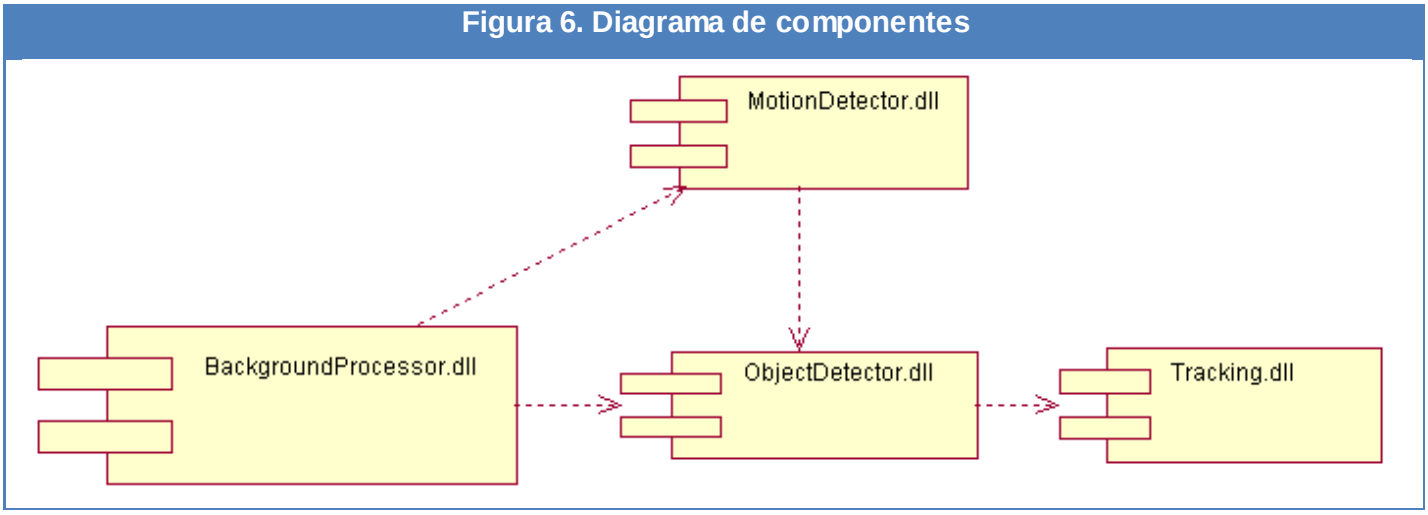
Y finalmente el Tracking.dll, tendrá como objetivo el seguimiento de un objeto durante un periodo de tiempo.

4.2 Diagrama de Componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de

desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes.
Los componentes se encuentran repartidos según plantea la arquitectura, descrita en el capítulo anterior.

Figura 6. Diagrama de componentes



El diagrama de componente se muestra en la figura 6. Las flechas indican el flujo de datos. Estos componentes pueden funcionar de forma independiente o ser una parte de un sistema complejo formado por la integración de los diferentes algoritmos.

En este capítulo se desarrolló la fase implementación del sistema verificando que su comportamiento externo satisfaga los requisitos establecidos por los clientes y futuros usuarios del mismo. Todo esto a través de dos formas de representación como la del diagrama de componentes que mostró la organización y la dependencia entre un conjunto de componentes como es la vista estática del sistema. Además, el diagrama de despliegue que muestra los complementos del diagrama de componentes y que unidos proveen la vista de implementación del sistema.

CAPÍTULO 5: ALGORITMOS

En este capítulo se explican en detalle los algoritmos empleados para la realización del video sensor de seguimiento de objetos.

5.1 Algoritmo de detección de movimiento

El detector de movimiento constituye en sí mismo un video sensor, su propósito es detectar cambio en la escena. Por ser muchas las aplicaciones en las que el video estático carece de interés, el detector de movimiento es una herramienta útil para generar un evento que por ejemplo, inicie la grabación de video o haga saltar una alarma, cuando el video que se captura contiene información relevante.

La implementación que en este punto se va a detallar, tiene como resultado un detector de movimiento robusto frente a variaciones del nivel de ruido. Es conveniente señalar, antes de comenzar el estudio, que con esta técnica no se excluyen los cambios debidos a variaciones bruscas de iluminación en la escena, es decir, se detectarán tanto éstos como los cambios debidos a movimiento real. En la literatura [(T.Ebrahimi, October 2001), (Julio Pons, April 2002)] se encuentran algunas propuestas que son parcialmente capaces de ignorar los cambios debidos a la iluminación. En general, son la respuesta a una necesidad específica. En este proyecto, el interés se centra en la presencia o no de cambio en la escena.

