UNIVERSIDAD AUTÓNOMA DEL ESTADO
DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE
INFORMACIÓN Y SISTEMAS

Detección y Seguimiento de Movimiento Remoto

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS COMPUTACIONALES

P R E S E N T A

URIEL EDGARDO ESCOBAR FRANCO

ASESORES: **DR. GUILLERMO SÁNCHEZ DÍAZ**
DR. JAIR GARCÍA LAMONT

PACHUCA DE SOTO, HIDALGO. AGOSTO 2008

Gracias…

… en primer lugar, a Dios que me ha permitido vivir, aprender, aplicar y compartir su sabiduría.

… a mi Papá, mi Mamá y mi Hermano por los consejos, la guía, el apoyo, la confianza y el amor que me dieron y dan sin medida.

… a mi Esposa, por la paciencia y por las horas de ausencia que soportó mientras escribía esta tesis.

… a Víctor Manuel Lugo Sánchez, mi otro hermano, por compartir la vida y hacer ameno el camino.

… a mis asesores Dr. Guillermo y Dr. Jair, por el tiempo dedicado a este trabajo, por compartir sus conocimientos conmigo y por las respuestas que encontramos juntos.

A Blanca Paulina, mi hija, tu sonrisa me inspira y me alienta.

Resumen

La detección y seguimiento de movimiento de objetos en imágenes ha tenido una aplicación importante en diversas áreas, como seguridad, medicina, entre otras.

Actualmente, se han desarrollado diversas aplicaciones tanto libres como comerciales para dar una solución al problema de detección y seguimiento de movimiento, de manera local o remota (vía Internet).

Las técnicas libres encontradas en el estado del arte, presentan ciertas limitaciones que no permiten hacer una detección y seguimiento de los objetos en movimiento en línea de manera práctica. Algunos tienen la capacidad de almacenar las imágenes captadas y pueden recuperarlas accesando a una cuenta de correo, lo cual le impide definitivamente la opción de poder monitorear en línea el movimiento. Otras herramientas realizan la detección de movimiento, pero exclusivamente de manera local.

En este trabajo de tesis, se plantea el diseño y desarrollo de un prototipo de software para la detección y seguimiento de movimiento de manera remota, basada en la adquisición de imágenes por medio de una cámara web digital, fija.

El prototipo se orienta al uso de cámaras web comerciales y comunes, no especializadas, evitando el uso de hardware adicional. Se implementa un algoritmo que calcula el porcentaje de diferencia entre dos imágenes como medida para la detección del movimiento, aunque es posible modificar diferentes módulos con pocos ajustes en el prototipo planteado.

Se propone además, el planteamiento de un modelo de procesamiento que utilice tanto los recursos de la computadora conectada a la cámara, así como los recursos de la computadora remota en donde se realizará el seguimiento del movimiento.

Una vez que se ha detectado movimiento, el sistema transmitirá las imágenes a través de Internet, desde una computadora en la cual se realiza la detección de movimiento y que tiene conectada la cámara web, hacia una computadora en la cual se hará el seguimiento del movimiento usando las imágenes recibidas.

Además, de manera experimental, se verificó que el prototipo presentado en este trabajo opera en forma remota con una mayor rapidez que las aplicaciones libres que se encontraron en el estado del arte.

Abstract

The detection and tracking moving objects in pictures have been an important application in various areas such as security, medicine, among others.

Currently, many applications have been developed, much free trade, as a solution to the problem of detection and monitoring, so both local and remote (via Internet).

The techniques found in the free state of the art, have some limitations that do not allow for detection and tracking of moving objects online in a practical way. Some have the ability to store images captured and can recover them accessing to an email account, which definitely takes away the option of being able to monitor online movement. Other tools perform motion detection, but only on a local level.

In this thesis work raises the design and development of a prototype software to detect and track movement of remotely based on imaging through a digital webcam.

The prototype is geared to use commercial webcams and common in this way is to avoid the use of special hardware. So original, it implements an algorithm that calculates the percentage difference between two images as a measure for the detection of movement. However, it is possible to modify different modules with a few adjustments in the prototype raised.

It is also proposed, the approach of a processing model that uses both the resources of the computer connected to the camera, as well as the resources of the remote computer where they monitor the movement.

Once movement has been detected, the system will transmit images over the Internet from a computer in which it conducts motion detection and is connected webcam, to a computer which will monitor the movement using the images received.

In addition, on an experimental basis, it was verified that the prototype presented in this paper operates with a remote faster than the free applications that were found in the state of the art.

Contenido

Capítulo 4 “Experimentación”

Capítulo 5 “Conclusiones”

Índice de Figuras

Índice de Tablas

Introducción

La detección y seguimiento de objetos en video es de gran utilidad en numerosas aplicaciones como robótica, diagnóstico médico, monitoreo de sistemas de seguridad y animación por computadora, entre otras.

La detección y seguimiento de movimiento en video es una manera alternativa de implementar la monitorización de algunas situaciones en donde se requiera la presencia de un observador humano por largo tiempo, como es el caso de los sistemas de vigilancia. Actualmente algunos prototipos automatizados de vigilancia permiten obtener estadísticas de movimientos detectados sin necesidad de la intervención humana.

Además, con la evolución que se ha dado en las herramientas computacionales se ha obtenido significativamente una mayor velocidad de transmisión de datos en las redes de telecomunicaciones y el aumento de la capacidad de procesamiento y almacenamiento de las computadoras y dispositivos móviles. Dicha evolución ha traído consigo el abaratamiento de la tecnología y los servicios de telecomunicaciones haciendo posible el desarrollo de prototipos económicos y eficientes para la detección y seguimiento de movimiento en video.

Gracias a lo anterior, existen diversos algoritmos ampliamente utilizados para la detección de movimiento [MaF+ 2002], así como también para el seguimiento del movimiento [PaD+ 2004]. Entre ellos destacan por su sencillez, para la detección de movimiento el algoritmo de promedio de diferencias absolutas entre píxeles correspondientes para dos imágenes presentado en [BrT 2003] y el algoritmo del Error Cuadrado Mínimo (LSE), y para el seguimiento, el uso de operaciones booleanas aplicadas a imágenes o comparación de intensidades entre los píxeles de las imágenes. Sin embargo, en la mayoría de los prototipos, la información extraída por estos algoritmos es tratada de manera local limitando las aplicaciones y flexibilidad de los mismos.

En este trabajo se utilizan técnicas de procesamiento digital de imágenes como herramienta fundamental para la detección y seguimiento de movimiento. En la detección de movimiento se implementa un algoritmo basado en el método presentado en [BrT 2003] ya que requiere pocos parámetros y además necesita hacer un número de cálculos relativamente bajo.

Cuando el prototipo detecta movimiento en el video, éste procede a enviarlo a través de Internet usando una implementación del Protocolo de Tiempo Real (RTP) y compresiones JPEG o H.263, el primero utilizado para conexiones de alta velocidad y el segundo para conexiones de media y baja velocidad.

El seguimiento del movimiento se realiza utilizando un algoritmo que compara la diferencia que existe entre cada píxel que compone a una imagen con el píxel correspondiente en una imagen de referencia que se guarda cada *n* cuadros de imagen, denominados frames, normalmente cada 1, 2, 3, 5, 7 o 10 frames.

Aportaciones del trabajo de tesis

Esta tesis plantea una propuesta novedosa, modular, simple y de fácil modificación, para solucionar algunos de los problemas que acarrea la detección y seguimiento de movimiento remoto. Se aporta al estado del arte, además del modelo, un prototipo que:

- Es capaz de procesar dos tamaños de video.
- Permite utilizar dos protocolos para la transmisión de video por Internet, dejando a criterio del usuario la elección de cada uno de ellos dependiendo del entorno de operación del prototipo.
- No hace uso de hardware ni software especializado ni costoso.
- Su implementación es económica.
- Puede ser usado con computadoras y cámaras web convencionales; y los sistemas operativos más extendidos.

Objetivo general

Desarrollar un prototipo de software que haga uso de cámaras web digitales e Internet y que implemente algoritmos de procesamiento digital de imágenes para realizar la detección y seguimiento de movimiento remoto.

Objetivos específicos

- Diseñar una arquitectura del prototipo base para aplicaciones de procesamiento de video a través de Internet haciendo uso del modelado orientado a objetos para que sea: modular, adaptable y sencillo de modificar.
- Obtener imágenes de manera local utilizando una cámara web digital para poder procesarlas.
- Realizar la detección de movimiento de manera local utilizando algoritmos de procesamiento digital de imágenes para determinar cuándo se enviará el video a través de Internet.
- Transmitir por Internet la secuencia de video donde se observa al objeto en movimiento mediante el uso del protocolo de transporte de tiempo real para que la computadora que recibe el video lo procese.
- Realizar el seguimiento de objetos en movimiento del video recibido utilizando algoritmos de procesamiento digital de imágenes para resaltar el movimiento de los objetos.
- Construir el prototipo de software sin hacer uso de hardware ni software especial (tarjetas, cámaras, sistemas operativos de tiempo real, etc) explotando las características del software libre y las cámara web digitales de uso común para reducir el costo de desarrollo y operación del prototipo.

- Implementar el prototipo de software, tratando de que sea lo más rápido y eficiente posible, en lenguaje Java mediante el uso de herramientas de desarrollo gratuitas para poder diseñar interfaces visuales ricas e intuitivas para la mayoría de los usuarios que operen el prototipo (comparado con las interfaces de las aplicaciones libres encontradas en el estado del arte).

Justificación

La detección y seguimiento de movimiento de objetos en imágenes ha tenido una aplicación importante en diversas áreas como seguridad, medicina, observaciones en experimentos de laboratorio, aplicaciones militares, entre otras.

A pesar de la importancia que implican estos problemas, los prototipos de detección y seguimiento de movimiento que existen disponibles actualmente en el mercado presentan las siguientes limitantes:

- La mayoría de ellos son costosos.
- Algunos requieren hardware especializado (tarjetas de adquisición y/o transmisión, PID, etc.).
- Cuentan con detección y seguimiento de movimiento de manera local exclusivamente.
- Algunos sistemas que realizan transmisión de video por Internet y que detectan movimiento, primero almacenan el video y después lo comienzan a transmitir, y en el peor de los casos, únicamente lo almacenan y lo ponen a disposición de los usuarios a través de un navegador de Internet, lo cual genera el inconveniente de que el envío de video no se hace en un tiempo real ni cercano a éste.
- Algunos sistemas son dependientes de plataformas específicas y difíciles de modificar.

Tomando en cuenta lo anterior, la implementación de un prototipo para la detección y seguimiento de movimiento remoto que utilice técnicas de procesamiento digital de imágenes, y que además:

I. Haga uso de computadoras y cámaras convencionales,
II. Reduzca el tiempo de envío de las secuencias de imágenes a través de Internet,
III. Sea capaz de funcionar en los sistemas operativos más extendidos y,
IV. Esté construido de manera modular.

servirá como base para la construcción de nuevos prototipos más robustos, económicos y de fácil modificación para probar nuevos algoritmos de detección y seguimiento de movimiento de manera combinada o individual y también de forma local o remota.

Capítulo 1

Estado del Arte

1.1 Trabajos relacionados

En el trabajo de Zeljkovic y Pokrajac [ZeV+ 2006] se propone un sistema de vigilancia para apartamentos y condominios utilizando la integración de inteligencia artificial, detección y seguimiento de movimiento, bases de datos multimedia y conectividad con telefonía celular e Internet. En la Figura 1.1 se muestra la arquitectura del sistema de vigilancia propuesto en [ZeV+ 2006].

El sistema está compuesto de las siguientes partes:

1) Una cámara RGB/IR estática e inalámbrica de circuito cerrado de televisión.
2) Una computadora que sirve de servidor para el almacenamiento de las imágenes captadas por el sistema y el envío de éstas a través de Internet. Éste servidor se encuentra permanentemente conectado a Internet a través de una conexión tipo DSL con un modem.

El funcionamiento del sistema se inicia a través de la adquisición de datos en tiempo real desde la cámara hacia el servidor, en donde el video es analizado para efectuar la detección de movimiento. Si se detecta movimiento en la secuencia de video, entonces se procede a tomar una fotografía digital de la persona, esta fotografía es almacenada en una base de datos multimedia y se procede a la identificación de la persona comparando la fotografía con otras previamente almacenadas.

El sistema puede enviar dos tipos de mensajes al usuario: un mensaje de información junto con la fotografía de la persona, hora y fecha del evento, y otro mensaje de alerta con la fotografía de la persona, hora y fecha del evento con mayor prioridad que el mensaje de información. Ambos mensajes son enviados por correo electrónico y opcionalmente al teléfono celular del usuario del sistema. El sistema también brinda la posibilidad de transmitir una secuencia de video a través de Internet para ser observada por el usuario utilizando un navegador de web.

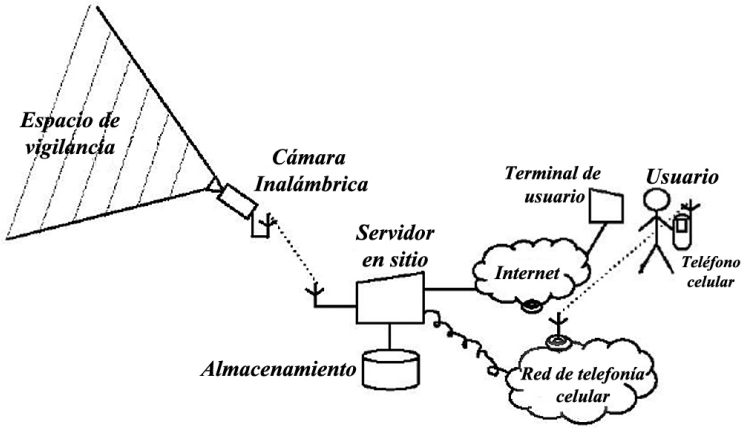


Figura 1.1. Arquitectura del sistema propuesto. [ZeV+ 2006].

El trabajo de los autores cuenta con un algoritmo que determina el carácter de las actividades de una persona frente a una puerta. Cuando el objeto se mueve dentro del campo de visión de la cámara el sistema identifica el movimiento e inicia un cronómetro a la vez de que continúa el seguimiento de los objetos y los graba. En un momento particular, cuando el objeto se detiene, el sistema toma la fotografía del objeto y la almacena en el servidor. Cuando el objeto sale del campo de visión el cronómetro se detiene.

Si el tiempo de grabación de video en donde se aprecia el objeto o persona es mayor a un tiempo preestablecido (umbral), el sistema lo considera como una situación inusual y envía la alarma al usuario.

La detección del movimiento en el sistema propuesto se puede hacer utilizando dos métodos distintos:

1) Basado en el cambio de iluminación de los píxeles y,
2) Basado en bloques espaciotemporales.

En el trabajo Zeljkovic y Pokrajac los autores no dan detalles de los algoritmos de detección de movimiento utilizados, ni de las implementaciones de la transmisión del video ni la conexión con los teléfonos celulares. Solo referencias a otros trabajos donde se han utilizado.

El seguimiento de movimiento se realiza a través de la binarización de las imágenes aplicando la operación XOR entre dos frames, como se muestra en la figura 1.2.

Además en el trabajo de [ZeV+ 2006] los autores mencionan que las evaluaciones en tiempo real de su sistema aún están por verificarse y también se incluyen algunas pruebas experimentales de su sistema.

Por otro lado en el documento publicado por M. Wang, C. Huang y H. Lin [WaM+ 2006] se presenta un enfoque innovador de los sistemas de vigilancia, al utilizar una videocámara omnidireccional, la cual provee un ángulo de visión de 360 grados en una sola secuencia de imágenes. También se propone una técnica de reconocimiento de la actividad humana.

La cámara omnidireccional debe ser instalada siempre en el centro de los lugares del entorno que será vigilado, además, para el mejor aprovechamiento de las características propias de la cámara, ésta debe situarse en una parte alta y regularmente se coloca en el techo para proveer una vigilancia global del entorno. Ver figura 1.3.

En el prototipo desarrollado por M. Wang, C. Huang y H. Lin son implementadas funciones como la detección de movimiento, el seguimiento de objetos y un análisis del comportamiento de los objetos en movimiento.

Figura 1.2. Seguimiento de movimiento en el sistema [ZeV+ 2006].

Figura 1.3. Percepción de la cámara omnidireccional [WaM+ 2006].

Figura 1.4. Resultado del seguimiento omnidireccional de una persona utilizando el algoritmo CamShift [WaM+ 2006].

Figura 1.5. Resultado del seguimiento omnidireccional de una persona utilizando el flujo óptico [WaM+ 2006].

La detección de movimiento se basa en la creación de una imagen de fondo. El algoritmo que se ha implementado es el de MHI (Motion History Image) y se encuentra explicado en [WaM+ 2006].

El seguimiento de los objetos se basa en el algoritmo CamShift (Continuously Adaptive Mean Shift) el cual extrae la información del color del objetivo, como se define en [WaM+ 2006]. Ver figura 1.4.

Para mayor robustez del prototipo y hacerlo adaptable a diferentes cambios de iluminación se ha implementado también un modelo de flujo óptico [BuA 2002], que ayuda a detectar pequeños cambios en el objeto en movimiento. Ver figura 1.5.

En el prototipo, el seguimiento de un objeto se puede realizar con la combinación de ambos métodos.

Para lograr la ubicación espacial del objeto, la cámara es calibrada para establecer correspondencias uno a uno entre los píxeles de la imagen y algunas referencias en el fondo de la escena, de tal forma que se pueda obtener una estimación del objeto en movimiento en la escena.

En el documento [WaM+ 2006], los autores presentan un método de detección de caídas para el cuidado de ancianos que se basa en las correspondencias uno a uno entre la imagen de la

cámara y el suelo de la habitación, combinado con la trayectoria que tiene el objetivo que se sigue en la imagen.

La detección de caídas se basa en la correspondencia uno a uno y en la definición de tres tipos de zonas en el video que son establecidas en base a la densidad de puntos con movimiento que se presenta en la imagen. Estas zonas son:

a) **Zona de inactividad:** que son las regiones en las cuales no existen cambios de movimiento significantes, estas zonas se definen como zonas en donde las personas pasan la mayor parte del tiempo.
b) **Zona de actividad:** que son las regiones en las cuales existen cambios frecuentes en la escena en las cuales existe en la mayor cantidad de movimiento.
c) **Zonas de no actividad:** se refiere a zonas en las cuales no debe existir ninguna actividad humana

Con la ayuda de las definiciones de zonas anteriores y de acuerdo con la zona en donde es detectado el movimiento se lanza una advertencia.

Los aspectos más relevantes del prototipo se pueden resumir en:

1) Los componentes del sistema son: Cámara de Circuito Cerrado equipada con un lente anular panorámico, un grabador de imágenes y una PC.

2) La velocidad de transferencia de las imágenes desde la cámara a la PC es de 7.5 frames por segundo (fps) con dimensiones de 320 x 240 píxeles de ancho y alto respectivamente.

3) La geometría de la imagen panorámica que recibe la PC es compleja y puede modelarse como un sistema coordenado cilíndrico, la detección y seguimiento de movimiento se realiza en ese sistema de coordenadas.

Otro trabajo relacionado con la temática que se aborda en ésta tesis es el sistema presentado por M.F. Abdelkader, R. Chellapa, Q. Zheng y A. L. Chan [AbM+ 2006] el cual es un sistema integrado de vigilancia visual para exteriores. Este sistema adquiere las imágenes desde una cámara fija, de color o infra-roja.

El método utilizado por los autores en [AbM+ 2006] para realizar la detección del movimiento se basa en la combinación de dos técnicas de detección, la variación temporal de la intensidad de los píxeles y el modelo de fondo de imagen.

Con la combinación de los modelos anteriores los autores pretenden alcanzar un comportamiento de detección fiable y exacto con el objetivo de evitar falsas alarmas.

Los autores en [AbM+ 2006] calculan el promedio y la varianza de la intensidad de cada píxel a través del varios frames previos y es actualizada continuamente cada nuevo frame.

También se hace uso de un modelo de fondo de escena el cual es construido al actualizar recursivamente un conjunto independiente de promedio y varianza.

Una vez que el movimiento ha sido detectado se procede a la segmentación del movimiento, que los autores en [AbM+ 2006] consideran como una interfaz entre la detección de movimiento y el seguimiento del movimiento del sistema de vigilancia.

En esta etapa se lleva a cabo la segmentación de las áreas donde hay movimiento en el mapa de detección binario en objetos disjuntos, se elimina cualquier ruido pequeño o aislado, y se inicializan los recuadros que son pasados a la etapa de seguimiento de movimiento.

En la etapa de segmentación los autores de [AbM+ 2006] utilizan el algoritmo de etiquetamiento de componentes conectados presentado en [SuK+ 2000].

El seguimiento de los objetos en movimiento se realiza utilizando la estimación de la localización de cada objeto en cada nuevo frame. Esta estimación se realiza utilizando un algoritmo que incorpora información de la apariencia y movimiento del objeto.

Las funciones principales del sistema son las siguientes:

- Algoritmo de detección de movimiento que integra la variación temporal y el modelo de fondo de escena que permite una robusta detección de objetos en movimiento.
- Algoritmo adaptable de seguimiento visual del objeto en movimiento, que utiliza el aspecto y las observaciones del movimiento para construir un marco estadístico. También se utilizan filtros de partículas en la estimación para conciliar los cambios de apariencia. Este algoritmo de seguimiento se adapta a la velocidad del movimiento, apariencia, ruido y al número de partículas del nuevo frame.
- La integración de las etapas de la detección de movimiento y seguimiento de movimiento utilizando un módulo de detección de movimiento para la inicialización del modelo de apariencia para el seguimiento visual, y para obtener la observación del movimiento.
- Se implementa un módulo de evaluación responsable de la medición y evaluación del rendimiento del sistema con respecto a un conjunto de datos verdaderos de detección de movimiento.