A continuación se presenta la implementación del sensor dividido en tres bloques:

1. Cálculo del módulo de la diferencia de dos imágenes consecutivas y conversión a gris.

2. Estimación de la varianza de la señal diferencia

3. Control de cese de movimiento

Módulo 1

Del primer bloque, módulo de la diferencia de dos imágenes consecutivas y conversor a gris, resta hacer varias puntualizaciones:

- Se procesa la imagen entera a nivel de píxel.

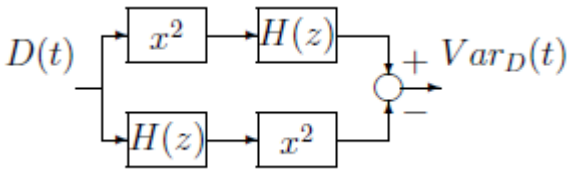
- La resta se realiza para cada componente de color RGB a pesar de que se quiere que la salida D(t) tenga un único valor de diferencia por píxel, que se consigue haciendo la posterior conversión a gris. Esto es así, debido a que si la resta se realiza directamente con imágenes de grises, niveles de luminancia, la medida de la diferencia es menos precisa. Por ejemplo, un píxel rojo (255,0,0) tiene una luminancia $Y = 0.3 * 255 \approx 76$ y un píxel cuyo color predominante sea el verde (0,92,200) tiene una luminancia $Y = 0.59 * 92 + 0.11 * 200 \approx 76$, con lo que la diferencia de luminancia sería prácticamente nula, mientras que si la resta se hace en color, el valor diferencia resultante daría 99.

Módulo 2

En el segundo bloque se realiza la estima de la varianza, VarD(t).
Para el cálculo de la varianza local de cada elemento o ROI de la imagen, que se llamará d(t), se parte de la ecuación:

$$\mathrm{Var_D(t)} = \mathrm{E\{|d(t)^2|\}} - |\mathrm{E\{d(t)\}}|^2$$

Siendo E {·} el operador esperanza matemática. Como se muestra en la figura, se consigue con dos módulos de cuadrado y dos filtros H(z) que realizan un promedio recursivo de la señal en el tiempo.



El filtro H(z) es:

$$\mathrm{H(z)} = \frac{1-\alpha}{1-\alpha\, \mathbf{z}^{-1}}$$

Y tiene un único parámetro, α

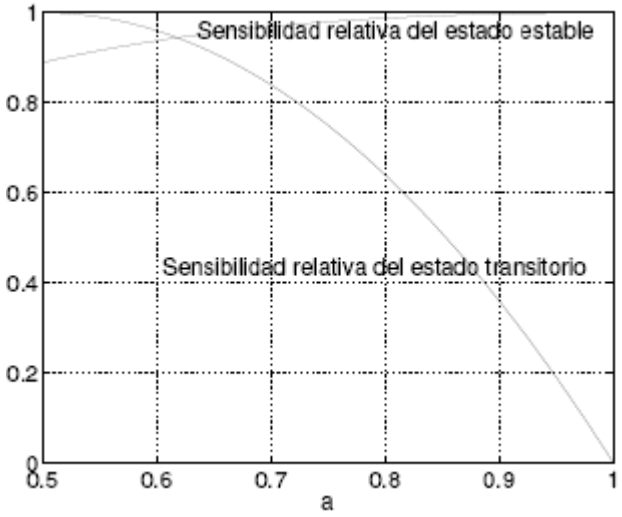
Finalmente, se establece un umbral de decisión sobre VarD(t) para determinar la presencia de movimiento.
La justificación de la elección del parámetro $\propto$ del filtro se detalla a continuación:

Parámetro $\propto$
Con el parámetro del filtro, se tiene dos objetivos:

- Debe proporcionar una respuesta rápida.
- Debe tener una alta sensibilidad.

Los resultados indican que se debe elegir un valor de compromiso entre 0.5 y 1 para cumplir ambos objetivos. Se puede determinar la variación con $\propto$ de ambos criterios con respecto a su correspondiente valor máximo. Esta variación se observa en la figura 7

Figura 7. Variación de $\propto$



El punto de cruce de ambos criterios se corresponde con el valor:

$$\propto_{opt} = \frac{\sqrt{5}-1}{2} = 0{,}618$$

Con la elección de este valor de $\propto$, el valor de cualquiera de los dos criterios, está al 94.4% del máximo.

Módulo 3

El tercer y último bloque, controla que el detector funcione adecuadamente tanto en detección de presencia de movimiento como en detección de ausencia de movimiento.
El motivo de incorporar este tercer módulo es que enventana la respuesta del filtro. La ventana se crea comprobando que en D(t) existen valores relativamente altos, indicativos de la existencia de movimiento.

5.2 Algoritmo de estimación y sustracción de fondo

Uno de los primeros pasos en el seguimiento de objetos en secuencias de vídeo consiste en separar el fondo, o background, del primer plano, o foreground. Conceptualmente, se distingue fondo de primer plano definiendo el fondo como la parte estacionaria de la escena, mientras el primer plano estaría formado por los objetos en movimiento o temporalmente estáticos.

Sin embargo, esta distinción roza lo subjetivo porque: ¿cuál es la diferencia entre un objeto estacionario y uno con movimiento potencial?, ¿cuánto tiempo debe pasar para que lo estático se convierta en estacionario? En la práctica, se necesitará establecer un umbral de tiempo para que, una vez sobrepasado, el objeto sin movimiento comience a considerarse parte del fondo.

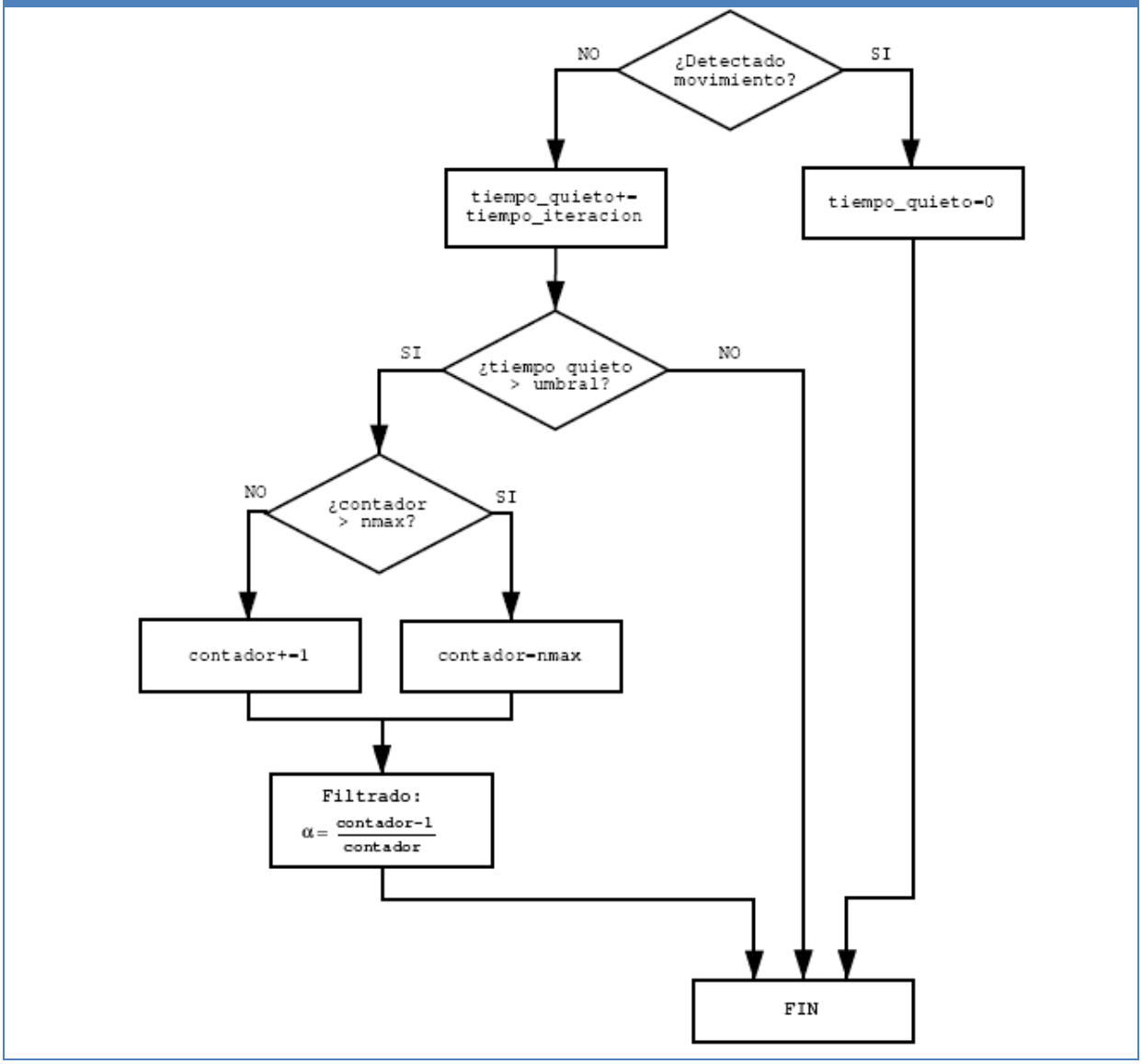
Para separar objetos y fondo, habitualmente, se lleva a cabo una sustracción de fondo, calculando la diferencia entre los pixels del fondo, que deben ignorarse, y los pixels de la imagen capturada. Mantener una representación precisa del fondo es importante para detectar con claridad todos los objetos no estacionarios de la escena, y a esto se le llama estimación de fondo.

Una vez que elementos tales como coches, personas, etc., se identifican como foreground, constituyen la información que puede servir de entrada a módulos de nivel superior que realicen tareas sobre objetos como seguimiento, reconocimiento, clasificación, etc.