También se encontró en el estado del arte otro trabajo interesante, se trata de la tesis de maestría elaborada por Domínguez Jiménez [DoI 2006] en la cual se desarrolló un prototipo capaz de detectar y seguir objetos en movimiento de manera local.

La meta principal del prototipo de [DoI 2006] es conocer la ubicación espacial de los objetos que presenten movimiento dentro de las secuencias de imágenes captadas mediante una cámara digital fija.

En [DoI 2006] se trabaja con una cámara web conectada a la PC mediante el puerto USB, con un tamaño de imagen de 320 píxeles de ancho por 240 píxeles de alto.

La imagen se adquiere de la cámara web con una paleta de colores RGB que utiliza 8 bits por cada uno de los componentes R, G y B, para representar 256 niveles de brillo en cada uno de ellos.

En [DoI 2006] se realiza una conversión de la paleta de colores RGB a la paleta de 256 colores en escala de grises para lograr una mayor optimización de las operaciones matemáticas que el prototipo tiene que realizar.

En el desarrollo del prototipo se implementaron técnicas de procesamiento digital de imágenes, entre las más importantes destacan:

- Algoritmo del Error Cuadrado Mínimo (*Least Square Error* o *LSE*) que es utilizado como medida de disimilaridad para obtener diferencias entre las imágenes y así

conocer cuando existe un cambio significativo que pudiera considerarse como movimiento.

- Operadores de Sobel para llevar a cabo el cálculo de gradientes en las imágenes y hacer la segmentación de los objetos presentes en la escena.
- Operaciones booleanas aplicadas a las imágenes, para la obtención de los objetos en movimiento dentro de la escena.
- Filtrado de la imagen resultante para obtener un mejor cálculo del centroide de los objetos en movimiento.

En el trabajo de [DoI 2006] se considera que cuando existen objetos en movimiento en una escena, siempre existen cambios en la intensidad de los píxeles de la imagen, y para conocer dichos cambios se utiliza el Algoritmo del Error Cuadrado Mínimo mencionado en [DoI 2006].

El autor de [DoI 2006] determina que las imágenes obtenidas con la cámara digital, presentan ruido de tipo gaussiano por lo que se requiere de la utilización de un valor de umbral que representa una cierta cantidad de ruido que es considerado como "normal" y cualquier valor mayor a éste umbral es considerado como movimiento detectado.

Después de que es detectado el movimiento en la secuencia de imágenes se procede a realizar su segmentación, y dado que en una escena pueden existir diversos objetos en movimiento se debe determinar cuáles son los nuevos objetos que aparecen en la escena o segmentar solo los nuevos objetos.

Para lograr la diferenciación entre objetos previos y objetos nuevos, en [DoI 2006], se utilizan los operadores lógicos *AND* y *XOR* aplicados a las imágenes segmentadas.

En [DoI 2006] se obtiene la ubicación espacial del objeto dentro de la imagen obteniendo el centro del objeto en movimiento detectado.

Otro trabajo interesante dentro del estado del arte es el trabajo publicado por Jing, Rajan y Eng Siong [JiG+ 2005], en el cual se propone un método de detección de movimiento que hace uso del ajuste dinámico del umbral de detección de movimiento.

La idea básica es utilizar un umbral dinámico para binarizar la diferencia que existe entre una imagen con el fondo de escena y una nueva imagen entrante. Una imagen con el fondo de escena adaptativo es mantenida y actualizada periódicamente.

La imagen entrante es comparada con la imagen que contiene el fondo de la escena para obtener una diferencia absoluta de las imágenes, si se establece que existe un objeto en movimiento en la escena se utiliza el método de umbral de Otsu [OtN 1979] para encontrar el valor del umbral para binarizar la diferencia entre las imágenes y poder clasificar a cada uno de los píxeles como píxel de fondo de escena o píxel de primer plano de escena.

El método de [JiG+ 2005] realiza cada una de las siguientes etapas:

- Aprender el modelo de fondo inicial de escena y el nivel de ruido de la imagen: algunas imágenes de entrenamiento son usadas para construir el modelo de fondo de escena y estimar el nivel de ruido de las imágenes.

- Detección de movimiento: los pasos para la detección de movimiento que se siguen son:
    - Obtener la diferencia absoluta entre las imágenes y calcular la diferencia absoluta minimizada (*N AD* [JiG+ 2005]).

- Comparar el *N AD* con el promedio de los niveles de ruido para decidir si existe un objeto en movimiento en la escena o no.
- Aplicar el umbral de diferencia de imagen para obtener el primer plano.

- Actualizar el fondo de escena y el promedio del nivel de ruido: el fondo de escena y el promedio del nivel de ruido pueden ser actualizados periódicamente para hacer el algoritmo más robusto.

Algunos de los aspectos más importantes de este método son su rapidez y además es robusto en cuanto al cambio gradual de iluminación.

Este método de detección de movimiento fue probado en varias secuencias de imágenes y se han obtenido buenos resultados [JiG+ 2005]. Ver figura 1.6.

Otro trabajo que se encontró en el estado del arte es el trabajo publicado por Park, Gambini y Mejail [PaD+ 2004], en el cual se propone un método estable y eficiente para evitar errores de parametrización al ajustar el contorno del objeto con una curva B-spline al comienzo del método de seguimiento. Se utiliza una estructura de aceleración para evitar conflictos al estimar el contorno del objeto.

El algoritmo comienza con una curva B-spline inicial que se ajusta al contorno del objeto en el primer cuadro de la secuencia. Para encontrar esta curva es necesario hallar los puntos de borde del objeto.

Para hallar los puntos de borde del objeto se define un área inicial de búsqueda determinada también por una curva B-spline y se consideran rectas equiespaciadas normales a esta curva. Luego se aplica algún algoritmo de detección de bordes sobre los segmentos de recta.

Entonces la imagen se recorre por regiones en lugar de hacerlo sobre toda la imagen, lo que significa un gran ahorro en costo computacional. Luego, la curva inicial se deforma según movimientos permitidos, restringidos al espacio de formas. Sin embargo una mala elección de normales al comienzo del algoritmo de seguimiento puede significar un ajuste poco adecuado por lo que el algoritmo falla.

Figura 1.6. Detección de movimiento de personas caminando. La primera fila muestra las escenas originales, la segunda y la tercera fila muestran los primeros planos obtenidos por el método de [JiG+ 2005] y el método del modelo mixto gaussiano, respectivamente [JiG+ 2005].

En el trabajo de [PaD+ 2004] se utilizan *Bounding Boxes* para obtener una curva B-spline de ajuste muy preciso al comienzo del seguimiento.

Las *Bounding Boxes* se utilizan en muchas aplicaciones de procesamiento digital de imágenes. Esta representación simplifica la geometría del objeto tratándolo como un rectángulo que lo contiene (de ahí la definición de *caja contenedora*).

Observaciones importantes sobre este trabajo:

- El algoritmo de seguimiento utilizado en este trabajo comienza con la definición de una curva B-spline que es el área inicial de búsqueda del contorno de un objeto. Luego se consideran una serie de segmentos de rectas normales a esta curva y se aplica algún método de detección de bordes para hallar puntos sobre el contorno a lo largo de las rectas.
- El método de las Bounding Boxes permite obtener una curva B-spline de ajuste muy preciso.

Otro trabajo que forma parte del estado de arte es el realizado por Mélendez Islas [MeA 2003], en el cual se describe una técnica para detectar objetos en movimiento a partir de secuencias de imágenes, donde la cámara está fija y las imágenes son de tipo radar, infrarojas o de luz visible. Ver figura 1.7.

En [MeA 2003] se usan dos métodos en el proceso de segmentación, ambos fundamentados en la técnica de detección de fondo. En el primero se usan diferencias acumulativas, mientras que en el segundo se basa en probabilidades, representando cada píxel por medio de una mezcla de distribuciones normales.

Una vez detectado el fondo, se lleva a cabo una umbralización y después se aplican operadores morfológicos con el fin de eliminar regiones que no corresponden a los objetivos.

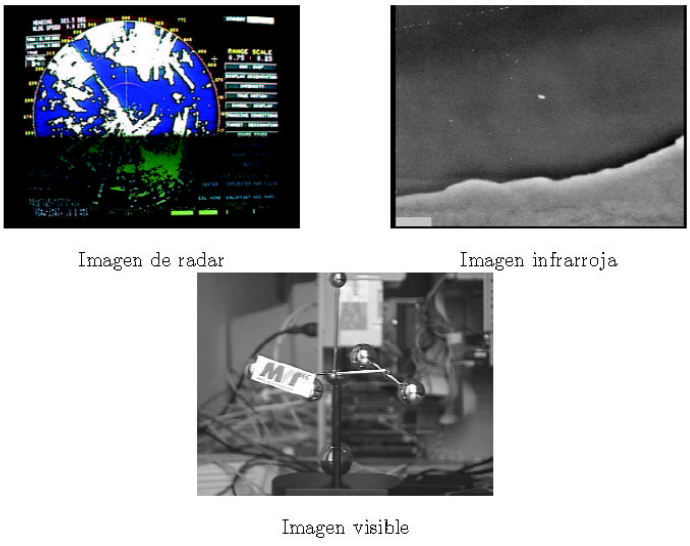


Figura 1.7. Tipos de secuencias de imágenes que pueden ser procesadas en el trabajo de Meléndez [MeA 2003].

Además, con el fin de refinar los resultados del proceso de segmentación, se implementó un algoritmo de seguimiento basado en el principio de establecer correspondencias de objetos entre las imágenes de la secuencia, dada la distancia entre ellos.

En el caso del método de detección de fondo basado en diferencias acumulativas en [MeA 2003] se concluyó que se tienen mejores resultados cuando el fondo es muy contrastante con los objetos, ya que se define un umbral para determinar si cada píxel en estudio forma parte de un objeto o no. Así, si el fondo tiene un nivel de gris cercano al umbral, se tomará el fondo como parte de un objeto, resultando en la detección errónea de grandes áreas.

El método de detección de fondo basado en probabilidades tiene el inconveniente del tiempo que tarda en ejecutarse. En [MeA 2003] se menciona que la mayor parte del tiempo la consume la etapa de supresión de detecciones falsas.

En el trabajo de Meléndez no se realiza procesamiento en tiempo real sino que se hace una comparativa entre los dos métodos propuestos que implementa.

Por otra parte, en el trabajo de Durán de Jesús, Villacorta Calvo e Izquierdo Fuente [DuJ+ 2000] se presenta un modelo para un sistema basado en dos sistemas sensoriales complementarios, imagen y acústico, que permiten fusionar la información recibida y priorizar los recursos computacionales en las zonas de mayor riesgo.

El sistema está compuesto por las siguientes partes:

- Módulo de radar acústico
- Módulo de captura de video
- Módulo de red

El sistema utiliza un número arbitrario de sensores (sondas de monitorización), distribuidos en el recinto de vigilancia, una red de datos y un servidor centralizado (gestor de monitorización).

Los sensores están constituidos por un microprocesador con su memoria y tres módulos principales:

- Radar acústico
- Tarjeta de captura de video
- Tarjeta de red

El microprocesador procesa la información recibida por los módulos, ésta se ensambla en paquetes y se transmite al servidor central a través de la red de datos. El sistema utiliza protocolos IP y SNMPT v2 para la gestión de las comunicaciones.

La filosofía de trabajo del sistema se basa en dos modos de monitorización. El primero soportado por el radar acústico que permite detectar automáticamente la presencia de intrusos al tiempo de estimar y seguir su posición bidimensionalmente.

Con la información obtenida por el radar acústico se activa el sistema de video, que es de mayor carga computacional, que se posiciona sobre la zona predetectada y mediante una red neuronal clasifica la zona inspeccionada aumentando notablemente la probabilidad de detección.

También se utilizan algoritmos de monitorización que predicen la posición de los intrusos y permiten realizar un seguimiento visual más preciso.

Características importantes del sistema:

- El sistema radar permite detectar por procedimientos acústicos en la banda de audio (5khz – 20khz) la existencia de objetos en un espacio de vigilancia acotado, estimar su posición espacial bidimensional y generar una alarma de detección acústica.

- Se utilizan técnicas pulso-eco empleadas en los sistemas radar y sonar que permiten determinar el rango.

- Las imágenes obtenidas mediante la tarjeta de video son fraccionadas en celdas de tamaño N x N que son utilizadas como entradas de una red neuronal. La salida de la red neuronal nos indicará si la celda ha cambiado con respecto al contenido anterior de la celda.

- La red neuronal utilizada es un perceptrón multicapa con realimentación simple. Para el cálculo de los pesos de cada unidad de la red se debe realizar un entrenamiento previo utilizando el algoritmo de backpropagation.

- Cada sonda de monitorización tiene asignada una dirección de red IP y un identificador SNMP con el nombre, localización, etc.

- El gestor realiza búsquedas de las sondas de monitorización utilizando mensajes ICMP y obtiene los datos sobre el hardware y el software por medio de mensajes SNMP.

- El gestor de monitorización está asociado a un sistema gestor de base de datos que proporciona el soporte para una base de datos en la cual se almacena información procedente de las sondas de monitorización tales como dirección IP, nombre, parámetros de configuración, imágenes asociadas, archivos de imágenes y sonidos capturados. Ver figura 1.8.

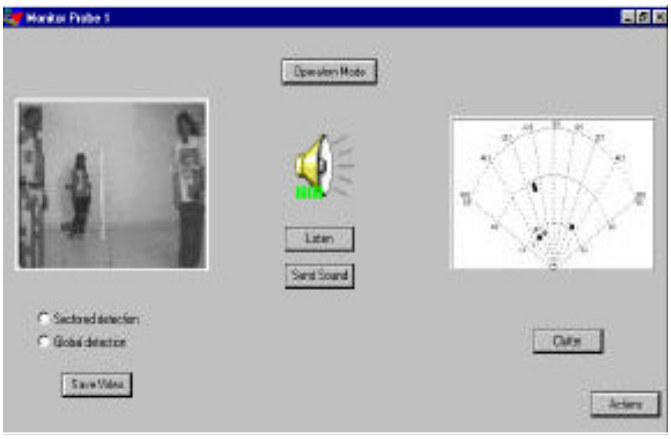


Figura 1.8. Pantalla del Gestor de Monitorización del sistema [DuJ+ 2000].

Otro trabajo cuyo contexto se encuentra dentro del estado de arte de esta tesis es el prototipo desarrollado por Azama Makishi y Huamán Huanca [AzA+]. Dicho prototipo es capaz de detectar automáticamente eventos remotos a partir de la adquisición de señales de video desde una red IP.

En el prototipo desarrollado en [AzA+] se utiliza la arquitectura cliente-servidor, en el cual el módulo cliente realiza la conexión remota con las cámaras IP o cámaras web (mediante la dirección IP de la PC a la que está conectada).

En el programa cliente también se realiza el procesamiento de las imágenes recibidas, con las cuales se detectarán remotamente los eventos ocurridos.

En [AzA+] la transmisión de imágenes la realiza únicamente el programa servidor, el cuál es instalado en la computadora que tiene conectada la cámara web.

El programa servidor captura cuadros (frames) de 160 x 120 píxeles de tamaño, ancho y alto respectivamente, en formato BMP. Luego estos frames se comprimen en formato JPEG, de esta forma sólo se transmiten los coeficientes generados al comprimir la imagen.

La transmisión se realiza enviando vectores de 128 bytes formados por los coeficientes JPEG mas 2 bytes de cabecera. En los bytes de cabecera se coloca la información del número de cámara y códigos de verificación para la correcta transmisión de los coeficientes (figura 1.9).

Las tramas de bytes recibidas por el usuario son reagrupadas para reconstruir los frames en formato JPEG. Además, la imagen se descomprime para llevarla nuevamente al formato BMP y procesarla en ese formato.

El prototipo desarrollado en [AzA+] necesita de una etapa de calibración en la cual, en el programa cliente, se seleccionan los objetos a ser detectados en la imagen.

La etapa de calibración consiste en capturar una imagen inicial en la cual se seleccionaran las regiones en las que se encuentran cada uno de los objetos que se desean detectar. La selección se realiza dando cuatro clics consecutivos para construir un cuadro que encierra al objeto deseado y posteriormente se le asigna un nombre lo identifique.

Para la detección de movimiento de los objetos determinados por el usuario se toma como variable principal el componente de luminancia de los píxeles.

En el prototipo se realiza el cálculo de posición, velocidad y dirección del objeto conociendo primeramente el centro de masa del objeto en movimiento, lo cual se hace a través de la binarización de la imagen en donde se contrastan dos frames donde el primero sirve como frame de referencia o patrón y se guarda justo antes de que el objeto ingrese a la imagen y el segundo frame constituye la escena actual.

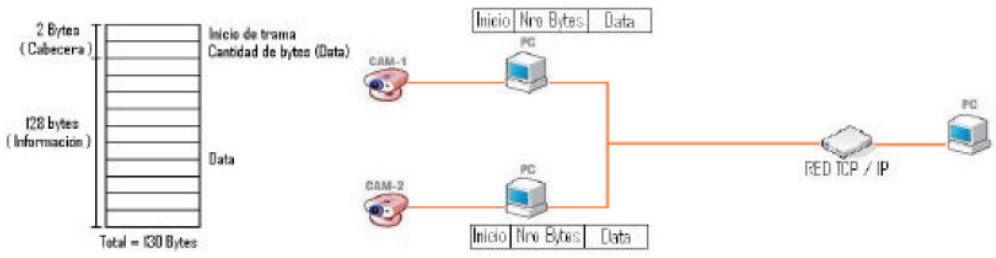


Figura 1.9. Conformación de la trama de datos transmitida [AzA+].

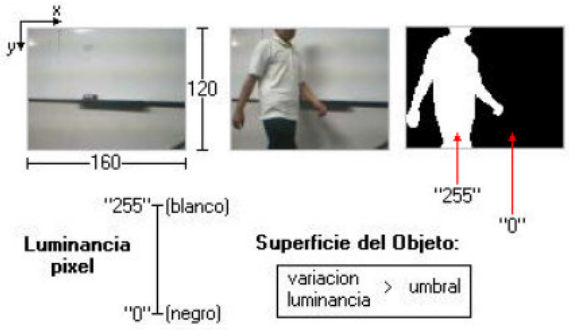


Figura 1.10. Binarización de imágenes en [AzA].

A los pixeles del frame actual que contrastan con el frame de referencia se les asigna un valor de luminancia de 255 y a los demás 0. Ver figura 1.10.

En el cálculo de la velocidad se necesita la distancia euclidiana entre dos centros de masa y una unidad de tiempo transcurrido la cuál es el periodo de la captura de frames. La velocidad de los objetos se muestra en píxeles/segundo.

Para obtener la dirección en la que se desplaza el objeto a través de la imagen, se calcula el ángulo hacia donde se mueve el centro de masa del objeto en cada instante de captura y de esta forma se define el cuadrante en que se encuentra.

En resumen, el prototipo aplica técnicas de procesamiento digital de video para detectar eventos en una determinada escena. Entre los eventos básicos considerados por el prototipo se encuentran:

- Detección de movimiento de objetos especificados previamente por el usuario.
- Dirección de desplazamiento de objetos especificados previamente por el usuario.
- Velocidad de objetos especificados previamente por el usuario.
- Alteración de la iluminación de la escena.
- Tamaño de imagen utilizada es de 160 x 120 píxeles de ancho y alto respectivamente.

Uno de los trabajos más interesantes es sin duda el sistema desarrollado por Barriuso [BaR 2004], el cual tiene por objetivo la detección de intrusos interconectado con una pasarela doméstica.

Una pasarela doméstica es un dispositivo encargado de interconectar la red doméstica con el exterior (a través de un proveedor de servicios de Internet), de gestionar y monitorizar remotamente aplicaciones y dispositivos domésticos; y de aplicar y actualizar el software que alberga, todo ello de forma transparente para el usuario.

El trabajo desarrollado en [BaR 2004] se centra en la pasarela doméstica OSGi (Open Services Gateway Initiative) que es un estándar definido por la compañía del mismo nombre y que hace posible crear aplicaciones que pueden ser desplegadas a través de la red y gestionadas remotamente sobre cualquier combinación hardware/software que soporte una maquina virtual Java.

En [BaR 2004] se describen tres módulos con los que cuenta el sistema (ver figura 1.11) el módulo de detección de intrusos, el módulo de centro de control y el módulo de dispositivo de usuario.

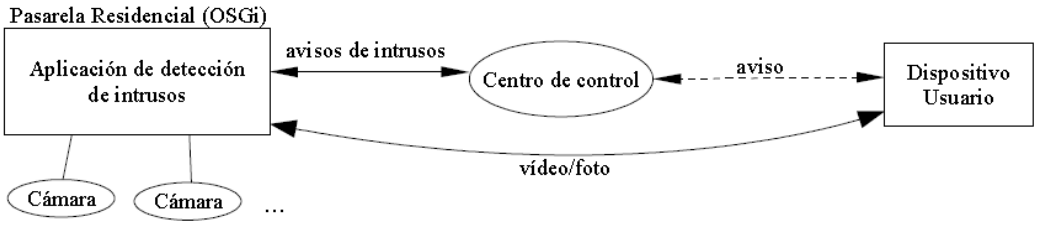


Figura 1.11. Esquema general de la aplicación de [BaR 2004].

El sistema se basa en procesar digitalmente, en el módulo de detección de intrusos, las imágenes recibidas por una o más cámaras de video (cámaras IP, web cam, cámaras WiFi) para detectar movimiento en las escenas.

En caso de que exista movimiento detectado en las escenas, el módulo emitirá una petición al centro de control para que a su vez el centro de control pueda enviar un aviso a un dispositivo móvil que el usuario llevará.