El algoritmo de estimación de fondo, está diseñado para que cada píxel entre a formar parte del fondo cuando sus características particulares sean las apropiadas, y en la medida que

éstas determinen. Cada píxel se trata de forma independiente siguiendo el diagrama lógico presentado en la siguiente figura:

Figura 8. Algoritmo de estimación de fondo



La figura muestra que se trata de un algoritmo sencillo. Se precisa:

- La salida del detector de movimiento: Proporciona la información de movimiento y es necesaria para saber cuánto tiempo lleva quieto cada pixel.

- Una variable para guardar la información del tiempo que lleva quieto cada pixel, representada por tiempo quieto.

- Un contador del número de veces que ha sido promediado el pixel: Necesaria para calcular el parámetro «p del filtro que se le aplica, representado por contador.

Sustracción del fondo

El objetivo perseguido con este algoritmo es separar los objetos dinámicos, como personas o coches, de un fondo relativamente estático como paso previo a un procesado de alto nivel. Una vez que se ha conseguido una estimación apropiada del fondo, extraer los objetos de foreground es una técnica sencilla.

Básicamente, por comparación de la imagen capturada con el fondo estimado,

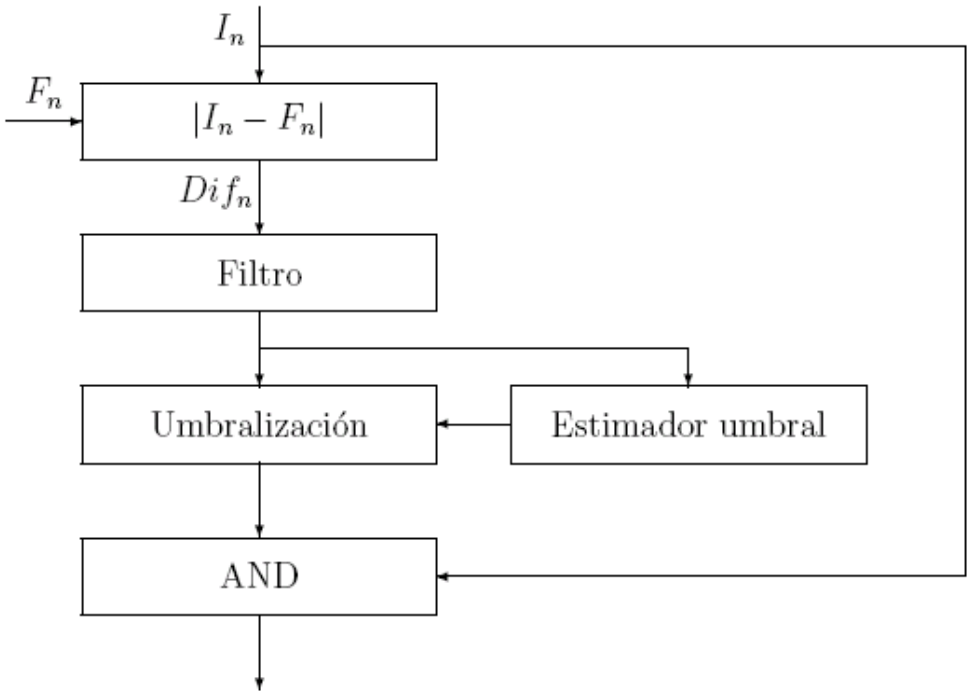
$$Dif_n = |I_n - F_n|$$

y umbralización posterior de la imagen diferencia, se obtiene una imagen con los objetos buscados.
La binarización es realizada por el método de Otsu, el cual es un método de binarización adaptativa. Propone la división de la imagen en dos grupos o clases, el fondo de la imagen y los objetos que se encuentran en esta. Luego para cada umbral candidato se calcula la varianza entre las dos clases por la fórmula.

$$\sigma(T) = n_f(T)n_0(T)[\mu_f(T) - \mu_0(T)]^2$$

Donde $\sigma(T)$ es la varianza entre las clases, $\mu_f(T)$ la media del fondo, $\mu_0(T)$ la media de los objetos, $n_f(T)$ y $n_0(T)$ el número de pixeles de fondo y de objetos respectivamente. El umbral óptimo T es el que maximiza la varianza entre las clases.
El diagrama de bloques de la siguiente figura presenta el proceso de sustracción del fondo.

Figura 9. **Proceso de sustracción del fondo**



Se observa que previamente a la umbralización se realiza un filtrado. Se trata de un filtro de media deslizante utilizado para eliminar el ruido impulsivo que aparece en la imagen diferencia.

El bloque AND con entradas la imagen binaria con la "plantilla" de los objetos y la imagen capturada In, permite tener a la salida una imagen en RGB que únicamente presenta los objetos de foreground sin perder sus características.

Resultados y limitaciones

Los problemas que, idealmente, un sistema de estimación y mantenimiento de fondo (Kentaro Toyama, 1999) debe evitar son:

- **Hora del día**: Los cambios graduales de iluminación alteran la apariencia del fondo.
- **Objetos desplazados**: Un objeto del fondo puede moverse. Cuando un objeto que inicialmente estaba en el fondo se mueve, tanto él como la parte del fondo que antes ocultaba aparecen como cambios.

- **Modificación de luces**: Cambios repentinos de iluminación alteran la apariencia del fondo.
- **Hojas de árbol**: El fondo puede presentar variaciones oscilatorias, como las hojas de un árbol que se mueven con el viento. Requieren modelos que puedan representar grupos disjuntos de valores de píxel.
- **Camuflaje**: Los píxels de un objeto del primer plano con características similares al fondo modelado, pueden quedar camufladas y no aparecer en el foreground.
- **Inicialización**: En algunos entornos no es posible disponer de un periodo de inicialización en ausencia de objetos en el primer plano.
- **Objetos uniformes**: Cuando un objeto de color uniforme se mueve, puede que no se detecten cambios en los píxels interiores. Así, es posible que en el foreground no aparezca el objeto entero.
- **Sombras**: Los objetos del foreground habitualmente proyectan sombras que hacen que el fondo se muestre diferente.

Los sistemas perfectos no existen. En (Kentaro Toyama, 1999) se presenta una técnica más compleja basada en predicciones probabilísticas a nivel de píxel, de región y de imagen, para tratar de minimizar estos problemas. También se encuentran en la literatura métodos basados en la diferencia de frames adyacentes, media y umbral, media y covarianza, mezcla de gaussianas, decisión bayesiana, etc. Todos ellos presentan ciertas limitaciones en la solución de los problemas presentados.

El modelo que se ha utilizado en este proyecto es relativamente sencillo y de bajo coste computacional. A continuación, se van a analizar los resultados que se obtienen con este modelo frente algunos de los problemas y en qué medida pueden afectar a las necesidades para las que se ha implementado este vídeo sensor. El resto de problemas: sombras y objetos uniformes, se considera que son problemas no relacionados directamente con la estimación y mantenimiento de fondo, por lo que más adelante se mostrará el efecto que producen al determinar los objetos del primer plano.

- **Hora del día**: Este vídeo sensor se ha diseñado con la intención de absorber los cambios lentos del entorno. El cambio de iluminación gradual debido a la situación horaria es un ejemplo típico de este tipo de variación lenta.

- **Objetos desplazados**: El hecho de que el hueco que deja un objeto del fondo cuando se desplaza sea tratado como un nuevo objeto, en el caso que afecta a este

proyecto, no es considerado un problema a solucionar, sino más bien una información interesante para analizar. Un ejemplo de aplicación de esta información es la detección de objetos robados/desaparecidos. Un elemento que desaparece del fondo puede ser un objeto robado, mientras que uno que aparece puede haber sido abandonado.

En las aplicaciones en las que no interesa la detección de los objetos que desaparecen del fondo, existe un intervalo de tiempo en que se incluirá en el foreground un objeto que no existe en la realidad. Este intervalo de tiempo termina cuando la zona absorbe el nuevo fondo.

En cualquier caso, como se viene resaltando a lo largo de la memoria, un vídeo sensor no pretende ser una herramienta totalmente autónoma y debe existir un supervisor humano quién, en última instancia, valide la información que ofrece el sensor.

- **Modificación de luces**: En todos los algoritmos de este sistema se requiere ausencia de variaciones abruptas de iluminación. En este algoritmo el resultado que se obtendría sería que la imagen entera se consideraría primer plano.

  Si se produce esta situación, transcurrido el tiempo de adaptación, la referencia de fondo será válida para las nuevas condiciones y el sistema proporcionará, de nuevo, buenos resultados. Los datos obtenidos desde el instante de cambio hasta la adaptación deberán desecharse.

  La limitación que presenta el sistema respecto a la variación brusca de luces se puede minimizar incorporando un proceso que detecte esta situación e inmediatamente después ponga al sistema en fase de inicialización de fondo.

- **Hojas de árbol**: Debido a que el funcionamiento del estimador de fondo depende en gran medida del detector de movimiento, en el caso de movimiento oscilatorio, se pueden distinguir dos situaciones:

  1. Los objetos que se mueven son grandes respecto al tamaño de la ROI del detector de movimiento. Las zonas con movimiento nunca pasan al fondo. Si se producen durante la inicialización, quedarán zonas del fondo sin inicializar.
  2. Los objetos son tan pequeños que el sensor de movimiento no los aprecia. Es el caso habitual de hojas de árboles movidas por el viento. En

este caso, el fondo en estas zonas es un promedio de imágenes que son relativamente parecidas, y lo que se obtiene es un fondo difuso.