El aviso que el centro de control puede enviar hacia el dispositivo móvil puede ser una fotografía o secuencia de video de la escena donde se ha detectado el movimiento, datos sobre la fecha y hora de la intrusión.

El algoritmo que se utiliza para detectar el movimiento dentro de la secuencia de video se basa en la suma de las diferencias absolutas que existe entre todos y cada uno de los píxeles correspondientes que existen entre dos frames consecutivos.

No se realiza ninguna conversión a alguna paleta de colores por lo que el video se procesa sin compresión alguna y en la paleta de colores RGB, tomando como elemento representativo del píxel únicamente el componente R (rojo) de la paleta de colores.

En el sistema mencionado en [BaR 2004] no se realiza ningún tipo de seguimiento del movimiento detectado por el módulo de detección de intrusos.

Es importante mencionar que el módulo de centro de control puede recibir peticiones del dispositivo del usuario para enviar video bajo demanda, este video bajo demanda es solicitado al módulo de detección de intrusos el cual inicia la transmisión del video hacia el dispositivo del usuario.

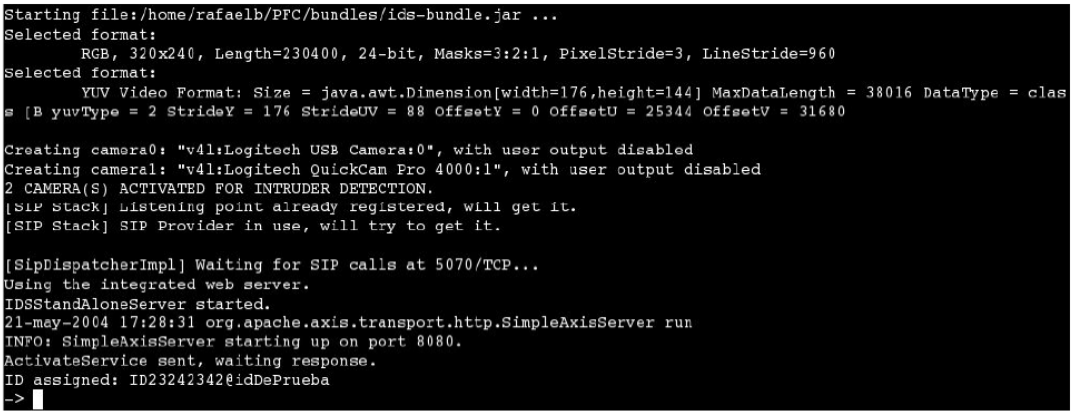


Figura 1.12. Captura de pantalla de operación del sistema de [BaR 2004].

En [BaR 2004] se indica que, no importando si se trata de un aviso de alarma o una petición de video bajo demanda, el video es transmitido utilizando el protocolo RTP (Protocolo de Tiempo Real para Transmisiones) y una compresión de video JPEG.

El sistema de [BaR 2004] se ha probó con simuladores y dentro de una red local, no se menciona el ancho de banda de la misma.

Por último debemos mencionar que las pruebas realizadas en [BaR 2004] fueron hechas con un tamaño de imagen de 320 x 240 píxeles, de ancho y alto respectivamente, a una tasa de adquisición de 10 frames por segundo, dichos valores fueron considerados por el autor de [BaR 2004] como estándar para una eventual implementación de su sistema.

1.2 Observaciones

Con la revisión del estado del arte mencionado en las páginas previas se encontraron las siguientes observaciones:

- Existe un abanico muy amplio de cámaras (web usb, inalámbricas, infrarojas, omnidireccionales, etc) que se pueden utilizar para los propósitos de detección y seguimiento de objetos en movimiento, las más utilizadas son cámaras capaces de captar y transmitir la imagen en la paleta de colores RGB.

- Se pueden considerar las dimensiones de imagen de 320 píxeles de ancho por 240 píxeles de alto como una resolución común entre los prototipos del estado de arte, ya que la mayoría de ellos realiza la detección de objetos utilizando imágenes de estas dimensiones a una velocidad de adquisición promedio de 10 frames por segundo.

- Los algoritmos de detección de movimiento utilizados en los prototipos mencionados fueron: los que realizan la comparación de diferencias entre dos frames “consecutivos” y los que realizan la comparación de diferencias que existen entre un determinado frame y un frame de referencia que contiene un modelo del fondo de escena (background).

- Algunos prototipos ofrecen la capacidad de calcular la velocidad a la que se desplaza el objeto en movimiento, así como la dirección en la que lo hace. Esto es logrado calculando el centroide del objeto en movimiento, lo cual es una operación opcional en el seguimiento de objetos en movimiento.

- Existen innovaciones importantes en algoritmos de clasificación de movimientos basados en la cantidad y posición de movimientos detectados de un objeto o persona, con lo cual se pueden obtener patrones de comportamiento o inferir conclusiones tomando como evidencia esas estadísticas.

- La mayoría de los prototipos tienen como objetivo la vigilancia de casas habitación, empresas o personas.

Además de las observaciones anteriores se concluyó que ningún sistema o prototipo computacional revisado, realiza la detección y seguimiento de movimiento remoto, en línea o en el menor tiempo posible. A continuación se listan las limitaciones que fueron encontradas:

- La detección y seguimiento del movimiento se realiza de manera local, es decir, ambos procesos se realizan en la misma computadora en donde se encuentra conectada la cámara de video digital.

- Aquellos prototipos que son capaces de transmitir secuencias de imágenes (fotografías o video) y mensajes de alerta por lo regular lo hacen a través de una conexión a Internet de banda ancha.

- En la mayoría de los casos la transmisión de video es bajo demanda, esto quiere decir que es el usuario del sistema el que verdaderamente inicia la transmisión de video por medio de una petición al sistema, lo cual elimina la posibilidad de una transmisión de video en tiempo real cuando un evento es detectado ya que el sistema no envía de manera automática el video cuando este ocurre.

- El seguimiento de los objetos en movimiento se realiza principalmente utilizando la binarización de las imágenes en las que se detecta el objeto en movimiento. Usualmente, para llevar a cabo esta binarización, se utilizan los operadores booleanos como son el operador XOR y el operador AND.

- No se encontró reportado en la literatura ningún prototipo que realice la detección y el seguimiento remoto de movimiento, ni bajo transmisión de video bajo demanda ni en tiempo real o cercano a él.

Así, con base en las limitaciones mencionadas anteriormente, este trabajo de tesis pretende cubrir, en parte, el hueco dejado por los prototipos que se enfocan en el procesamiento de imágenes de manera local.

Capítulo 2

Planteamiento del Esquema del Modelo

Al día de hoy se han desarrollado diversos prototipos de detección y seguimiento de movimiento, baste leer el Capítulo 1 de esta tesis, en los cuales se implementan diversos algoritmos tanto de detección como de seguimiento de movimiento.

En este capítulo se plantea una propuesta novedosa para solucionar algunos de los problemas que acarrea la detección y seguimiento de movimiento remoto.

El esquema que se plantea en este capítulo permite desarrollar aplicaciones de procesamiento de video a través de Internet y tiene como características principales: modularidad, adaptabilidad y sencillez de modificación. El esquema deberá permitir realizar detección y seguimiento de movimiento remoto, enfocándose principalmente al envío del video a través de Internet y su posterior procesamiento.

De allí que en este trabajo sea presentado un modelo de detección y seguimiento de movimiento remoto, además de que el modelo es probado mediante la implementación de un prototipo basado en dicho modelo.

El modelo propuesto, ver figura 2.1, está basado en la arquitectura Cliente-Servidor [MiC 2003], en donde el servidor será siempre la computadora que tiene conectada la cámara web y el cliente podría ser una computadora capaz de procesar video.

Para la detección y seguimiento de movimiento remoto, en el modelo planteado, es necesario que el análisis del video se lleve a cabo en dos computadoras diferentes.

El primer procesamiento es llevado a cabo en la computadora Servidor, la cual deberá de implementar un análisis de video para determinar si existen o no objetos en movimiento dentro de las escenas.

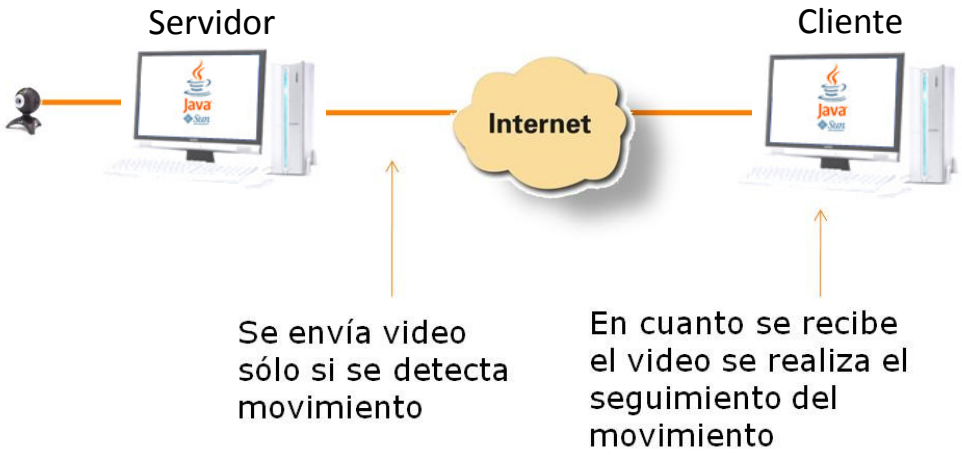


Figura 2.1. Modelo propuesto de detección y seguimiento de movimiento.

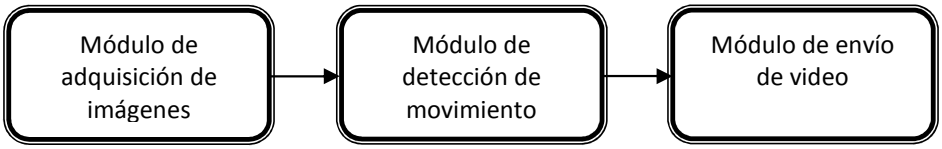


Figura 2.2. Diagrama a bloques de la aplicación servidor.

De manera inicial y más específicamente para la detección de movimiento se buscarán cambios temporales en la imagen, ya que cuando existen objetos en movimiento en una escena también existen cambios de intensidad de los píxeles en la imagen.

Para poder determinar dichos cambios se utiliza un algoritmo basado en [BrT 2003], el cual determina el porcentaje de variación total entre dos imágenes.

Posteriormente otro procesamiento es realizado en la computadora cliente, la cual deberá implementar un análisis de video para realizar el seguimiento de los objetos en movimiento dentro de las escenas.

El seguimiento del movimiento se realizará utilizando cuatro niveles de diferencia entre píxeles, dependiendo del nivel de diferencia que se presenten en píxeles correspondientes se le asignará un nuevo color a dicho píxel.

En las siguientes secciones se describen con más detalle los componentes del modelo desarrollado.

2.1 Aplicación servidor

Los bloques que componen a la aplicación servidor se pueden observar en el diagrama de la figura 2.2.

Dicho diagrama representa de manera general el proceso llevado a cabo por la aplicación servidor.

Cada una de las fases que se muestran en el diagrama a bloques, se describen a continuación.

2.1.1 Módulo de adquisición de imágenes

En esta fase, la tarea principal que se debe de llevar a cabo es la captación de las imágenes a través de la cámara web digital. En la figura 2.3 se muestra el diagrama a bloques de los procesos que se realizan dentro de este módulo, posteriormente se da una explicación general del mismo.

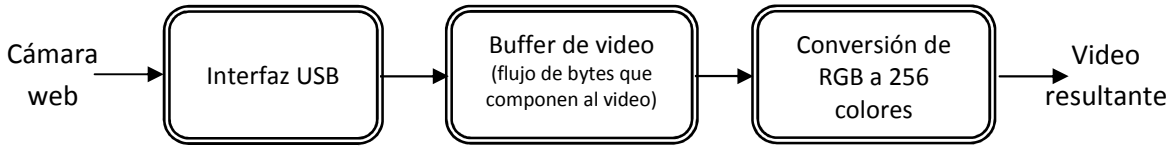


Figura 2.3. Diagrama a bloques del módulo de adquisición de imágenes.

Para que la captura de imágenes comience, la cámara debe ser conectada a la computadora utilizando la interfaz USB y posteriormente deberá activarse por medio del prototipo para que dé comienzo la transferencia de las imágenes captadas.

Las imágenes captadas por la cámara son digitalizadas como un arreglo bidimensional, en donde cada uno de los elementos contiene un arreglo unidimensional de tres elementos. Cada uno de estos elementos son las componentes Red, Green y Blue de cada píxel, que dependiendo del valor de cada componente pueden formar diferentes colores.

El espacio de color RGB puede ser visto como un cubo tridimensional teniendo como coordenadas a los colores R (rojo), G (verde) y B (azul) como se muestra en la figura 2.4.

La línea punteada que une los puntos (0,0,0) y (1,1,1) es la diagonal principal que representa todos los tonos de gris desde el color negro hasta el color blanco.

Usualmente se normalizan los valores RGB entre 0 y 1 para operaciones de punto flotante o se usa una representación de tipo byte que toma valores entre 0 y 255 para poder realizar operaciones con números enteros.

En este espacio de color, cada color es determinado por la suma de los valores de sus componentes rojo, verde y azul. Sin embargo, la principal desventaja de este espacio de color es que el tono de color no es independiente de la intensidad y saturación de color.

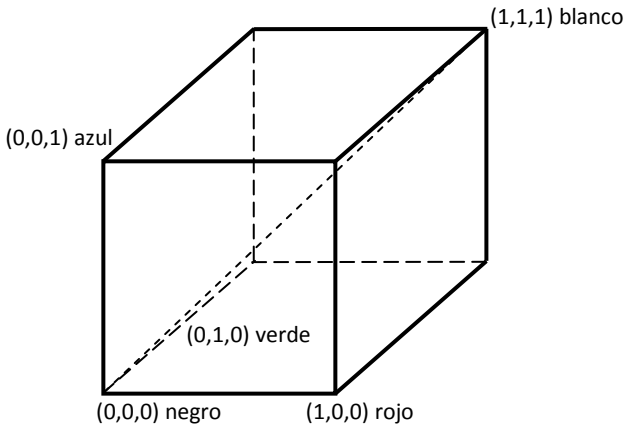


Figura 2.4. Cubo de color RGB.

La luminosidad L en el espacio de color RGB es definido como la suma de los tres componentes, como se muestra en la ecuación 2.1

$$L = R + G + B \tag{2.1}$$

Para el desarrollo del prototipo que se ha planteado, la desventaja mencionada anteriormente, no es limitante para el desempeño del mismo.

Utilizando el espacio de color RGB con la representación de tipo byte se obtienen 256 niveles de brillo para cada uno de los componentes R, G y B. De tal forma que obtendríamos 256^3 colores en formato RGB.

Sin embargo, para la detección de movimiento no es necesario utilizar en su totalidad el espacio de colores RGB ya que generaría un mayor número de operaciones y por consiguiente mayores requerimientos computacionales.

Para evitar la carga excesiva de operaciones computacionales, el prototipo debe ser optimizado comenzando con una conversión del espacio de colores RGB a un espacio de 256 colores en escala de grises, lo cual impactará en un menor número de operaciones computacionales.

La conversión entre los diferentes espacios de colores puede realizarse utilizando la componente de Intensidad (I) del espacio de colores HSI (Hue, Saturation, Intensity) la cual se obtiene calculando el promedio de los tres componentes RGB de cada uno de los píxeles [BrT 2003], como se muestra en la ecuación 2.2.

$$I(x,y) = \frac{R(x,y)+G(x,y)+B(x,y)}{3} \tag{2.2}$$

2.1.2 Módulo de detección de movimiento

En esta fase, la tarea principal es realizar el análisis de una secuencia de imágenes para determinar la presencia o no de objetos en movimiento. En la figura 2.5 se muestra el diagrama a bloques de los procesos que se realizan dentro de este módulo, posteriormente se da una explicación general del mismo.

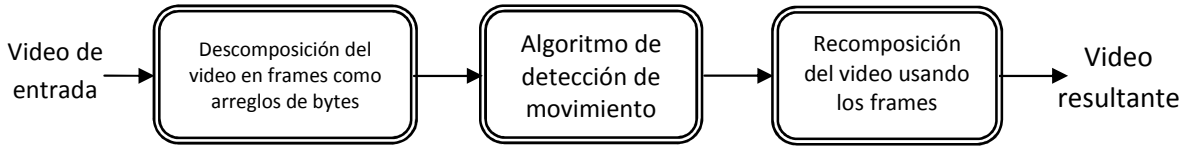


Figura 2.5. Diagrama a bloques del módulo de detección de movimiento.

La idea para un algoritmo básico de detección de movimiento es analizar, al menos, dos imágenes consecutivas y realizar los siguientes pasos [BrT 2003]:

1. Calcular el valor absoluto de la diferencia de todos los pares de píxeles correspondientes de dos imágenes consecutivas y sumarlos.
2. Calcular el porcentaje de diferencia que existe entre las dos imágenes.
3. Si el porcentaje de diferencia es mayor a un umbral (porcentaje de diferencia preestablecido), entonces se ha detectado movimiento.

Este método solo detecta movimiento en un par de imágenes pero no determina la dirección ni el área en donde es detectado. Sin embargo, el algoritmo puede ser extendido para detectar el movimiento separadamente, dividiendo la imagen en áreas y aplicando a cada una de ellas el algoritmo.

Es importante hacer notar que aunque se ha definido al par de imágenes como imágenes consecutivas, estas pueden ser captadas dejando una ventana de tiempo *t* entre cada una de ellas.

La figura 2.6 muestra el diagrama de flujo del algoritmo mencionado anteriormente, en el cual se puede apreciar que en el prototipo trataremos a la imagen como un arreglo unidimensional de bytes donde la longitud del arreglo dependerá directamente del tamaño de la imagen que deseemos procesar.

Aunque las imágenes captadas por la cámara web presentan ruido de tipo gaussiano [DoI 2006], el prototipo no implementa ningún tipo de algoritmo de mejora de imagen pero sí considera, y de hecho, calcula de manera automática, el umbral *U* de ruido que es considerado “normal” para el dispositivo de captura que esté usando.

Una vez que el umbral *U* es calculado, se realiza el cálculo del porcentaje de diferencia *P*, si $P > U$ entonces se determina que existe movimiento en las imágenes, en caso contrario, se vuelve a calcular *P* con nuevas imágenes y se vuele a hacer la comparación.

El prototipo ha sido diseñado para calcular, en tiempo de ejecución, el valor de umbral *U*. Esto le da la ventaja de poder utilizar diferentes dispositivos de captura ajustando de manera automática este parámetro sin hacer modificaciones en la implementación.

En la figura 2.7 se puede observar una secuencia de imágenes, en el espacio de 256 colores en escala de grises, generada por el prototipo a la cual se le puede aplicar el algoritmo de detección de movimiento descrito en esta sección.

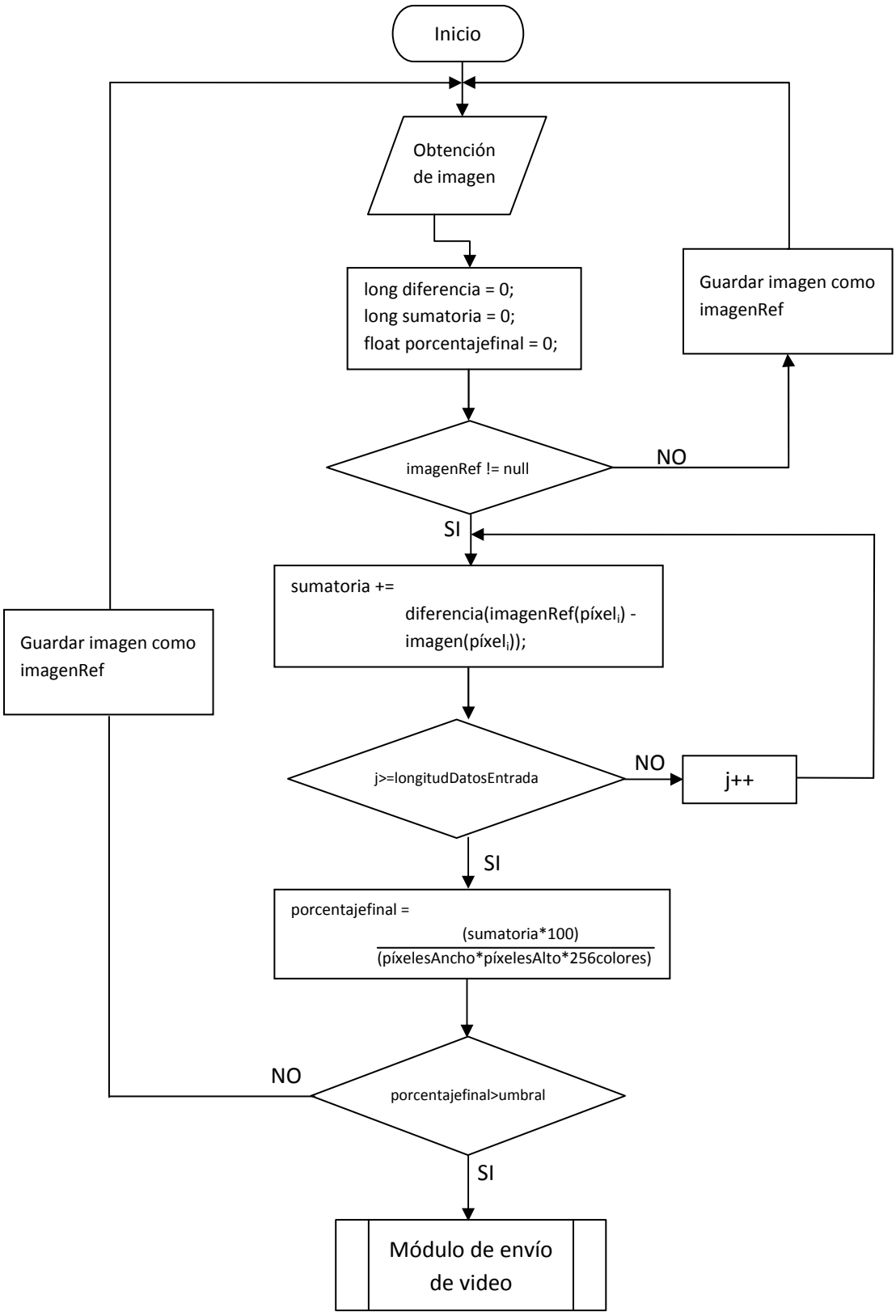


Figura 2.6. Diagrama de flujo del algoritmo de detección de movimiento del que hace uso el prototipo.

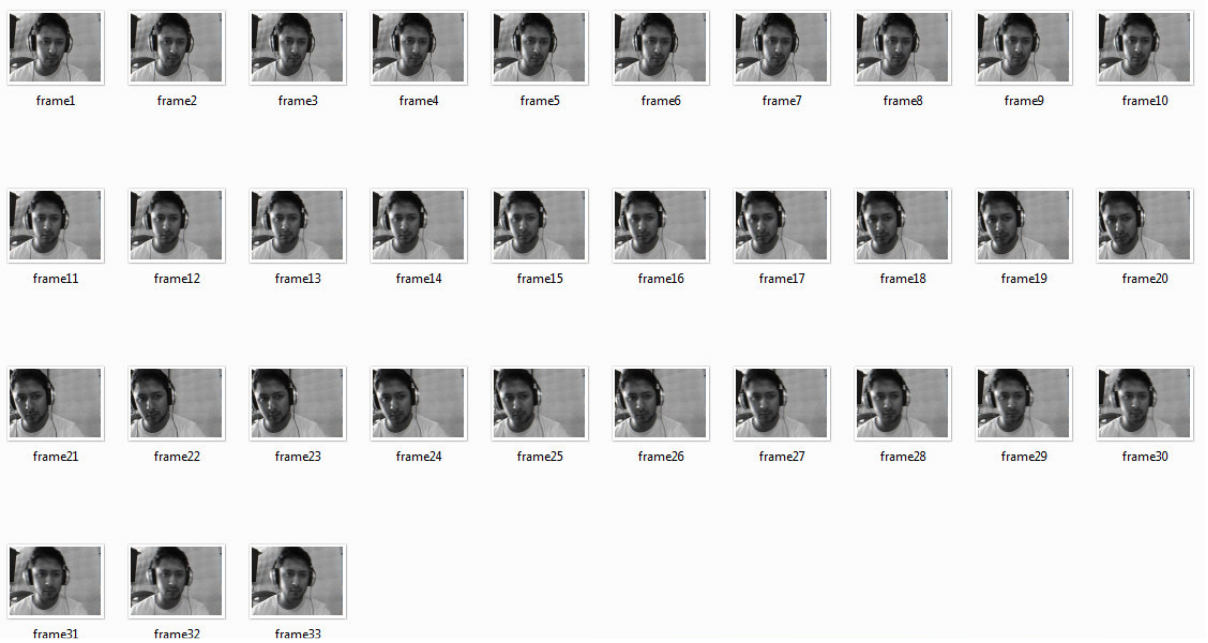


Figura 2.7. Ejemplo de secuencia de imágenes que captura y procesa el prototipo desarrollado.

2.1.3 Módulo de envío de video

En la figura 2.8 se muestra el diagrama a bloques de los procesos que se realizan dentro de este módulo, posteriormente se da una explicación general del mismo.

De manera abreviada en este módulo se llevan a cabo las tareas de

1. Ubicar a la computadora en la que reside la aplicación cliente,
2. Comprimir las secuencias de imágenes,
3. Enviar el video capturado a la computadora en donde reside la aplicación cliente.

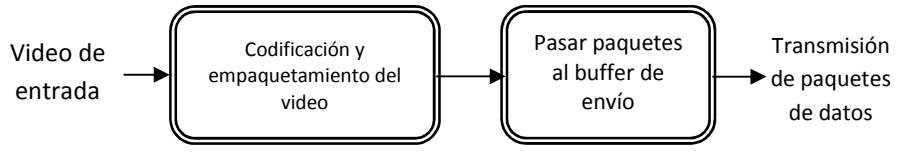


Figura 2.8. Diagrama a bloques del módulo de envío de video.

Para localizar a la computadora en la que reside la aplicación cliente es necesario proporcionar a la aplicación servidor una dirección IP y un puerto. De manera predeterminada el prototipo trabaja utilizando el puerto 12548.

Si las computadoras en donde residen las aplicaciones se encuentran detrás de algún firewall o ruteador, éstos deberán ser configurados para admitir y redireccionar el tráfico de datos generados por el prototipo utilizando el puerto y las direcciones IP.

Debido a que los recursos de los que hará uso el prototipo para el envío de video no se conocen, ni tampoco su capacidad y velocidad de transferencia de datos; se hace necesaria la incorporación de métodos de compresión para reducir al máximo la cantidad de datos que se deben transmitir y de esta forma optimizar el uso del canal de comunicación.

Dependiendo del entorno en el que se utilice la aplicación, el prototipo se ha diseñado tomando en cuenta las velocidades de las redes más comúnmente utilizadas, por ejemplo: Ethernet, conexión a Internet de tipo ADSL (1024 kbps entrantes y 128 kbps salientes) y de tipo RTC (Red Telefónica Conmutada, modem de 56 kbps), etc.

Por lo cual se han considerado dos métodos de compresión de video, JPEG y H.263, que requieren gran ancho de banda y poco ancho de banda, respectivamente.

En la tabla 2.1 podemos observar una comparación entre los diferentes formatos de compresión de video, en la tabla 2.1 se pueden apreciar tres parámetros: calidad, requerimientos de CPU para presentar el video en el receptor y requerimientos de ancho de banda.

En la primera versión del prototipo sólo se ha diseñado para la transmisión de video de tipo *unicast*, que se refiere a la transmisión de datos desde un único emisor a un único receptor, siendo posible la implementación de tipo *multicast*, que se refiere a la transmisión de datos desde un único emisor a múltiples destinatarios específicos, o *broadcast*, que se refiere a la transmisión de datos a todas las direcciones de la red.

Después de una investigación realizada sobre las tecnologías de transmisión de datos en tiempo real [HeE 2000], se decidió que el protocolo RTP (Real Time Transport Protocol, ver capítulo 3 sección 3.2.3) es el mejor candidato para el envío de video, por lo cual el prototipo ha sido diseñado para hacer uso de él.

El empaquetamiento de los datos, así como las tareas derivadas de la transmisión del video son delegadas al protocolo seleccionado.

Formato	*Tipo Contenido*	*Calidad*	*Requerimientos de CPU*	*Requerimientos de Ancho de Banda*
Cinepak	AVI QuickTime	Media	Bajos	Altos
MPEG-1	MPEG	Alta	Altos	Altos
H.261	AVI RTP	Baja	Medianos	Medianos
H.263	QuickTime AVI RTP	Media	Medianos	Bajos
JPEG	QuickTime AVI RTP	Alta	Altos	Altos
Indeo	QuickTime AVI	Media	Medianos	Medianos

Tabla 2.1. Formatos comunes de compresión de video [JMF 1999].

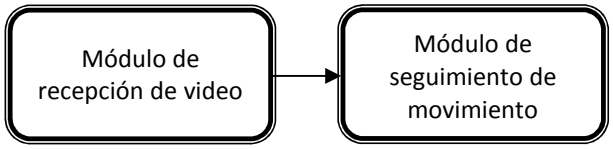


Figura 2.9. Diagrama a bloques de la aplicación cliente.

2.2 Aplicación cliente

El diagrama de bloques que componen a la aplicación cliente se puede observar en la figura 2.9.

Dicho diagrama representa de manera general el proceso llevado a cabo por la aplicación cliente.

Cada una de las fases que se muestran en el diagrama a bloques se describen a continuación.

2.2.1 Módulo de recepción de video

La recepción de video se encuentra ligada a dos componentes:

1. Tipo de compresión del video recibido.
2. Tipo de protocolo de transporte utilizado para el envío de datos (TCP/IP o UDP) [SiC+ 2003].

En la figura 2.10 se muestra el diagrama a bloques de los procesos que se realizan dentro de este módulo, posteriormente se da una explicación general del mismo.

De tal forma que en este módulo, de manera abreviada, se llevan a cabo las siguientes tareas:

1. Ubicar a la computadora en la que reside la aplicación servidor,
2. Esperar a que la aplicación servidor comience el envío de paquetes de datos,
3. Recibir y ensamblar los paquetes de datos que envía la aplicación servidor, y
4. Reconstruir el video recibido y enviarlo al *códec* de procesamiento de video (en el prototipo se diseño el *códec* de seguimiento de movimiento).

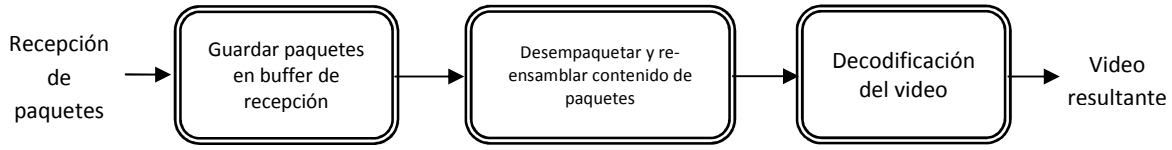


Figura 2.10. Diagrama a bloques del módulo de recepción de video.

Para localizar a la computadora en donde reside la aplicación servidor, la aplicación cliente debe “apuntar” usando el módulo de recepción de video al cual el usuario le proporcionará una dirección IP.

Una vez que se ha ubicado a la computadora en donde reside la aplicación servidor, la aplicación cliente entra a un estado de espera en el que permanece hasta que la aplicación servidor comience el envío de datos.

Cuando se comienzan a recibir los datos provenientes de la aplicación servidor éstos deben de ser desempaquetados y re-ensamblados en la aplicación cliente para que, posteriormente, el video reconstruido pueda ser mostrado o pasado al *códec* de seguimiento o a cualquier otro códec o serie de *codecs* de procesamiento.

Todas las tareas relacionadas con la recepción, ensamblado y reconstrucción del video son delegadas al protocolo seleccionado.

2.2.2 Módulo de seguimiento de movimiento

Éste módulo tiene como primera fase, la detección del movimiento y después la aplicación de algún método de resaltado o enmarcado del mismo. En la figura 2.11 se muestra el diagrama a bloques de los procesos que se realizan dentro de este módulo, posteriormente se da una explicación general del mismo.

Tomando en cuenta lo anterior, para el módulo descrito en esta sección, se ha diseñado una variante del algoritmo de detección de movimiento que utiliza la aplicación servidor, la diferencia entre estos algoritmos es su precisión.

Dado que en la aplicación servidor el módulo de detección de movimiento es el encargado de detectar el movimiento, el algoritmo del que hace uso debe ser lo más sensible posible, por lo que lleva las imágenes desde el espacio de colores RGB al espacio de 256 colores en escala de grises y después realiza el procesamiento.

En cambio, la aplicación cliente necesita detectar el movimiento únicamente para llevar a cabo el seguimiento del movimiento, por lo que puede hacer uso de un algoritmo de detección de igual o menor robustez que el implementado en la aplicación servidor.

El algoritmo que se diseñó para este módulo de seguimiento de movimiento no fue concebido para ser demasiado sensible y lleva las imágenes recibidas desde el espacio de colores RGB al espacio de 16 colores en escala de grises y después realiza la comparación píxel a píxel para poder obtener un porcentaje de movimiento significante.

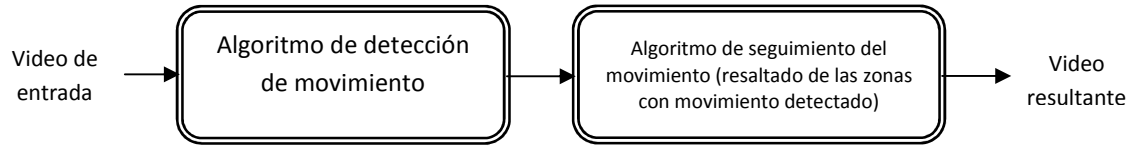


Figura 2.11. Diagrama a bloques del módulo de seguimiento de movimiento.

Si el porcentaje de movimiento significante es mayor a un umbral preestablecido, entonces se da paso a la fase de seguimiento de movimiento, de lo contrario se decide que no hay movimiento al cual realizar el seguimiento.

Para realizar el seguimiento de objetos en movimiento pueden ser utilizadas operaciones booleanas, por ejemplo las operaciones AND y XOR. Ambos operadores nos sirven para generar una nueva imagen haciendo una comparación píxel a píxel entre otras dos imágenes.

Sin embargo, para el prototipo, se ha diseñado un algoritmo de seguimiento de movimiento basado en la idea del algoritmo presentado en [BrT 2003], el cual es denominado asignación estática de color de clase, el cual es comúnmente usado en la etapa segmentación de imagen para la identificación de objetos independientes presentes en una imagen.

En el diseño del prototipo no se consideró ninguna etapa de segmentación de imagen, ya que sólo interesa el seguimiento del movimiento presente en una secuencia de imágenes sin identificar a qué objeto pertenece el movimiento detectado.

Se ha establecido que el prototipo necesita distinguir solo cuatro niveles de evidencia de cambio en la intensidad de los píxeles.

Las clases de colores se refieren a la cantidad de movimiento presentado por un píxel, es decir, el valor de la diferencia absoluta de intensidad entre píxeles correspondientes en imágenes consecutivas.

Las cuatro clases de colores que se han definido son: rojo para identificar una diferencia muy grande en el cambio de intensidad de los píxeles, naranja para identificar una diferencia media alta en el cambio de intensidad de los píxeles, amarillo oscuro para identificar una diferencia media baja en el cambio de intensidad de los píxeles, y amarillo claro para identificar una diferencia baja en el cambio de intensidad de los píxeles.

Es posible distinguir a cada uno de los colores rojo, naranja, amarillo oscuro, amarillo claro por medio de una combinación de umbrales previamente definidos:

$$\text{Clase de color} = \begin{cases} \text{rojo} & \text{if } diffValue >= threshold_1 \\ \text{naranja} & \text{if } threshold_1 > diffValue >= threshold_2 \\ \text{amarillo oscuro} & \text{if } threshold_2 > diffValue >= threshold_3 \\ \text{amarillo claro} & \text{if } threshold_3 > diffValue >= threshold_4 \end{cases}$$

Los valores de los umbrales pueden ser establecidos de manera arbitraria, sin embargo dichos valores deberán cumplir con la siguiente condición:

$$threshold_1 > threshold_2 > threshold_3 > threshold_4$$

Dependiendo de los valores elegidos para cada uno de los umbrales, podría ser necesario poner en práctica un procedimiento de prueba y error para poder verificar que los valores elegidos para cada uno de los umbrales proporcionan un comportamiento aceptable.

En la figura 2.12 puede observarse una secuencia de imágenes a la cual se ha aplicado el método de seguimiento descrito anteriormente, tanto la regla como el sujeto se encuentran en movimiento. En esta figura es posible apreciar los resultados del método de seguimiento.

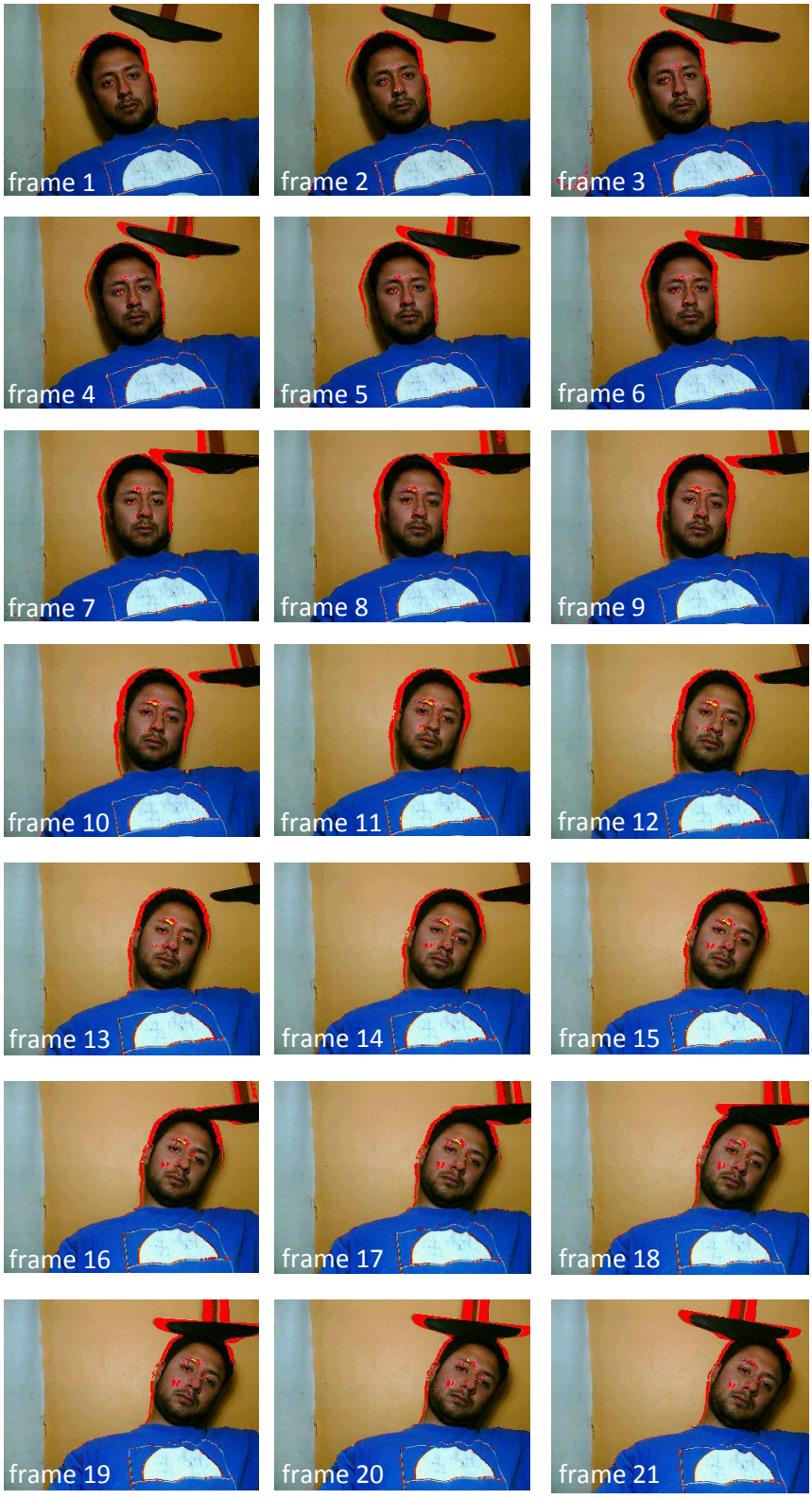


Figura 2.12. Secuencia de imágenes aplicando seguimiento.

Capítulo 3

Diseño e Implementación del Prototipo

3.1 Diseño del sistema

El paradigma de la programación orientada a objetos ha sido considerado como la mejor metodología aplicable para el diseño del modelo y del prototipo que se desarrolla en esta tesis.

Los objetos son entidades que combinan estado, comportamiento e identidad:

- El estado está compuesto de datos a los cuales se les denominan atributos a los cuales se les asignan valores concretos.
- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La identidad es una propiedad de un objeto que lo diferencia del resto, dicho en otras palabras, es su identificador.

En la programación orientada a objetos se expresa un programa como un conjunto de objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

Todos los objetos son creados a partir de las definiciones de clases, las cuales establecen las propiedades y comportamientos de un tipo de objeto concreto. La instanciación de un objeto es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Para mayor referencia de la programación orientada a objetos ver [EcB 2003].

El lenguaje de modelado más utilizado para las aplicaciones que se desarrollan siguiendo el paradigma de la programación orientada a objetos es el Lenguaje Unificado de Modelado (UML por sus siglas en inglés).

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos tales como procesos del negocio y funciones del sistema.

En pocas palabras, UML es el lenguaje en el que está descrito el modelo del sistema. UML cuenta con varios tipos de diagramas los cuales muestran diversos aspectos de las entidades representadas.

En UML 2.0 (que es la versión en la se basa el modelo presentado en esta tesis) existen 13 diferentes tipos de diagramas clasificados en tres categorías [WIKI 2008]:

- **Diagramas de Estructura**: enfatizan en los elementos que deben existir en el sistema modelado.
    - *Diagrama de clases*
    - *Diagrama de componentes*
    - *Diagrama de objetos*
    - *Diagrama de estructura compuesta*
    - *Diagrama de despliegue*
    - *Diagrama de paquetes*

- **Diagramas de Comportamiento**: enfatizan en lo que debe suceder en el sistema modelado.
    - *Diagrama de actividades*
    - *Diagrama de casos de uso*
    - *Diagrama de estados*

- **Diagramas de Interacción**: enfatizan sobre el flujo de control y de datos entre los elementos del sistema modelado.
    - *Diagrama de secuencia*
    - *Diagrama de colaboración*
    - *Diagrama de tiempos*
    - *Diagrama de vista de interacción*

Los diagramas UML siguen también una jerarquía que se muestra en la figura 3.1.