- **Camuflaje**: El problema del camuflaje es de difícil solución. No se podrá ser capaz de distinguir del fondo un objeto, o parte de él, que sea muy similar.

- **Inicialización**: Se hace necesario de una etapa de inicialización en ausencia de movimiento. En algunos entornos y condiciones esto no es posible. El resultado es que quedan zonas sin inicializar en el fondo.
  Estas zonas no quedan indefinidamente en dicho estado. En el momento en que se detecta que no ha habido movimiento en la región durante un periodo de tiempo suficiente, se actualiza su fondo correspondiente.

5.3 Segmentación de imágenes y análisis de objetos

La segmentación de imágenes tiene su origen en numerosos estudios sicológicos que indican la preferencia de los humanos por agrupar regiones visuales en términos de proximidad, similitud y continuidad, para construir un conjunto de unidades significativas.
La segmentación es uno de los elementos más importantes de cualquier sistema automatizado de visión, es el primer nivel de la tarea de entendimiento de la imagen y afecta en gran medida al proceso posterior de interpretación de la imagen, proporcionando estructuras útiles.
El objetivo que persigue el proceso de segmentación es definir e identificar regiones que el sistema considerará objetos de primer plano. Se referirá al concepto de segmentación como el marcado de los píxeles de la imagen con un valor indicativo de su pertenencia a una determinada región o clase Ri según un criterio C.
La unidad significativa que rige la segmentación suele corresponder a píxeles, regiones o contornos que muestran o disciernen una similitud en cuanto a color, textura, movimiento.

Técnica de segmentación empleada

Region Growing

Esta técnica de segmentación examina los pixeles vecinos de los llamados “puntos semillas” y determina si estos pixeles deberían ser adicionados a los puntos semillas o no para formar un área. El proceso es puramente iterativo.

Algoritmo

Se utilizará un recorrido a lo ancho por toda la imagen, suponiendo como nodos, los pixeles y las aristas, los nodos vecinos.

Dado un punto y la imagen, se devolverá el objeto que conforma parte de ese punto. Primeramente se crea una cola con este punto y realiza sucesivas iteraciones, realizando la extracción de un punto de la cola, este se marca como ya tomado, para que no se procese dos veces, este punto se adiciona como otro punto del objeto de que se está analizando. Se buscan sus 8-vecinos y agregando estos a la cola. Este proceso se realiza hasta que la cola quede vacía y al final se dará como resultado una lista de puntos que conforman el objeto.

Análisis de objetos

Cada una de las regiones conexas etiquetadas se denominará objeto. Hasta ahora, la información que se tiene de un objeto es la posición de los píxeles que lo integran, identificados todos ellos con la misma etiqueta.

De estas regiones resulta muy interesante extraer información que permita caracterizar el objeto al que hacen referencia. Existen muchas formas de caracterizar un objeto, se escogieron las siguientes formas:

- **etiqueta**: Etiqueta k asociada a la región Rk.
- **num_puntos**: Número de píxeles de la región Rk, también se llamará N.
- **xmin, xmax, ymin, ymax**: Coordenadas máxima y mínima.
- **cgravx, cgravy**: Coordenadas del centro de gravedad.
- **cuadratura**: Da una medida de cómo se adapta un objeto a su bounding box. Si la cuadratura es ≈ 1 su forma es muy aproximada a la del bounding box. Si es $\ll 1$ se tratará de un objeto filiforme.
- **covar**: Es la matriz de los momentos de segundo orden. Es una estructura con cuatro campos correspondientes a $\sigma_x^2, \sigma_y^2, \sigma_{xy}, \sigma_{yx}$
- **media_color**: Es una estructura del tipo RGBQUAD1 que tiene tres campos, uno por componente RGB. Para cada componente de color se calcula el nivel medio. Si la imagen es de grises, se obtiene el nivel medio de gris y se guarda el mismo valor en los tres campos.

5.4 Algoritmo de seguimiento de objetos

En una secuencia de vídeo tomada en un escenario cualquiera durante un periodo de tiempo se puede distinguir objetos activos y pasivos. Los primeros tienen una evolución a lo largo del tiempo: se mueven –deprisa o despacio–, se quedan quietos, se cruzan unos con otros, se separan, ... Los segundos son estacionarios no cambian su posición ni características y constituyen lo que se ha denominado fondo. El fondo no ofrece demasiada información, siendo los objetos activos los que resultan de especial interés. Por este motivo se han desarrollado técnicas a lo largo del proyecto para extraer los objetos activos y caracterizarlos –tamaño, color, . . . –, pero aún puede darse un paso más que consiste en caracterizar su comportamiento y acciones a lo largo del tiempo, y esto se denomina seguimiento de objetos.