La explicación de cada uno de los diagramas mencionados anteriormente rebaza el objetivo de esta tesis por lo que sólo se mencionan como referencia para el lector.

Para resaltar la modularidad, flexibilidad y facilidad de modificación de la arquitectura propuesta, en las siguientes secciones se incluyen únicamente los diagramas de clase y de paquetes tanto de la aplicación servidor como de la aplicación cliente.

Adicionalmente se incluye el diagrama de despliegue del prototipo desarrollado.

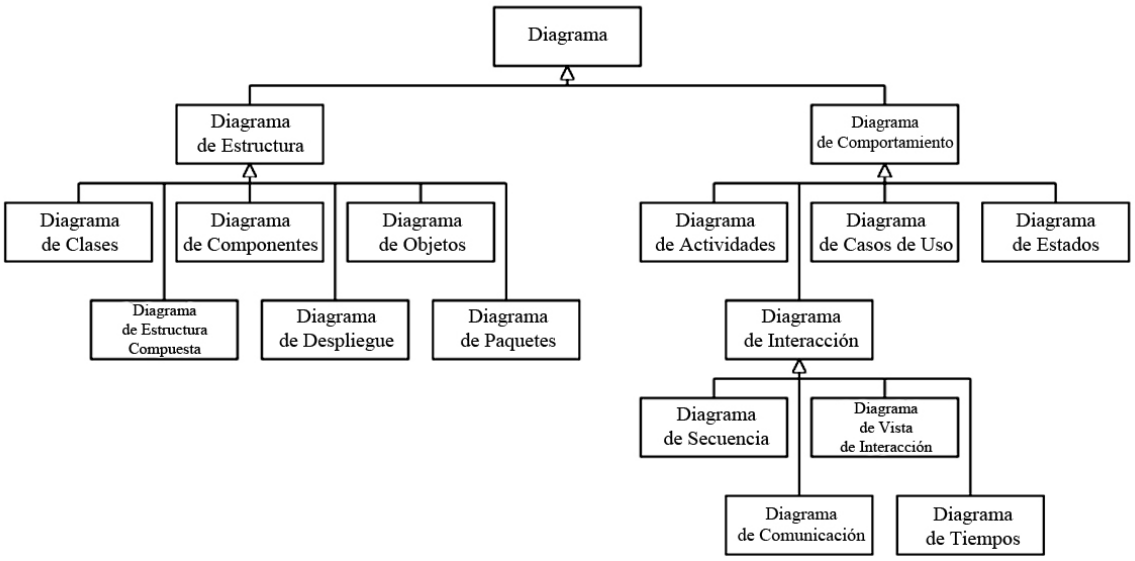


Figura 3.1. Jerarquía de los diagramas UML 2.0 [WIKI 2008].

3.1.1 Diagrama de clases

En UML, el diagrama de clases es un tipo de diagrama estático que describe la arquitectura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro [WIKI 2008].

3.1.1.1 Aplicación servidor

Las clases propias del prototipo que componen a la aplicación servidor son:

- **GUIConfiguraciónVideoAlgoritmo:** en esta clase está definido el método que habrá de arrancar a toda la aplicación además de implementar elementos gráficos para interactuar con el usuario del prototipo y poder obtener parámetros de configuración con los cuales operará el prototipo, una vez obtenidos los parámetros se crea un objeto anónimo del tipo GUIDetectorMovimiento.

- **GUIDetectorMovimiento:** esta clase implementa elementos gráficos para que el usuario pueda controlar las funciones del prototipo, cambiar parámetros. Se encarga de instanciar a un objeto de la clase DetectorMovimiento, que es la clase central del prototipo.

- **DetectorMovimiento:** es la clase central del prototipo, por lo que el objeto instanciado de esta clase estará presente durante todo el tiempo de ejecución del prototipo. Es la clase encargada de iniciar la captura de video, enviarlo a la clase que lo procesa y mostrarlo en el monitor. Aquí se implementó la lógica del negocio que consiste, por ejemplo, en activar la alarma, decidir si el video debe ser enviado a la aplicación cliente o no, etc.

- **CodecMovimiento:** es la clase especializada en el procesamiento del video, aquí se implementó el códec de detección de movimiento, en otras palabras, en esta clase se implementó el algoritmo de detección de movimiento del que hará uso el prototipo. Se diseño esta clase para que admita únicamente tamaño de video de 320x240 píxeles de ancho y alto, respectivamente.

- **CodecMovimiento640x480:** es la clase especializada en el procesamiento del video, aquí se implementó el códec de detección de movimiento, en otras palabras, en esta clase se implementó el algoritmo de detección de movimiento del que hará uso el prototipo. Se diseño esta clase para que admita únicamente tamaño de video de 640x480 píxeles de ancho y alto, respectivamente. El código de la clase CodecMovimiento fue reutilizado en esta clase y solo se modificaron algunos elementos.

- **EnviarVideo:** es la clase especializada en el envío de video hacia la aplicación cliente, aquí se implementó la utilización del tipo de protocolo para el envío de video, así como la selección, en base a lo que el usuario haya especificado, del tipo de compresión que se empleará en el envío de video.

- **CfgPrm:** esta clase contiene parámetros de configuración adicional que pueden ser utilizados por el prototipo en futuras versiones.

En la figura 3.2 es posible observar un diagrama simplificado de las clases que componen a la aplicación servidor. Para poder observar un diagrama detallado se puede consultar el Apéndice A.

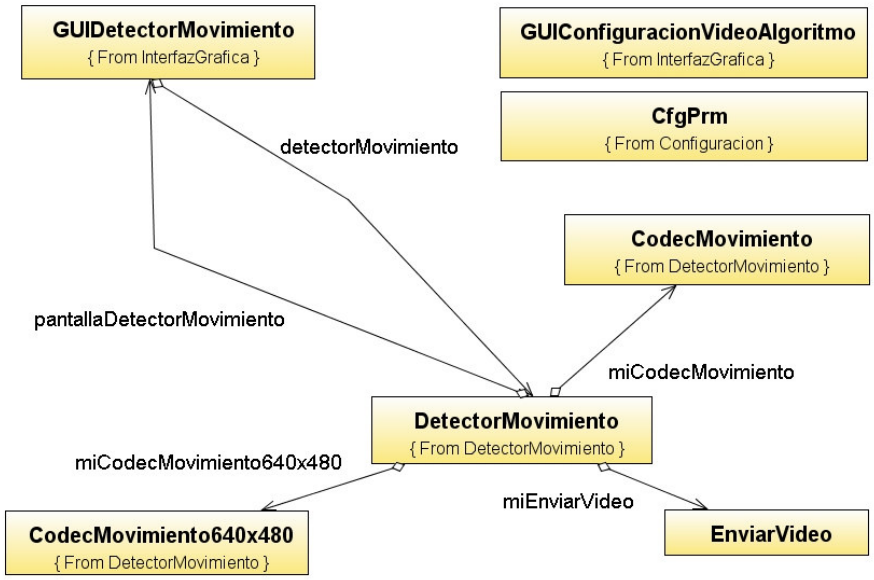


Figura 3.2. Diagrama de clases simplificado de la aplicación servidor.

3.1.1.2 Aplicación cliente

Las clases propias del prototipo que componen a la aplicación cliente son:

- **GUIConfigurarRecibirVideo:** en esta clase está definido el método que habrá de arrancar a toda la aplicación además de implementar elementos gráficos para interactuar con el usuario del prototipo y poder obtener la dirección IP de la computadora donde reside la aplicación servidor, después de proporcionar la dirección IP, la aplicación cliente entra en el estado de "espera" y cuando recibe datos, instancia un objeto de tipo GUIRecibirVideo.

- **GUIRecibirVideo:** esta clase implementa elementos gráficos para que el usuario pueda controlar las funciones del prototipo, cambiar parámetros y mostrar el video recibido. Se encarga de instanciar a un objeto de la clase RecibirVideo, que es la clase central del prototipo.

- **RecibirVideo:** el objeto instanciado de esta clase estará presente durante todo el tiempo de ejecución del prototipo. Es la clase encargada de recibir el video, enviarlo a la clase que lo procesa y mostrarlo en el monitor.

- **CodecSeguimiento:** es la clase especializada en el procesamiento del video, aquí se implementó el códec de seguimiento de movimiento, en otras palabras, en esta clase se implementó el algoritmo de seguimiento de movimiento del que hará uso el prototipo. Se diseño esta clase para que admita cualquier tamaño de video recibido.

- **CfgPrm**: esta clase contiene parámetros de configuración adicional que pueden ser utilizados por el prototipo en futuras versiones.

En la figura 3.3 es posible observar un diagrama simplificado de las clases que componen a la aplicación cliente. Para poder observar un diagrama detallado se puede consultar el Apéndice A.

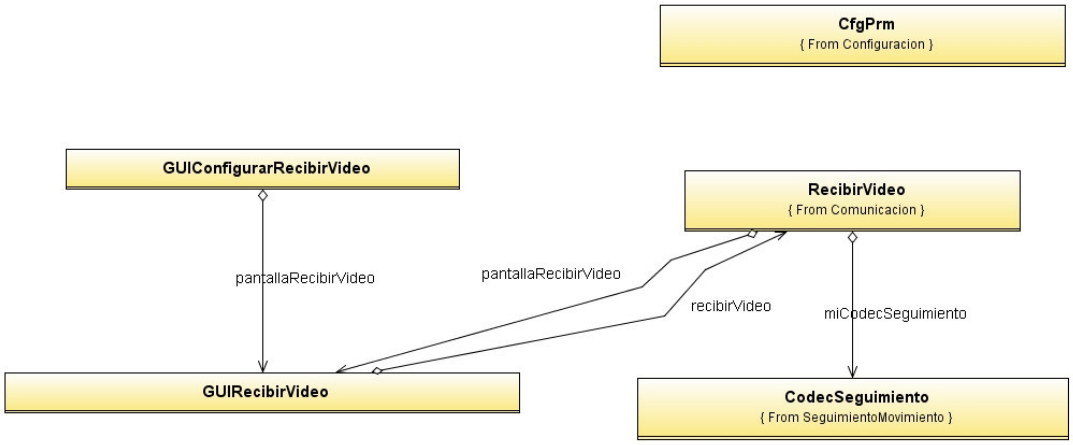


Figura 3.3. Diagrama de clases simplificado de la aplicación cliente.

3.1.2 Diagrama de paquetes

En UML, el diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas, mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquete suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes [WIKI 2008].

3.1.2.1 Aplicación servidor

Los paquetes que componen a la aplicación servidor son:

- **InterfazGrafica:** este paquete contiene las clases GUIConfiguracionVideoAlgoritmo y GUIDetectorMovimiento.
- **DetectorMovimiento:** este paquete contiene las clases DetectorMovimiento, CodecMovimiento y CodecMovimiento 640x480.
- **Comunicacion:** este paquete contiene la clase EnviarVideo.
- **Configuracion:** este paquete contiene la clase CfgPrm.

En la figura 3.4 puede observarse el diagrama de paquetes simplificado de la aplicación servidor. Para poder observar un diagrama detallado se puede consultar el Apéndice B.

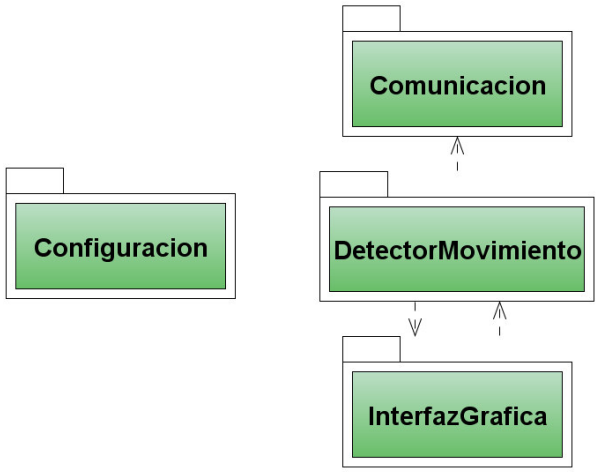


Figura 3.4. Diagrama de paquetes simplificado de la aplicación servidor.

3.1.2.2 Aplicación cliente

Los paquetes que componen a la aplicación cliente son:

- **InterfazGrafica:** este paquete contiene las clases GUIConfigurarRecibirVideo y GUIRecibirVideo.
- **SeguimientoMovimiento:** este paquete contiene las clases CodecSeguimiento.
- **Comunicacion:** este paquete contiene la clase RecibirVideo.
- **Configuracion:** este paquete contiene la clase CfgPrm.

En la figura 3.5 puede observarse el diagrama de paquetes simplificado de la aplicación cliente. Para poder observar un diagrama detallado se puede consultar el Apéndice B.

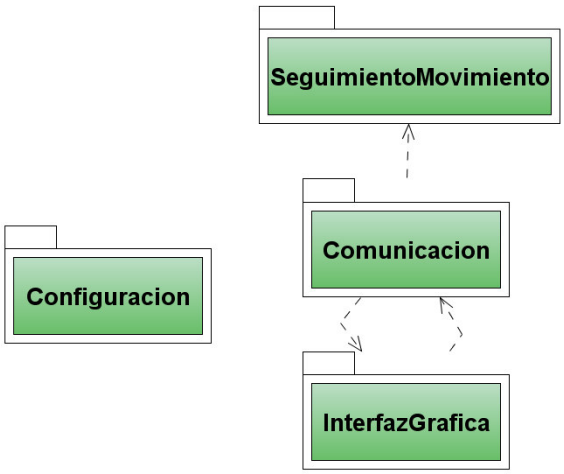


Figura 3.5. Diagrama de paquetes simplificado de la aplicación cliente.

3.1.3 Diagrama de despliegue

En UML los diagramas de despliegue se utilizan para modelar el hardware utilizado en la implementación de sistemas y las relaciones entre sus componentes. Los elementos usados en este tipo de diagrama son nodos, componentes y asociaciones [WIKI 2008].

En la figura 3.6 puede observarse el diagrama de despliegue del prototipo propuesto en esta tesis.

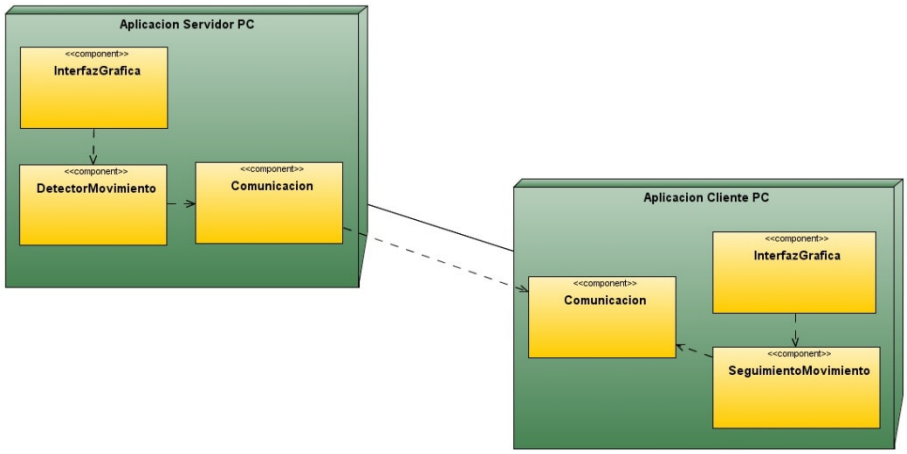


Figura 3.6. Diagrama de despliegue del prototipo.

3.2 Implementación del sistema

El desarrollo del prototipo (implementación) debe ser congruente con la metodología de diseño planteada anteriormente, de tal manera que se eligió un lenguaje de programación orientado a objetos.

Se optó por el lenguaje de programación Java para llevar a cabo la implementación computacional del prototipo.

3.2.1 ¿Por qué utilizar Java?

Java se convierte día a día en uno de los lenguajes de programación más extendidos, es decir, su uso ya no se limita únicamente a la programación de soluciones en Internet sino que se utiliza cada vez más como herramienta de programación orientada a objetos y también para el desarrollo de nuevas aplicaciones tanto científicas como de uso doméstico.

A continuación se mencionan algunas de las características de Java que lo hacen ideal para el desarrollo del prototipo:

- Es una tecnología de uso libre, el desarrollo del prototipo en este lenguaje de programación tiene un bajo costo de implementación.
- Es un lenguaje de programación orientado a objetos, esto permitirá la reutilización del prototipo para trabajos futuros.
- Java es multiplataforma, lo cual en teoría, permitirá utilizar el prototipo en otros sistemas operativos tales como Linux, Solaris, MAC OS además de otras versiones de Windows.
- Con más de 10 años en el mercado, Java se ha posicionado como uno de los lenguajes de programación más robustos, adaptables y preferidos, tanto en la comunidad científica como en el ambiente comercial, lo que le brinda al proyecto tintes de modernidad y prevalencia.
- El tiempo de aprendizaje para programar en lenguaje Java es relativamente corto si se cuenta con conocimientos previos de lógica de programación y lenguaje C, lo cual permitiría que en un futuro, personas interesadas en el prototipo puedan continuar su desarrollo con relativa facilidad.

Sin embargo, Java sigue siendo un lenguaje interpretado lo cual, en cierta manera, podría afectar el rendimiento del prototipo, aunque debe tomarse en cuenta el avance en la capacidad de procesamiento que presentan las computadoras personales actuales motivo por el cual, el hecho de ser un lenguaje interpretado es despreciable.

3.2.2 El entorno de trabajo JMF 2.1.1.e

El JMF o Java Media Framework (en español, Entorno de Trabajo Multimedia de Java) es una expansión de Java que permite la programación de tareas multimedia en este lenguaje de programación.

En otras palabras, JMF es un conjunto de clases (API) que permite añadir audio y video a aplicaciones y applets construidos en la tecnología Java. Es un paquete opcional ya que no se incluye con la JDK ni el JRE sino que debe conseguirse como un paquete externo [JMF 1999].

Este paquete (JMF) puede capturar, reproducir, transmitir y cambiar el tipo de codificación de múltiples formatos de medios. Algunas de sus características más importantes son:

- Estabilidad, debida a que funciona sobre la máquina virtual de Java (JVM).
- Sencillez, ya que permite, usando pocos comandos, realizar complejas tareas multimedia.
- Potencialidad, permitiendo la manipulación de elementos multimedia de video y audio locales, así como la retransmisión en tiempo real de video y audio a través de la red mediante el protocolo RTP.
- Tecnología multiplataforma.

3.2.3 El protocolo RTP

Transmitir datos multimedia a través de una red requiere de una red de gran rendimiento. En este tipo de redes es fácil compensar la pérdida de paquetes de datos y largos retrasos en la recepción de los mismos.

La transmisión de los datos es muy diferente en comparación con al acceso de datos estáticos tales como archivos en donde los más importante es que los datos lleguen a su destino. Consecutivamente, los protocolos usados para datos estáticos no funcionan bien para el envío de flujo de datos.

Los protocolos HTTP y FTP están basados en el protocolo TCP (Transmission Control Protocol). TCP es un protocolo de la capa de transporte del modelo OSI [StW 2004] diseñado para la comunicación confiable de datos en redes de bajo ancho de banda y alta tasa de errores. Cuando un paquete es perdido o corrompido éste es retransmitido, por lo que el tiempo para garantizar la transferencia fiable de datos hace lenta la tasa total de transmisión.

Por esta razón, protocolos subyacentes distintos de TCP son típicamente utilizados para el envío de datos multimedia. Entre ellos destaca el uso del protocolo UDP (User Datagram Protocol).

UDP es un protocolo poco confiable debido a que no garantiza que cada paquete alcance su destino, además de que no garantiza que los paquetes llegarán en el orden en que fueron enviados. Por lo que el receptor debe ser capaz de compensar la pérdida de datos, los paquetes duplicados y los paquetes que lleguen fuera de orden.

Al igual que TCP, UDP es un protocolo de la capa de transporte sobre el cual se construyen a su vez otros protocolos específicos para alguna aplicación, como es el caso del protocolo RTP.

RTP son las siglas de Real-time Transport Protocol (Protocolo de Transporte de Tiempo real). El protocolo RTP es utilizado para la transmisión de información en tiempo real, por ejemplo audio y video en una videoconferencia. [WIKI 2007].

Ha sido desarrollado por el grupo de trabajo de transporte de Audio y Video del IETF (Internet Engineering Task Force), fue publicado por primera vez como estándar en el año de 1996 y posteriormente actualizado en el año de 2003, actualmente es usado en sistemas de *streaming*, video conferencias y sistemas de comunicaciones VoIP.

El objetivo de RTP es brindar un medio uniforme de transmisión sobre IP de datos que estén sujetos a las limitaciones de tiempo real (audio, video, etc.). La función principal de RTP es implementar los números de secuencia de paquetes IP para rearmar la información de voz o de video, incluso cuando la red subyacente cambie el orden de los paquetes. [KIOS 2008]

De manera más general, RTP permite:

- Identificar el tipo de información transmitida;
- Agregarle marcadores temporales y números de secuencia a la información transmitida;
- Controlar la llegada de los paquetes al destino.
- Además, los paquetes de difusión múltiple pueden utilizar RTP para enrutar conversaciones a múltiples destinatarios.

RTP permite la administración de flujos multimedia (voz, video) sobre IP. RTP funciona sobre el protocolo UDP añadiendo el encabezado RTP que lleva información de sincronización y numeración. [KIOS 2008].

Para mayor información sobre el protocolo RTP, consultar la referencia [WIKI 2007].

3.2.4 Formatos manejados para la transmisión del video

Como se ha mencionado anteriormente, la transmisión de datos entre la aplicación servidor y la aplicación cliente se realiza utilizando el protocolo RTP.

Sin embargo, el protocolo RTP solo se encarga de la transmisión de los datos y no de la codificación o formato del dato. Es posible especificar el tipo de compresión de los datos de medios antes de que los datos sean pasados al protocolo RTP para su transmisión.

Al establecer el tipo de formato en el cual serán enviados los datos, se generan nuevos encabezados que formarán parte de los datos que transporta el paquete RTP en su carga útil.

En las siguientes secciones se describen brevemente los dos formatos utilizados para envió de video entre las aplicaciones.

3.2.4.1 El formato JPEG_RTP

JPEG_RTP es uno de los formatos de compresión de video admitidos por el protocolo RTP que trabaja sobre la plataforma JMF 2.1.1.e.

Un flujo transportado por JPEG_RTP significa que el medio (datos) van a ser comprimidos inicialmente por un códec JPEG y enseguida empaquetados y enviados en un paquete RTP (paquetizado en RTP) [EIM 2006].

Una consideración que se debe tomar al transmitir el video en JPEG_RTP es que solo pueden ser transmitidos videos cuyas dimensiones sean múltiplo de 8 píxeles.

3.2.4.2 El formato H.263_RTP

El formato de codificación de video H.263 fue propuesto por la Unión Internacional de Telecomunicaciones en el año 1995 enfocándose principalmente en las necesidades las videoconferencias. Posteriormente la versión fue mejorada agregándole características adicionales.

Los objetivos del formato de compresión H.263 consisten en los siguientes cuatro escenarios [RuR 2004]:

1. Dividir cada cuadro (frame) de video en bloques de píxeles de manera que el procesamiento del cuadro de video pueda ser llevado al nivel de bloque por medio del uso de la transformación DTC (Transformada Discreta del Coseno).
2. Aprovechar las redundancias espaciales existentes dentro del cuadro de video por medio de la codificación del bloque original a través de transformación, cuantización y entropía.
3. Aprovechar las dependencias temporales que existen entre los diferentes bloques en cuadros sucesivos, de manera que solo los cambios entre cuadros sucesivos sean codificados. Esto es realizando una estimación y compensación de movimiento.
4. Aprovechar cualquier redundancia espacial restante que exista dentro del cuadro de video por codificación de bloques residuales, por ejemplo, la diferencia entre el bloque original y el correspondiente bloque estimado.

En el formato de compresión de video H.263 cada imagen se divide en bloques (GOB, Group Of Blocks) o bien en rebanadas.

Los GOB son divididos posteriormente en macrobloques (MB), luego cuada macrobloque es dividido en bloques individuales a los cuales se les extrae un componente de luminancia y dos componentes de diferencia de color (Y, C_B y C_R). [UIT 2005].

La codificación del video es aplicada por medio de la predicción que se efectúa entre imágenes. El modo de codificación en el que se aplica la predicción temporal se denomina INTER; el modo de codificación se denomina INTRA cuando no se aplica la predicción temporal.

Para mejor comprensión de la terminología empleada se definen los conceptos [UIT 2005]:

- INTRA: imagen que no tiene imagen(es) de referencia para predicción (también denominada imagen I);
- INTER: imagen que utiliza una imagen de referencia temporalmente previa (también denominada imagen p).

En la figura 3.7 puede apreciarse una imagen ilustrativa de los GOB, macrobloques, bloques y componentes.

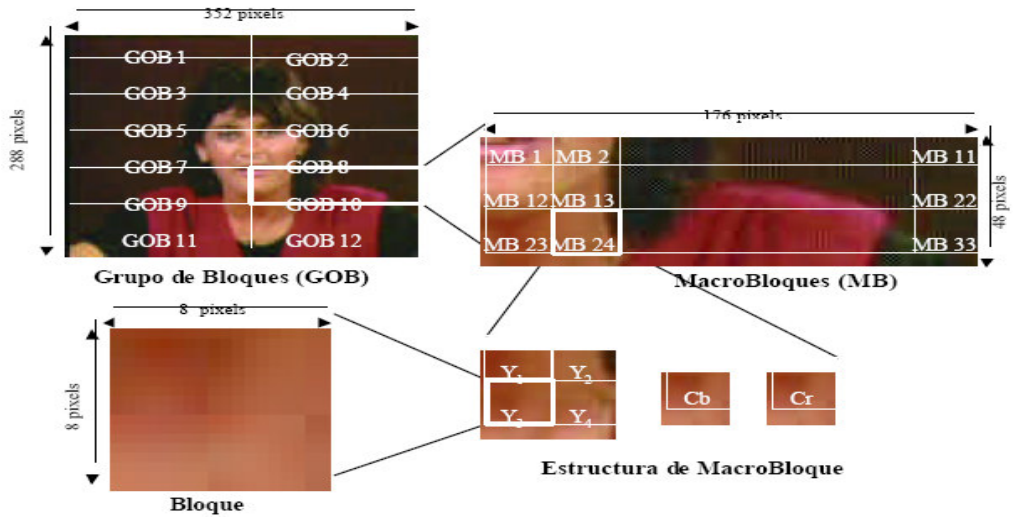


Figura 3.7. Ejemplo de imagen dividida en GOB, macrobloques, bloques y componentes. [VaH].

Todas las componentes de luminancia o diferencia de color de un macrobloque son pre-estimadas espacial o temporalmente y la predicción residual resultante es transmitida usando codificación de transformación.

Posteriormente cada componente de color de la predicción residual es subdividida en bloques y cada bloque es transformado usando una transformación entera y los coeficientes son cuantizados y transmitidos usando métodos de codificación de entropía. [RuR 2004].

Para transmitir el video codificado en H.263 a través de Internet, la salida del codificador puede ser directamente empaquetada por RTP. Para cada frame de video o flujo de bits en H.263, pude ser transportado en la carga útil del paquete (payload) RTP las alteraciones, incluyendo el código de inicio de la figura, el encabezado interno de la figura y cualquier otro código generado por la compresión [EIM 2006].

Los formatos de imagen permitidos por la implementación Java para la transmisión de video en formato H.263_RTP son:

- 128 x 96 píxeles de ancho y alto, respectivamente;
- 176 x 144 píxeles de ancho y alto, respectivamente;
- 352 x 288 píxeles de ancho y alto, respectivamente.

Como RTP no garantiza un servicio confiable en la entrega en orden de los datos, los paquetes pueden sufrir pérdidas en la red. Para conseguir la mejor recuperación posible de estas pérdidas, el decodificador necesita procesar otros paquetes que están siendo recibidos.

Para mayor información sobre el formato de compresión H.263 es posible consultar [UIT 2005].

3.2.5 Descripción del prototipo

En las siguientes secciones se dará una descripción del prototipo desarrollado.

3.2.5.1 Aplicación servidor

La aplicación servidor consta básicamente de dos ventanas:

- La ventana **Configuración Parámetros** y;
- La ventana **Pantalla Principal**.

Cuando el prototipo es iniciado, la ventana de **Configuración Parámetros** es mostrada. En esta ventana es posible elegir algunos parámetros básicos que influirán en el comportamiento del prototipo.

La ventana **Configuración Parámetros** tiene la apariencia mostrada en la figura 3.8; a continuación se describen cada una de funcionalidades que brinda.

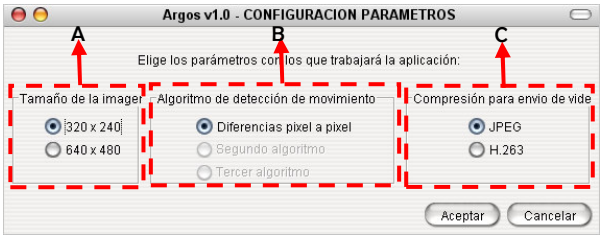


Figura 3.8. Ventana Configuración Parámetros de la aplicación servidor del prototipo desarrollado.

Las secciones referidas a continuación han sido marcadas en la figura 3.8 con líneas punteadas y debidamente señaladas.

- La sección **A** permite elegir el tamaño de imagen con la que trabajará el prototipo.
- La sección **B** permite elegir el algoritmo de detección de movimiento del que hará uso el prototipo en la sesión iniciada.
- La sección **C** permite elegir el tipo de formato de compresión que se utilizará para la transmisión del video.

Al dar clic en el botón Aceptar, después de haber seleccionado los parámetros de configuración con los que trabajará el prototipo, se mostrará un mensaje de confirmación (ver figura 3.9). Si los parámetros son confirmados, el prototipo inicializará el dispositivo de captura y mostrará la ventana **Pantalla Principal**.

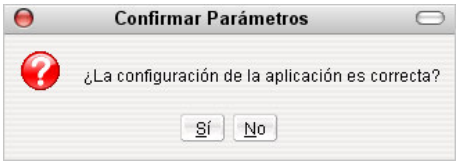


Figura 3.9. Ventana de confirmación de parámetros.

La ventana **Pantalla Principal** tiene la apariencia mostrada en la figura 3.10; a continuación son descritas las funcionalidades que proporciona.

Las secciones referidas a continuación han sido marcadas en la figura 3.10 con líneas punteadas y debidamente señaladas.

- La sección **A** permite revisar con qué parámetros de configuración está funcionando el prototipo.
- La sección **B** es el área en donde se muestran las imágenes captadas por el dispositivo de captura (cámara), las imágenes captadas se localizan inicialmente dentro de un marco color verde que índica que no se ha detectado movimiento, si el marco cambia a color rojo, entonces se ha detectado movimiento.
- La sección **C** es la botonera que compone el panel de control principal de la aplicación, es decir, en esta sección se puede iniciar o detener el procesamiento que se lleva a cabo para realizar la detección de movimiento. La botonera está compuesta por cuatro botones:

- **Detectar Movimiento**: inicia el procesamiento de detección de movimiento en el video, además activa el panel de Ajustes de la detección de movimiento (sección E).
- **Dejar de Detectar Mov**: detiene el procesamiento de detección de movimiento en el video, además desactiva el panel de Ajustes de la detección de movimiento (sección E).
- **Guardar 1 seg. de video:** guarda en el disco duro una secuencia de imágenes obtenidas del video captado.
- **Mostrar video a color:** permite que el video captado por la cámara pueda ser mostrado en escala de grises o a color.

- La sección **D** lleva un registro, a manera de texto, de cada una de las acciones que realiza el usuario en esta interfaz y las acciones que son iniciadas o detenidas internamente en el prototipo

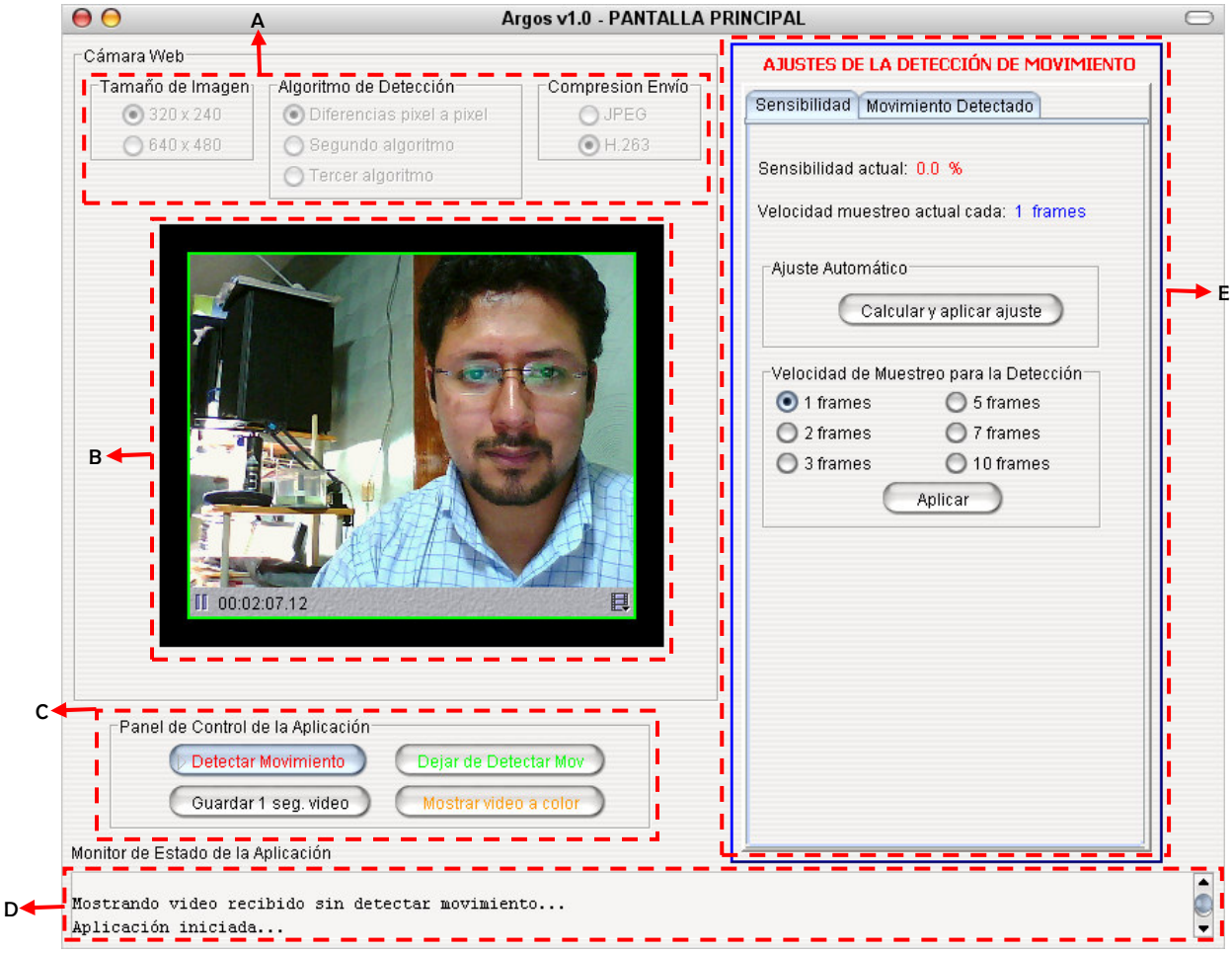


Figura 3.10. Ventana Pantalla Principal de la aplicación servidor del prototipo desarrollado.

- La sección **E** compone el panel de ajustes de la detección de movimiento (que se activa al presionar el botón Detectar Movimiento de la sección C) se divide a su vez en dos pestañas:

- **Sensibilidad**: en ésta pestaña se pueden encontrar opciones para ajustar la sensibilidad de la detección de movimiento. Se compone de las secciones:

  - **E1**: muestra el valor de la sensibilidad actual y cada cuantos frames se realiza la comparación de frames para detectar movimiento.
  - **E2**: permite calcular y aplicar el valor de sensibilidad que obtiene el prototipo de manera automática basado en la velocidad de muestreo seleccionada.
  - **E3**: permite seleccionar la velocidad de muestreo con la que trabajará el prototipo.

- **Movimiento Detectado**: en ésta pestaña se pueden configurar las distintas acciones que se realizarán en caso de que se detecte movimiento en el video. Se compone de las secciones:

  - **E4**: muestra el valor de umbral que utiliza el prototipo para decidir si se ha detectado o no movimiento en el video.
  - **E5**: permite cambiar el valor de umbral de manera manual, también permite activar o desactivar las alarmas visual y sonora que se emiten cuando se detecta movimiento en el video.
  - **E6**: permite configurar la dirección IP de la computadora donde se ejecuta la aplicación cliente; si se activa la casilla Enviar Video, la transmisión del video comenzará en el momento en el que se detecte movimiento en el video. Si se desactiva la casilla Enviar Video, la transmisión no se realizará.

En la figura 3.11 se muestran las dos pestañas que componen la sección E y se señalan las secciones que componen a cada pestaña.

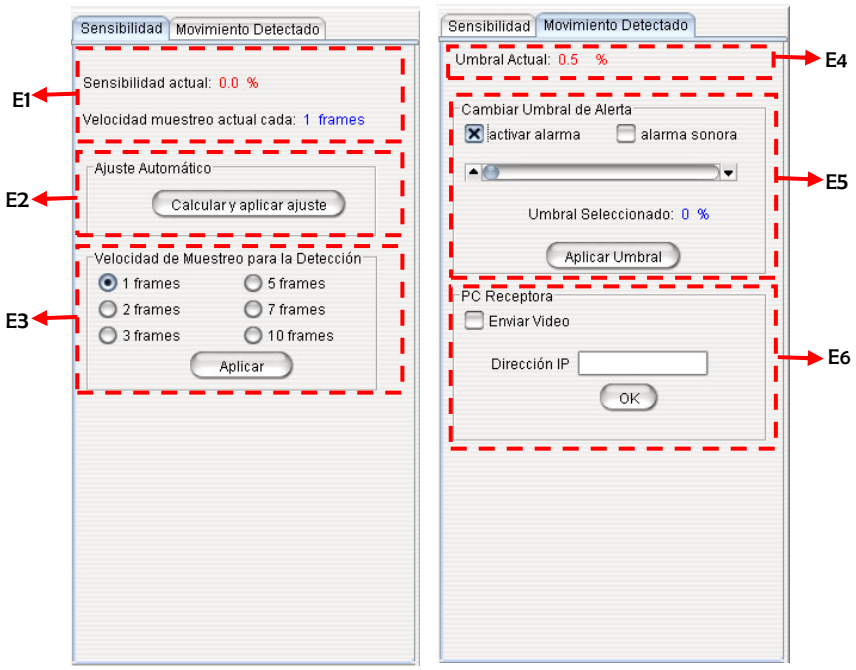


Figura 3.11. Elementos de las pestañas que componen a la sección E.

3.2.5.2 Aplicación cliente

La aplicación cliente consta básicamente de dos ventanas:

- La ventana **Instrucciones** y;
- La ventana **Pantalla Cliente**.

Cuando el prototipo es iniciado, la ventana de **Instrucciones** es mostrada. En esta ventana, además de poder leer instrucciones básicas, es posible escribir la dirección IP de la máquina en donde se ejecuta la aplicación servidor.

La ventana **Instrucciones** tiene la apariencia mostrada en la figura 3.12; a continuación se describen cada una de las funcionalidades que brinda.

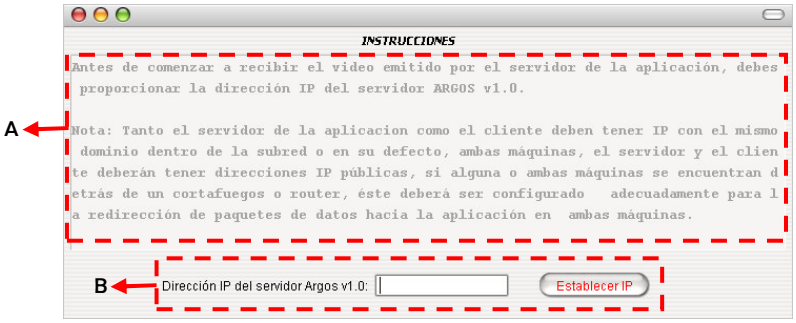


Figura 3.12. Ventana Instrucciones de la aplicación cliente del prototipo desarrollado.

Las secciones referidas a continuación han sido marcadas en la figura 3.12 con líneas punteadas y debidamente señaladas.

- La sección **A** permite mostrar algunas instrucciones.
- La sección **B** permite escribir y establecer la dirección IP de la máquina en donde se ejecuta la aplicación servidor.

Al dar clic en el botón Establecer IP, después de haber escrito la dirección IP con la que trabajará el prototipo, se mostrará un mensaje de confirmación (ver figura 3.13). Si la dirección IP es confirmada, el prototipo entrará en un estado de espera de datos y la ventana **Instrucciones** se bloqueará; cuando los datos comiencen a ser recibidos se mostrará la ventana **Pantalla Cliente**.

Figura 3.13. Ventana de confirmación de Dirección IP.

La ventana **Pantalla Cliente** tiene la apariencia mostrada en la figura 3.14; a continuación son descritas las funcionalidades que proporciona.

Las secciones referidas a continuación han sido marcadas en la figura 3.14 con líneas punteadas y debidamente señaladas.

- La sección **A** es el área en donde se muestran las imágenes recibidas después haber aplicado el procesamiento de seguimiento de movimiento.
- La sección **B** es una botonera que permite detener o reanudar la recepción de imágenes, cada operación se realiza cuando se da clic en el botón respectivo.
- La sección **C** forma parte del panel de control de la aplicación cliente y permite elegir la velocidad de muestreo del proceso de seguimiento de movimiento; es decir, cada cuantos frames se tomará el frame de referencia con el cual se realizará el seguimiento del movimiento.
- La sección **D** también forma parte del panel de control de la aplicación cliente y permite elegir de forma manual la sensibilidad del proceso de seguimiento de movimiento; es decir, establece el porcentaje de movimiento que se debe presentar para poder aplicar el proceso de seguimiento de movimiento.