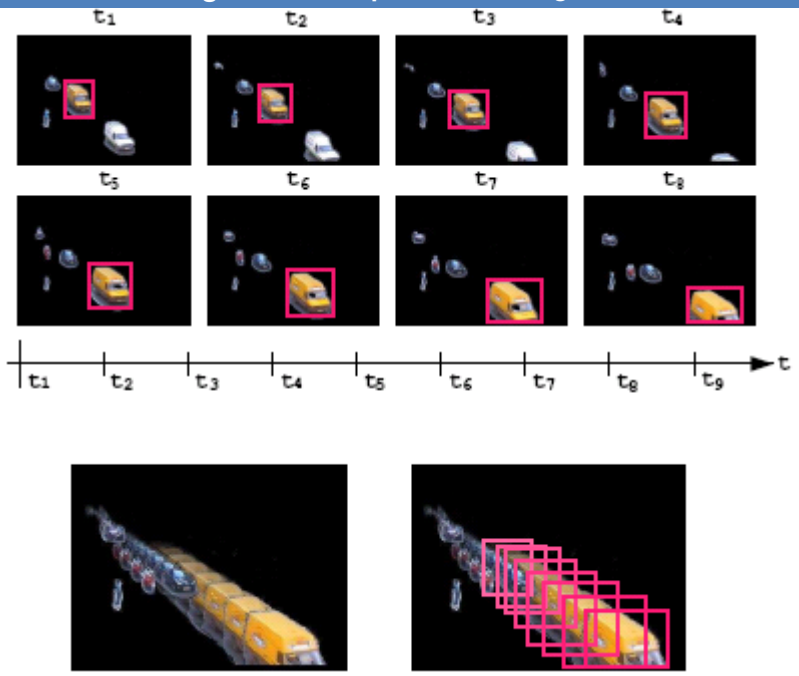
Desarrollar un sistema completo de seguimiento de objetos, object tracker, es el objetivo fundamental de este proyecto. El sistema pretende seguir cada objeto en movimiento, o temporalmente estático, como un blanco individual.

En los apartados anteriores se han descrito vídeo sensores que tienen la capacidad de funcionar de forma independiente. Este apartado, acerca al objetivo de sistema global perseguido, gracias al estudio de un algoritmo básico de control y de la forma de interacción de los anteriores vídeos sensores en su aplicación al object tracking.

Para el seguimiento de objetos se utilizó el criterio de solape de bounding box, así, si los objetos de frames seguidos, cuyos bounding boxes se solapan, están muy próximos, será el objeto que se está siguiendo.

En la figura 10 se puede ver el bounding box de uno de los objetos en cada fotograma y, la imagen formada por la superposición de ´estos, las zonas de solape entre ellos.

Figura 10 Solape de bounding box.



En este capítulo se explicaron en detalle los algoritmos empleados para la realización del video sensor de seguimiento de objetos. Se determinó que el Algoritmo de detección de movimiento quedaría dividido en tres bloques que realizan los procesos de: cálculo del módulo de la diferencia de dos imágenes consecutivas y conversión a gris, estimación de la varianza de la señal diferencia y el control de cese de movimiento. Además se describieron los algoritmos de estimación y sustracción de fondo, el de segmentación de imágenes y análisis de objetos, y el algoritmo de seguimiento de objetos, los cuales permitieron obtener el producto desarrollado.

CONCLUSIONES

Al concluir el presente trabajo, se ha cumplido con el objetivo planteado al inicio, aportando a través del análisis y el diseño, las bases para el futuro desarrollo de un video sensor para el seguimiento de objetos en secuencia de videos.

Se definió una arquitectura centrada en el flujo de datos, implementando el patrón “Filtros y Tuberías”, acorde a las necesidades operacionales, posibilitando la implementación de un sistema con alto rendimiento, extensibilidad, flexibilidad y reusabilidad del código.

Durante la presente investigación, se desarrolló un sistema que permite seguir un objeto en secuencias de video, con gran efectividad para el Sistema de Video Vigilancia.

RECOMENDACIONES

Al finalizar el proyecto, se recuerda a Blaise Pascal en su frase “lo último que uno sabe es por donde empezar”, ahora se conocen las debilidades del sistema y los posibles métodos para perfeccionar el trabajo. El inconformismo que provoca esta situación lleva a plantear mejoras sobre cada uno de los algoritmos implementados, con dos fines: uno a corto plazo con las mejoras necesarias para que el tracker tenga un funcionamiento óptimo, y el segundo, con una orientación más a largo plazo, que es conseguir sensores muy robustos en todas las situaciones y que proporcionen un base sólida en el desarrollo de nuevas herramientas y aplicaciones.

En cualquier caso, en estos momentos no se puede sino dejar algunas de ellas plasmadas como recomendaciones.