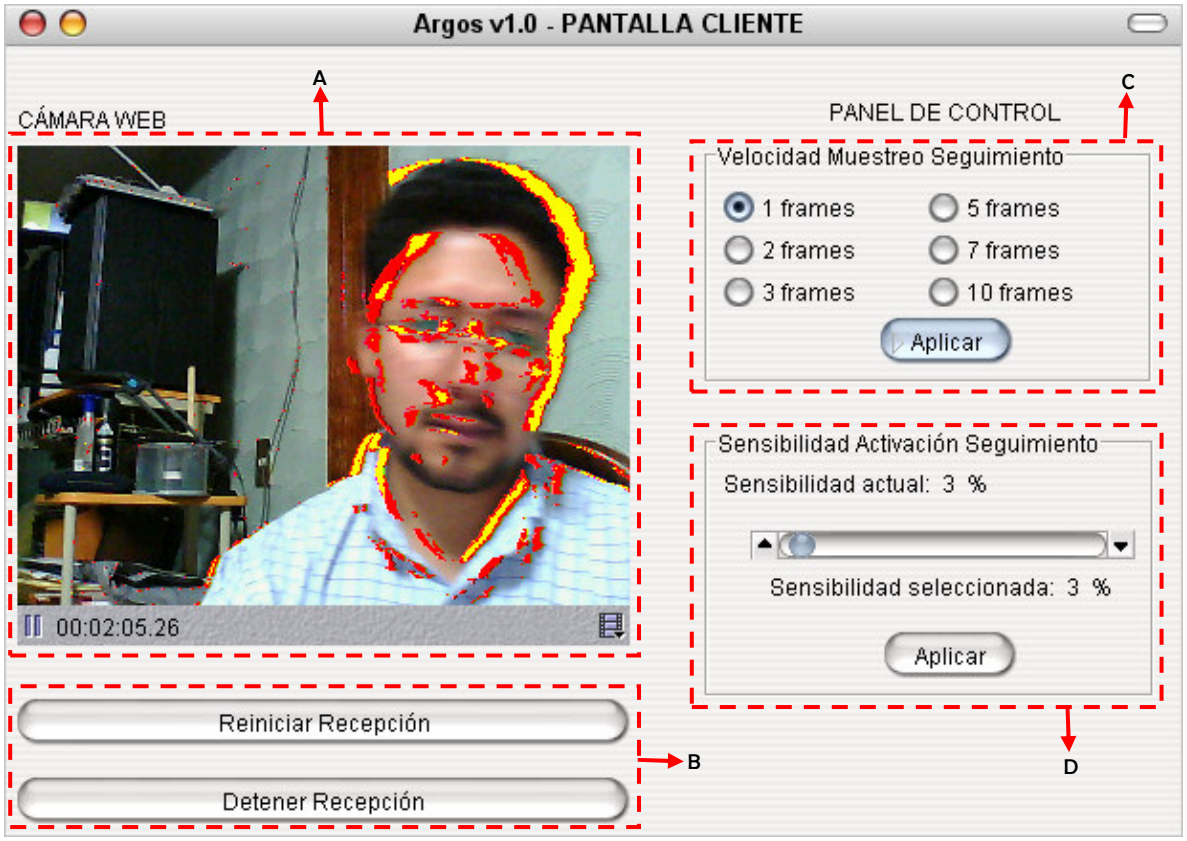


Figura 3.14. Ventana Pantalla Cliente de la aplicación cliente del prototipo desarrollado.

Capítulo 4

Experimentación

En este capítulo son presentadas las pruebas que se realizaron con el prototipo.

Las pruebas fueron realizadas con el prototipo terminado (versión 1.0), la cual fue diseñada para obtener los resultados finales y no resultados parciales de los procesamientos aplicados al video.

Es importante hacer mención de que las pruebas fueron ejecutadas en un ambiente controlado, respectivo a cada una de las pruebas. Cada escenario de prueba se describe dentro de la sección de cada prueba dentro del presente capítulo.

4.1 Descripción de la cámara de pruebas

Para las pruebas se utilizó la cámara mostrada en la figura 4.1.

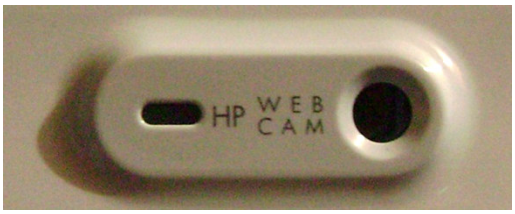


Figura 4.1. Cámara web utilizada para las pruebas.

La cámara de pruebas es un dispositivo de captura (cámara web) con conexión USB 2.0, integrada a la parte superior de la pantalla de un equipo portátil marca Hewlett Packard modelo Pavilion dv6620la. Las características específicas de la cámara de prueba se muestran en la tabla 4.1.

Característica	Descripción
Sensor	CMOS
Tamaño Máximo de Imagen	640 píxeles x 480 píxeles (ancho, alto)
Frecuencia Máxima	30 frames por segundo
Enfoque	Foco fijo al infinito
Tipo de Conexión	USB 2.0

Tabla 4.1. Características de la cámara de pruebas.

4.2 Descripción de la computadora servidor

Para cada una de las pruebas mencionadas en las siguientes secciones la computadora servidor (en donde se ejecutaba la aplicación servidor del prototipo) fue la misma con el fin de poder realizar una mejor comparación del desempeño de la aplicación en ambas pruebas.

La computadora servidor que se utilizó es una computadora portátil de marca Hewlett Packard modelo dv6620la cuyas características más importantes se mencionan en la tabla 4.2.

Característica	Descripción
Velocidad del Procesador	1.9 GHz (doble núcleo)
Memoria RAM	2048 MB
Memoria de Video	128 MB (compartidos)
Capacidad Disco Duro	160 GB
Interfaz de Red utilizada	Ethernet 10/100 Mbps
Sistema Operativo	Windows Vista Ultimate 64 bits

Tabla 4.2. Características de la computadora servidor.

En cada una de las pruebas las computadoras en donde se ejecutó la aplicación cliente fueron diferentes y sus características se mencionan dentro de la sección de cada una de las pruebas.

4.3 Prueba en casa habitación y conexión Ethernet LAN

En esta prueba se utilizó la infraestructura de red de datos existente, y supuso la simulación de un entorno empresarial controlado en el cual cada una de las dos habitaciones representa a una oficina en particular.

Una de las habitaciones simulaba ser la oficina que debía vigilarse permanentemente; mientras que otra habitación simulaba ser la oficina en donde se recibían las alertas de vigilancia.

4.3.1 Descripción de la computadora cliente

La computadora cliente que se utilizó para esta prueba es una computadora de escritorio de marca SONY VAIO modelo PCV-RS20M cuyas características más importantes se mencionan en la tabla 4.3.

Característica	Descripción
Velocidad del Procesador	2.66 GHz
Memoria RAM	1536 MB
Memoria de Video	128 MB
Capacidad Disco Duro	80 GB
Interfaz de Red utilizada	Ethernet 10/100 Mbps
Sistema Operativo	Windows XP Professional SP2

Tabla 4.3. Características de la computadora cliente (casa habitación).

4.3.2 Descripción de la conexión utilizada

La topología de red utilizada en la prueba es una topología física en estrella y lógica de bus que utiliza como medio físico de comunicación el cable de par trenzado (UTP categoría 5) con un ancho de banda máximo de 100 Mbps, una longitud máxima de segmento de 100 metros.

Como nodo central de la red se encuentra un módem ruteador inalámbrico marca 2WIRE modelo 1701HG proporcionado por Telmex (Teléfonos de México) al contratar Prodigy Infinitum.

El tipo de red utilizado en esta prueba es bastante eficiente ya que si un paquete de datos se pierde, su retransmisión es casi inmediata.

La conexión de red utilizada puede observarse en la figura 4.2.

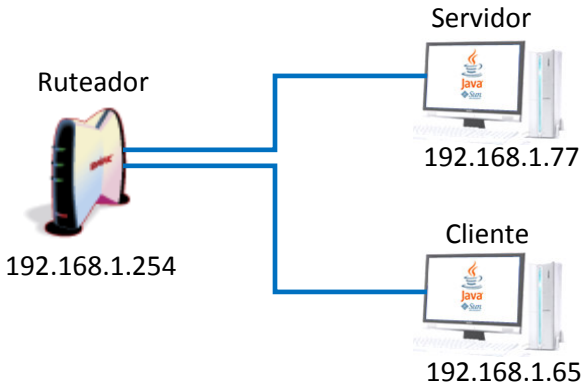


Figura 4.2. Diagrama de la red utilizada en la prueba (casa habitación).

4.3.3 Realización de pruebas

Los datos recabados de manera experimental en cada una de las cuatro pruebas se enfocan en el tiempo aproximado de retardo observado en la transmisión y procesamiento del video.

La primera prueba realizada en la casa habitación fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 320 x 240 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** JPEG_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.3.

Figura 4.3. Pantalla Cliente haciendo la primera prueba (casa habitación).

La segunda prueba realizada en la casa habitación fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 640 x 480 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** JPEG_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.4.

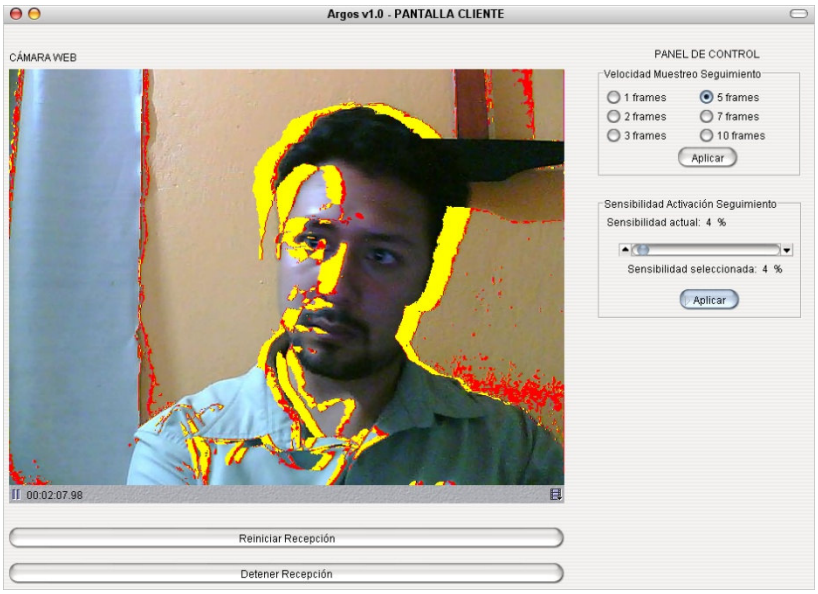


Figura 4.4. Pantalla Cliente haciendo la segunda prueba (casa habitación).

La tercera prueba realizada en la casa habitación fue con la siguiente configuración de la Aplicación Servidor:

- **Tamaño de video:** 320 x 240 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** H.263_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.5.

Figura 4.5. Pantalla Cliente haciendo la tercera prueba (casa habitación).

La cuarta prueba realizada en la casa habitación fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 640 x 480 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** H.263_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.6.

Figura 4.6. Pantalla Cliente haciendo la cuarta prueba (casa habitación).

4.3.4 Tabla de comparación

Los datos reunidos de manera experimental en las pruebas anteriores son resumidos en la tabla 4.4 para que el lector pueda obtener conclusiones con una mayor facilidad.

Servidor		Cliente	
Tamaño Video Enviado (píxeles)	Formato	Tamaño Video Recibido (píxeles)	Retardo Promedio
320 x 240	JPEG_RTP	320 x 240	30×10^{-2} seg
640 X 480	JPEG_RTP	640 x 480	50×10^{-2} seg
320 x 240	H.263_RTP	176 x 144	10×10^{-2} seg
640 X 480	H.263_RTP	352 x 288	16×10^{-2} seg

Tabla 4.4. Datos de las pruebas realizadas en Ethernet (casa habitación).

Los resultados presentados en la tabla 4.4 son discutidos al final de este capítulo.

4.4 Prueba en casa habitación y conexión ADSL (Internet)

En esta prueba se utilizó la infraestructura de red de datos utilizada por Telmex, y supuso la simulación de un entorno en donde dos empresas, una dedicada a la vigilancia y otra que debe ser vigilada, se comunican por medio de Internet y además las aplicaciones operan bajo condiciones controladas.

Se supuso que la computadora servidor se localizaba en algún área u oficina específica de la empresa que debe ser vigilada y por lo tanto; se supuso que la computadora cliente se localiza en la empresa dedicada a la vigilancia la cual se ubica geográficamente lejos de la empresa vigilada.

4.4.1 Descripción de la computadora cliente

La computadora cliente que se utilizó para esta prueba es una computadora portátil de marca Hewlett Packard modelo HP NX6325 cuyas características más importantes se mencionan en la tabla 4.5.

Característica	Descripción
Velocidad del Procesador	1.6 GHz (doble núcleo)
Memoria RAM	1024 MB
Memoria de Video	256 MB (compartida)
Capacidad Disco Duro	80 GB
Interfaz de Red utilizada	Wireless 54 Mbps
Sistema Operativo	Windows XP Professional SP2

Tabla 4.5. Características de la computadora cliente (Internet).

4.4.2 Descripción de la conexión utilizada

Para las empresas simuladas se representó una red interna para cada una de ellas.

La topología de red, para cada una de las empresas, utilizada en la prueba es una topología física en estrella y lógica de bus que utiliza como medio físico de comunicación el cable de par trenzado (UTP categoría 5) con un ancho de banda máximo de 100 Mbps, una longitud máxima de segmento de 100 metros.

Para la red de la empresa que será vigilada se tiene como nodo central un módem ruteador inalámbrico ADSL marca 2WIRE modelo 1701HG proporcionado por Telmex al contratar Prodigy Infinitum, la velocidad de conexión es de 1024 Kbps y 128 Kbps, de bajada y subida de datos respectivamente.

Para la red de la empresa que será vigilante se tiene como nodo central un módem ruteador inalámbrico ADSL marca 2WIRE modelo RG2701HG proporcionado por Telmex al contratar Prodigy Infinitum, la velocidad de conexión es de 512 Kbps y 128 Kbps, de bajada y subida de datos respectivamente.

La calidad del servicio de red utilizado en esta prueba depende directamente de la infraestructura del Proveedor del Servicio de Internet.

La conexión de red utilizada puede observarse en la figura 4.7.

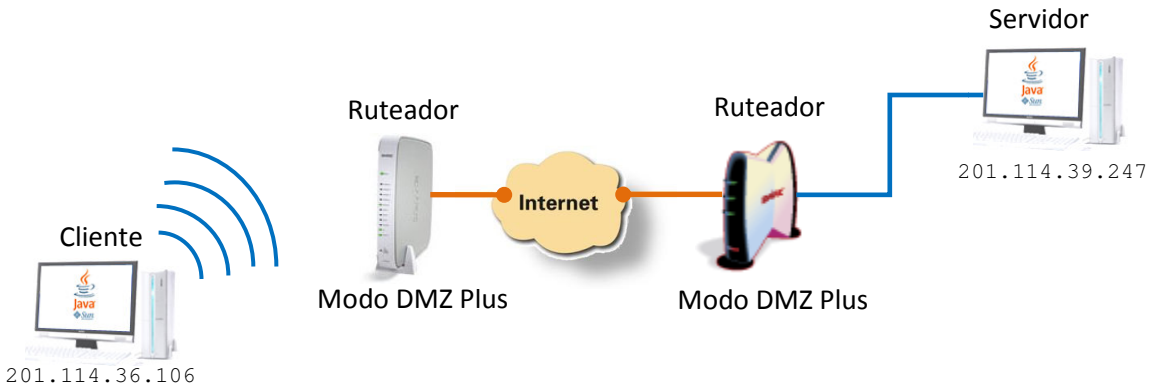


Figura 4.7. Diagrama de la red utilizada en la prueba (Internet).

4.4.3 Realización de pruebas

Los datos recabados de manera experimental en cada una de las cuatro pruebas se enfocan en el tiempo aproximado de retardo observado en la transmisión y procesamiento del video.

La primera prueba realizada en Internet fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 320 x 240 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** JPEG_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.8.

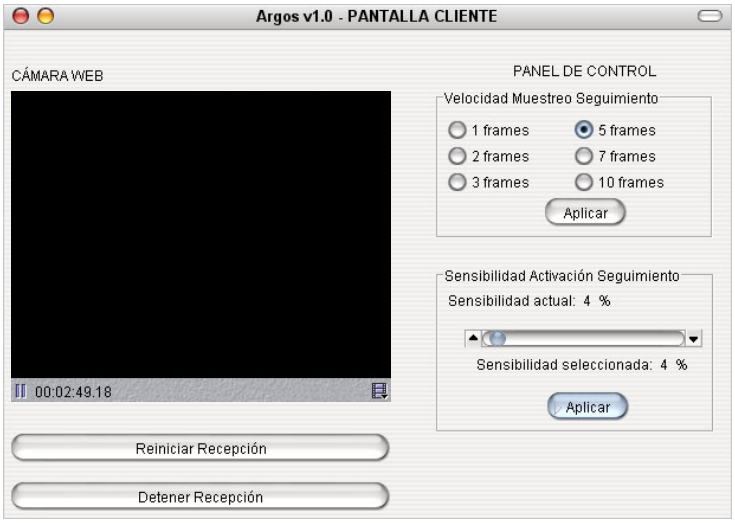


Figura 4.8. Pantalla Cliente haciendo la primera prueba (Internet).

En esta primera prueba se observó que la aplicación cliente muestra únicamente una pantalla en color negro debido a que muchos de los paquetes de datos se pierden en la red, muchos otros llegan en desorden y otros tantos llegan corruptos.

Así, la aplicación tiene que esperar a que se pueda formar una imagen con los paquetes de datos recibidos y luego mostrarla en el monitor, sin embargo, ninguna imagen se forma completamente ya que se reciben pocos paquetes de datos para interpretar el frame.

En la figura 4.9 se puede ver una herramienta de Java que permite observar en tiempo real el flujo de datos que se da entre los procesos involucrados en la recepción y presentación del video. Los diferentes procesos son mostrados como cajas negras.

Figura 4.9. Flujo de datos entre los procesos involucrados en la recepción y representación del video (primera prueba).

Con la ayuda de la herramienta mostrada en la figura 4.9 podemos deducir que los datos únicamente llegan al buffer de video el cual no se llena (únicamente se llena el 50%) por lo que no envía sus datos al proceso de despaquetización JPEG y por lo tanto no hay imágenes que decodificar, sin imágenes no se puede aplicar el seguimiento de movimiento ni mostrar resultados en la aplicación cliente.

La segunda prueba realizada en Internet fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 640 x 480 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** JPEG_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.10.

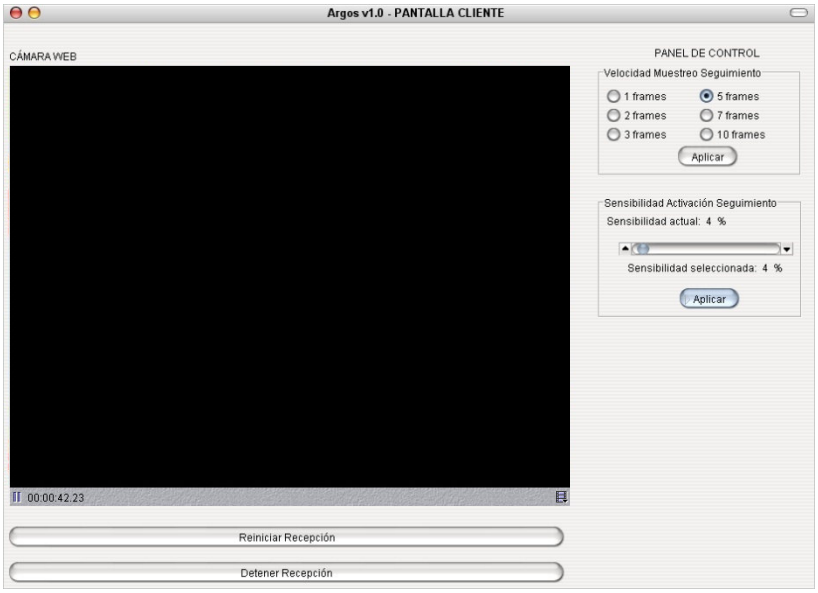


Figura 4.10. Pantalla Cliente haciendo la segunda prueba (Internet).

En la segunda prueba realizada se puede observar que la aplicación cliente también muestra únicamente una pantalla negra, esto se debe a las mismas razones que en la primera prueba salvando la diferencia de que el tamaño del video transmitido en esta segunda prueba es el doble del tamaño del video transmitido en la primera prueba.

En la figura 4.11 se puede observar que el buffer de video no se llena ni en un 50%, lo que sí ocurría en la primera prueba y por lo tanto los demás procesos no reciben datos de entrada y sólo se muestra una pantalla negra.

Figura 4.11. Flujo de datos entre los procesos involucrados en la recepción y representación del video (segunda prueba).

La tercera prueba realizada en Internet fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 320 x 240 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** H.263_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.12.

Figura 4.12. Pantalla Cliente haciendo la tercera prueba (Internet).

La cuarta prueba realizada en Internet fue con la siguiente configuración de la aplicación servidor:

- **Tamaño de video:** 640 x 480 píxeles de ancho y alto, respectivamente.
- **Formato de compresión de video:** H.263_RTP.

La captura de la pantalla de la aplicación cliente puede observarse en la figura 4.13.

Figura 4.13. Pantalla Cliente haciendo la cuarta prueba (Internet).

4.4.4 Tabla de comparación

Los datos reunidos de manera experimental en las pruebas anteriores son resumidos en la tabla 4.6 para que el lector pueda obtener conclusiones con una mayor facilidad.

Servidor		Cliente	
Tamaño Video Enviado (píxeles)	Formato	Tamaño Video Recibido (píxeles)	Retardo Promedio
320 x 240	JPEG_RTP	320 x 240	indeterminado
640 X 480	JPEG_RTP	640 x 480	Indeterminado
320 x 240	H.263_RTP	176 x 144	menor a 0.5 seg
640 X 480	H.263_RTP	352 x 288	menor a 1 seg

Tabla 4.6. Datos de las pruebas realizadas en Internet.

Los resultados presentados en la tabla 4.6 son discutidos al final de este capítulo.

4.5 Consideraciones

Aunque se realizaron más pruebas con otros equipos de cómputo, éstas no se documentaron por ser consideradas dentro del desarrollo del prototipo y no como pruebas de la versión final del prototipo.