Tras este planteamiento, se pueden diferenciar dos áreas de investigación:

1. Perfeccionamiento de los módulos básicos del object tracker.
    - En todos los módulos de este sistema se requiere ausencia de variaciones abruptas de iluminación. La implementación llevada a cabo permite que el sistema se adapte lentamente al nuevo entorno, pero durante el periodo de adaptación se obtienen datos inválidos. El problema proviene de la estimación y posterior sustracción del fondo que considera casi toda la imagen primer plano al encontrar demasiada diferencia en una misma escena con condiciones de iluminación muy distintas.

      A este respecto se propone desarrollar un detector de situaciones de cambios de iluminación, posiblemente integrado en el sustractor de fondo, que lleve al sistema a una etapa de adquisición rápida de fondo y de este modo minimizar el tiempo en el que el sistema no responde adecuadamente. Teniendo en cuenta que se tiene por hipótesis que los objetos nunca van a ocupar la totalidad de la escena capturada, la técnica podría ser tan sencilla como comprobar qué proporción de la escena pertenece a objetos de primer plano y si resulta excesiva se considerará que es debido a un cambio de iluminación.
    - Ampliar la arquitectura del object tracker con módulos de detección adicionales como un detector de color o textura, útiles para el proceso de segmentación intraframe.

2. Desarrollo de nuevas aplicaciones que basen su funcionamiento en el seguimiento de objetos.

- Detección de comportamientos sospechosos.
- Seguimiento multicámara. Permitir el seguimiento de un objeto a través de varias cámaras fijas.
- Control automático de una cámara móvil secundaria para enfocar objetos de interés en primer plano.
- Aplicaciones médicas. Análisis de interacciones entre moléculas en células vivas ya que el proceso puede basarse en adquirir una serie de imágenes de microscopía de las células a lo largo del tiempo para después realizar con ellas unas operaciones matemáticas y obtener una medida de la interacción. Estudio de migración de células tumorales en modelos de invasión que precisan el seguimiento de trayectorias.
- Diseño de interfaces hombre-máquina basados en reconocimiento de gestos.
- Aplicaciones en el software de entretenimiento. Promover la interactividad del usuario consiguiendo que las acciones que se realizan en el programa provengan de los movimientos de éste.

BIBLIOGRAFÍA

1. **Akio Namiki, Yoshihiro Nakabo, Idaku Ishii, and Masatoshi Ishikawa. 1999**. High speed grasping using visual and force feedback. Detroit : s.n., 1999.

2. **Alonso, Antonio Albiol Colomer y Cristina Sandoval. 2003**. Seguimiento de objetos en secuencia de video. 2003.

3. **Antonio Albiol, Cristina Sandoval, and Alberto Albiol. 2003**. Robust motion detector for video surveillance applications. Universidad Politecnica de Valencia. 2003.

4. **Antonio Albiol, Valery Naranjo, and Josep Prades. 1999**. Tratamiento digital de la señal. Teoría y aplicaciones. Universidad Politécnica de Valencia. 1999.

5. **C. Ridder, O. Munkelt, and H. Kirchner. 1995**. Adaptive background estimation and foreground detection using kalman filtering. 1995.

6. **C.S. Regazzoni, V.Ramesh, and G.L. Foresti. 2001**. Special issue on video communications, processing and understanding for third generation surveillance systems. 2001. Vol. 89.

7. **Crowley, J. H. Piater and J.L. 2001**. Multi-modal tracking of interacting targets using gaussian approximations. 2001.

8. **G.L. Foresti, Lucio Marcenaro, and C.S. Regazzoni. 2002**. Automatic detction and indexing of video-event shots for surveillance applications. 2002.

9. **Introduction to automata theory languages, and computation**. Ullman, John E. Hopcroft and Jeffrey D. 1993. 1993.

10. J**ulio Pons, Josep Prades-Nebot, Antonio Albiol, and Jesus Molina. April 2002.** Fast motion detection in the compressed domain for video surveillance. April 2002.

11. **Kentaro Toyama, John Krumm, Barry Brummit, and Brian Meyers. 1999**. Wallflowers: Principles and practice of background maintenance. Redmond : s.n., 1999.

12. **Koller, Dieter. 1996**. Moving object recognition and classification based on recursive shape parameter estimation. 1996.

13. **Lipton, Robert Collins and Alan. May, 2000**. A system for video surveillance and monitoring. Robotics Institute, Carnegie Mellon University. Pittsburgh : s.n., May, 2000.

14. **Pedro García, Encarna Segarra, Tomás Pérez, José M. Sempere, José Ruiz, and M. Vázquez de Parga. 1996**. Apuntes sobre la teoría de autómatas y lenguajes formales. 1996.

15. **T.Ebrahimi, E. Duracan and. October 2001**. Change detection and background extraction by linear algebra. October 2001.

16. **Takashi Komuro, Idaku Ishii, and Masatoshi Ishikawa. 1999**. General-purpose vision chip architecture for real-time machine vision. 1999.

17. **Whelan, B.G. Batchelor and P.F. 1994**. Machine vision systems: Proverbs, principles, prejudices and priorities. 1994.