Sin embargo, de manera experimental, gracias a las pruebas no documentadas se obtuvieron los siguientes requisitos básicos de hardware para la ejecución del prototipo:

- Procesador Intel Celeron a 266 MHz
- Memoria RAM de 96 MB
- Disco Duro de 20 GB
- Tarjeta de Red 10/100 Mbps

No se realizaron pruebas con otras velocidades de transmisión de datos debido a problemas de logística.

4.6 Discusión de resultados

Con la ayuda de las pruebas mencionadas y de los datos obtenidos sobre el desempeño del prototipo es posible destacar que en la prueba realizada en la casa habitación con infraestructura de red Ethernet se puede trabajar cualquiera de los dos tamaños de video que el

prototipo permite manejar y, además, el retardo observado en la transmisión del video a través de la red es mínimo con cualquiera de los dos formatos de transmisión del video.

Debe hacerse notar que en esta prueba el uso del formato de transmisión JPEG_RTP tuvo retrasos de 30 x 10^{-2} segundos y 50 x 10^{-2} segundos para los tamaños de video de 320 x 240 píxeles y 640 x 480 píxeles, respectivamente, lo cual es considerado como un retraso aceptable para las características de la prueba siendo justificado por la gran calidad observada en el video recibido por la aplicación cliente.

Al hacer uso del formato de transmisión H.263_RTP los tiempos de retraso se redujeron aproximadamente a un tercio de los tiempos de retraso observados con el uso del formato de transmisión JPEG_RTP. Los tiempos de retraso observados con el uso del formato de transmisión H.263_RTP fueron de 10 x 10^{-2} segundos y 16 x 10^{-2} segundos para los tamaños de video de 320 x 240 píxeles y 640 x 480 píxeles, respectivamente, aunque debe de tomarse en cuenta que con el uso del formato H.263_RTP el video recibido por la aplicación cliente es re-escalado a un menor tamaño.

Por tanto, si se tiene una infraestructura de red similar a la descrita en la prueba realizada en la casa habitación, puede utilizarse cualquier combinación que pueda hacerse entre los tamaños de video y los formatos de transmisión que permite usar el prototipo; siendo recomendada por su calidad la siguiente combinación: tamaño de video 320 x 240 píxeles y formato de transmisión JPEG_RTP.

En el caso de la prueba realizada utilizando la conexión a Internet de tipo ADSL, cuyas características se describen en la sección 4.4.2 de este capítulo, se observó que la transmisión de video de cualquier tamaño manejado por el prototipo utilizando el formato JPEG_RTP no se lleva a cabo de manera adecuada y muestra en la aplicación cliente un área cuyas dimensiones se corresponden con el tamaño de video transmitido y que únicamente presenta un color negro.

Las dimensiones del área se deben a que la aplicación cliente recibe algunos paquetes del video transmitido y el color negro se debe a que los paquetes llegan con retrasos muy grandes y no alcanzan a formar una imagen completa ya que cada imagen está formada de varios paquetes de manera que si no se puede completar una imagen ésta no es interpretada y es descartada junto con sus paquetes recibidos previamente.

El formato de transmisión de video H.263_RTP presentó mejores resultados en la prueba realizada con la conexión a Internet de tipo ADSL. Los tiempos de retraso observados con el uso de este formato de transmisión fueron de menos de medio segundo para el tamaño de video de 320 x 240 píxeles y de menos de un segundo para el tamaño de video de 640 x 480 píxeles.

Debe hacerse notar que al usar el formato de transmisión H.263_RTP el tamaño del video recibido es menor que el tamaño de video enviado y además la calidad del video, que es menor que la calidad mostrada con el formato JPEG_RTP, es todavía aceptable considerando el tipo de conexión a Internet utilizada y el tiempo de retraso obtenido en las pruebas.

Capítulo 5

Conclusiones

5.1 Resultados

Del desarrollo de la presente tesis se obtuvieron los siguientes resultados:

- Un prototipo de software que no necesita hacer uso de hardware ni de software especial para realizar la detección y seguimiento de movimiento remoto.
- Se diseñó una arquitectura base para aplicaciones de procesamiento de video a través de Internet cuyas características principales son: modularidad, adaptabilidad y facilidad de modificación.
- El prototipo fue implementado en su totalidad en lenguaje de programación Java el cual es un lenguaje moderno y le brinda al prototipo la posibilidad de ser utilizado por un periodo de tiempo mayor debido a la popularidad del lenguaje de programación.
- El prototipo tiene un costo de implementación relativamente bajo y muchas áreas de aplicación.

5.2 Conclusiones

Durante el desarrollo de este trabajo de tesis se cumplieron todos los objetivos planteados.

El uso de la tecnología Java para procesamiento de video (JMF 2.1.1.e) hace posible procesar cada frame de la secuencia de video como un arreglo unidimensional en donde cada elemento es a su vez un arreglo cuyos elementos son las componentes R, G y B de cada píxel. Esta forma de procesamiento resulta muy eficiente y proporciona un mejor tiempo de respuesta de los algoritmos implementados para la detección y seguimiento de movimiento.

La arquitectura de procesamiento propuesta permite que puedan ser implementados nuevos códec de procesamiento de video de forma sencilla y/o intercambiarlos con otros antiguos.

La interfaz gráfica desarrollada propone un ambiente intuitivo y amigable para que el usuario pueda ajustar en tiempo de ejecución algunos parámetros que influyen en el comportamiento del procesamiento teniendo así una mejor adaptabilidad del prototipo a entornos no previstos, por ejemplo, las características de la cámara utilizada, la iluminación y las características de la red de comunicación. Interfaces gráficas con las mismas características no se observaron en ninguna aplicación libre mencionada en el estado del arte.

El desarrollo del prototipo en el lenguaje de programación Java permite que el software pueda ser ejecutado en una amplia gama de sistemas operativos actuales y, de igual forma, utilizar una amplia gama de cámaras web disponibles en el mercado con conexión USB y de uso común (siempre y cuando manejen el formato *video for Windows*, “vfw”).

El prototipo fue probado con al menos dos cámaras web:

- HP Web Cam

- Logitech QuickCam

Además, también fue probado bajo los siguientes sistemas operativos y computadoras:

- Windows 98 SE en una computadora Compaq Presario 5010 con procesador Intel Celeron a 266 MHz y 96MB en RAM. Únicamente se probó la aplicación cliente.
- Windows XP Professional Service Pack 2, se probaron tanto la aplicación cliente como la aplicación servidor, ver capítulo 4.
- Windows Vista Ultimate Edition, se probaron tanto la aplicación cliente como la aplicación servidor.

De lo anterior se puede concluir que, de acuerdo a la revisión del estado de arte, el prototipo desarrollado, así como su arquitectura, aportan un camino viable para cubrir el espacio poco explotado de la detección y seguimiento de movimiento remoto con muy buenos resultados.

5.3 Trabajo futuro

El prototipo puede ser objeto de varias modificaciones:

- Mejorar el prototipo para que puedan ser admitidas más de una cámara web.

- Implementar más de un algoritmo de detección y/o de seguimiento de movimiento.

- Se puede complementar el prototipo para que únicamente envíe el video cuando se detecte movimiento y; cuando éste deje de ser detectado, el envío del video se detenga.

- Implementar un algoritmo de reconocimiento de patrones con el fin de obtener una identificación o clasificación de los objetos que presenten movimiento en el video.

- Dotar al prototipo de la capacidad de activar sensores, actuadores, movimiento de cámara, etc. con fin de aumentar su aplicabilidad.

- Implementar nuevos formatos de compresión para el envío de video, los cuales tienden a ser más eficientes y cada vez proporcionan una mejor calidad de imagen.

- Establecer un mecanismo de interacción entre la aplicación servidor y la aplicación cliente, por ejemplo, el envío de mensajes de texto, comandos que activen dispositivos externos o alarmas en el entorno de la aplicación servidor desde la aplicación cliente.

REFERENCIAS BIBLIOGRÁFICAS

[AbM+ 2006] M.F. Abdelkader, R. Chellapa, Q. Zheng y A. L. Chan, *Integrated Motion Detection and Tracking for Visual Surveillance,* Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS 2006), Enero 2006.

[AzA+] A. Y. Azama Makishi y T. F. Huamán Huanca*. Detector de Eventos Remotos Basado en Técnicas de Procesamiento Digital de Video. Universidad Peruana de Ciencias Aplicadas, Lima, Perú*.

[BaR 2004] R*. Barriuso. Desarrollo de un Sistema de Detección de Intrusos para Pasarelas Domésticas. Proyecto de Fin de Carrera*. Universidad Politécnica de Madrid, España, Mayo 2004.

[BlJ+ 2003] J. Black, T. J. Ellis, y P. Rosin. *A novel method for video tracking performance evaluation*. In Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), paginas 125-132, Octubre 2003.

[BrP+ 2005] P. Brox, S. Sánchez-Solano, I. Baturone, A. Barriga, J. Gutiérrez-Ríos, F. Fernández*. Implementación Sobre FPGA de un Algoritmo de Desentrelazado de Video Basado en Lógica Fuzzy. Simposio sobre Lógica Fuzzy y Soft Computing, LFSC2005 (EUSFLAT), pp.83-90,* Septiembre 2005.

[BrT 2003] T. Bräunl*. Embedded Robotic, Mobile Robot Design and Applications with Embedded Systems. 1st edition, editorial Springer, 2003.*

[BuA 2002] A. Bulcão, *Análisis Instrumental de la Imagen en Movimiento: Ritmo, Síncresis y Atención Visual*. Tesis Doctoral. Universidad Autónoma de Barcelona, 2002.

[DoI 2006] I. Domínguez Jiménez. *Detección y Seguimiento de Objetos en Movimiento*, Tesis de Maestría, UAEH, 2006.

[DuJ+ 2000] J. Durán de Jesus, J. J. Villacorta Calvo y A. Izquierdo Fuente*. Modelado de un Sistema de Vigilancia Video-Acústico. Universidad de Valladolid, Valladolid*, 2000.

[EcB 2003] B. Eckel*. Piensa en Java. 2da edición, editorial Prentice Hall, 2003.*

[ElM 2006] M. Elkind*.* UM SISTEMA CONFIÁVEL DE DISTRIBUIÇÃO DE MÍDIA*. Tesis de Maestría,* Universidade Federal do Rio de Janeiro, *2006. Portugués.*

[ElT 2002] T.J. Ellis. *Performance Metrics and Methods for Tracking in Surveillance*, Proceedings of the Third International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2002), Copenhagen, Junio 2002.

[HeE 2000] E. Hernández-Orallo*. Transmisión de Datos en Tiempo Real, síntesis de protocolos y redes para transmisión en tiempo real, Agosto 2000.*

[JiG+ 2005] G.Jing, D. Rajan y C.E. Siong. *Motion Detection With Adaptive Background And Dynamic Thresholds*, Information, Communications and Signal Processing, 2005 Fifth International Conference on, páginas 41-45, Diciembre 2005.

[JMF 1999] Guía de utilización de JMF2.1.1.e.
http://java.sun.com/products/java-media/jmf/2.1.1/guide/JMFIX.html

[KIOS 2008] Knowledge kiosk, kioskea.net.
http://es.kioskea.net/internet/rtcp.php3

[MaF+ 2002] *F. Matthew, S. Karl, N. Matthew, H. Fangpo. Evaluation of Motion Detection Techniques for Video Surveillance,* Information, Decision and Control, 2002. Final Program and Abstracts, páginas 247-252, Febrero 2002.

[MeA 2003] *A. Meléndez Islas. Estimación de Fondo y Primer Plano en Secuencias de Imágenes para la Detección de Movimiento. Tesis de Maestría, INAOE, Febrero 2003*.

[MiC 2003] Microsoft Corporation. *Diccionario de Internet y Redes de Microsoft. 1era ed. Español, editorial McGraw-Hil, 2003.*

[MoJ 2004] J*. Montejano. Sistemas Avanzados de Teleasistencia en el Hogar.* E-Cooperación en la Administración Pública*,* Octubre 2004.

[OtN 1979] N. Otsu. *A threshold selection method from grey-level histograms*, IEEE Trans. Systems, Man and Cybernetics, 9(1):62-66, January 1979.

[PaD+ 2004] *D. Park, M. Gambini, M. Mejail. Seguimiento de Objetos en Video usando Contornos Activos.* Universidad de Buenos Aires*.* Buenos Aires, Argentina, 2004.

[RuR 2004] *R. Ruíz. El H.264 un nuevo estándar para la compresión de video y su aplicación a la videoconferencia.* Universidad de Buenos Aires*.* Buenos Aires, Argentina, 2004.

[SiC+ 2003] C. Simmons y J. Causey. *Redes con Microsoft Windows XP. 1era ed. español, editorial McGraw-Hill, 2003.*

[StW 2004] W. Stallings. *Comunicaciones y Redes de Computadores. 7ma ed. español, editorial Pearson Education, 2004.*

[SuK+ 2000] K. Suzuki, I. Horiba, and N. Sugie. *Fast Connected-Component Labeling Based on Sequential Local Operations in the Course of Forward Raster Scan Followedby Backward Raster Scan* , IEEE International Conference on Pattern Recognition (ICPR'00)-Volume 2, p.24-34, 2000

[UIT 2005] UIT. *Recomendación UIT-T H.263 Codificación de Video para Comunicación a Baja Velocidad Binaria. Español, 2005.*

[VaH] H. Vargas-Cortés. *Análisis de Implementaciones de los Protocolos MPEG4 y H.263. Presentación.*

[WaM+ 2006] M. Wang, C. Huang y H. Lin, *An Intelligent Surveillance System Based on an Omnidirectional Vision Sensor,* IEEE Conference on Cybernetics and Intelligent Systems 2006, Junio 2006.

[WIKI 2008] Wikipedia, la enciclopedia libre
http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

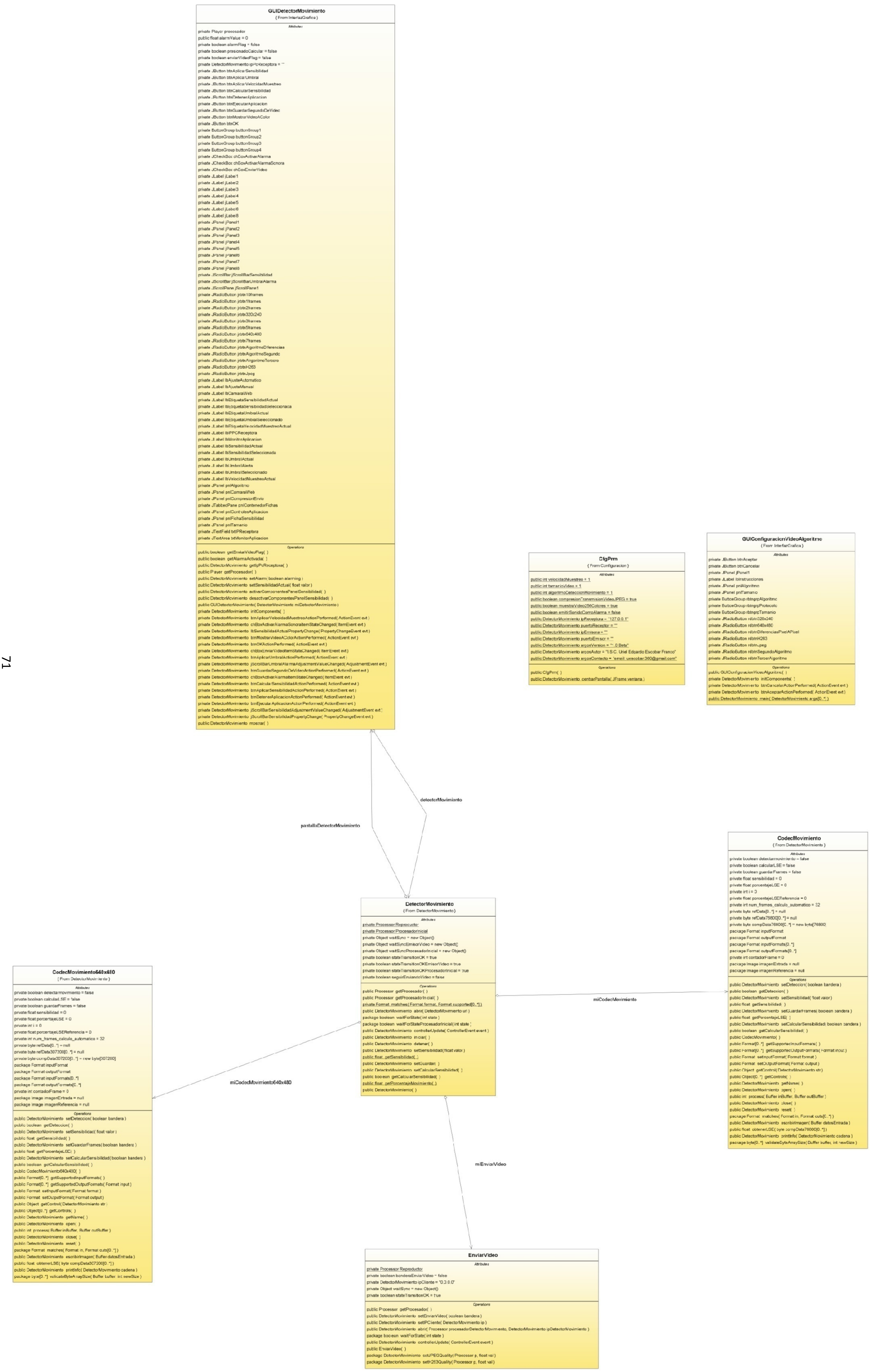
[WIKI 2007] Wikipedia, la enciclopedia libre
http://es.wikipedia.org/wiki/Real-time_Transport_Protocol

[ZeV+ 2006] V. Zeljkovic y D. Pokrajac, *Motion Detection Based Multimedia Supported Intelligent Video Surveillance System,* 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, Zadar, Croatia, Junio 2006.

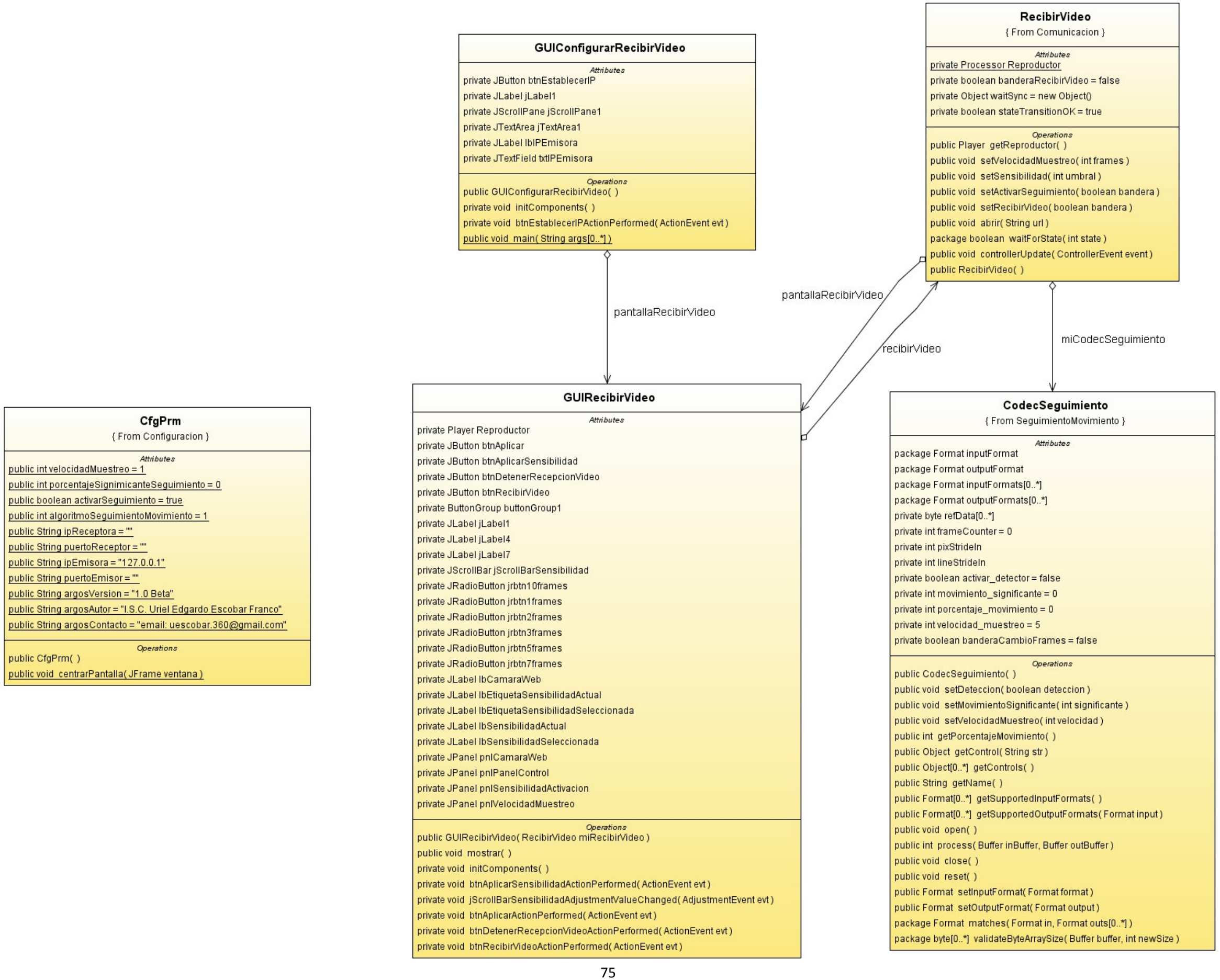
[ZhZ 2002] Z. Zhang. *Mining surveillance video for independent motion detection*, Second IEEE International Conference on Data Mining (ICDM'02), p. 741, Diciembre 2002.

APÉNDICE A. DIAGRAMAS DE CLASES

Aplicación Servidor

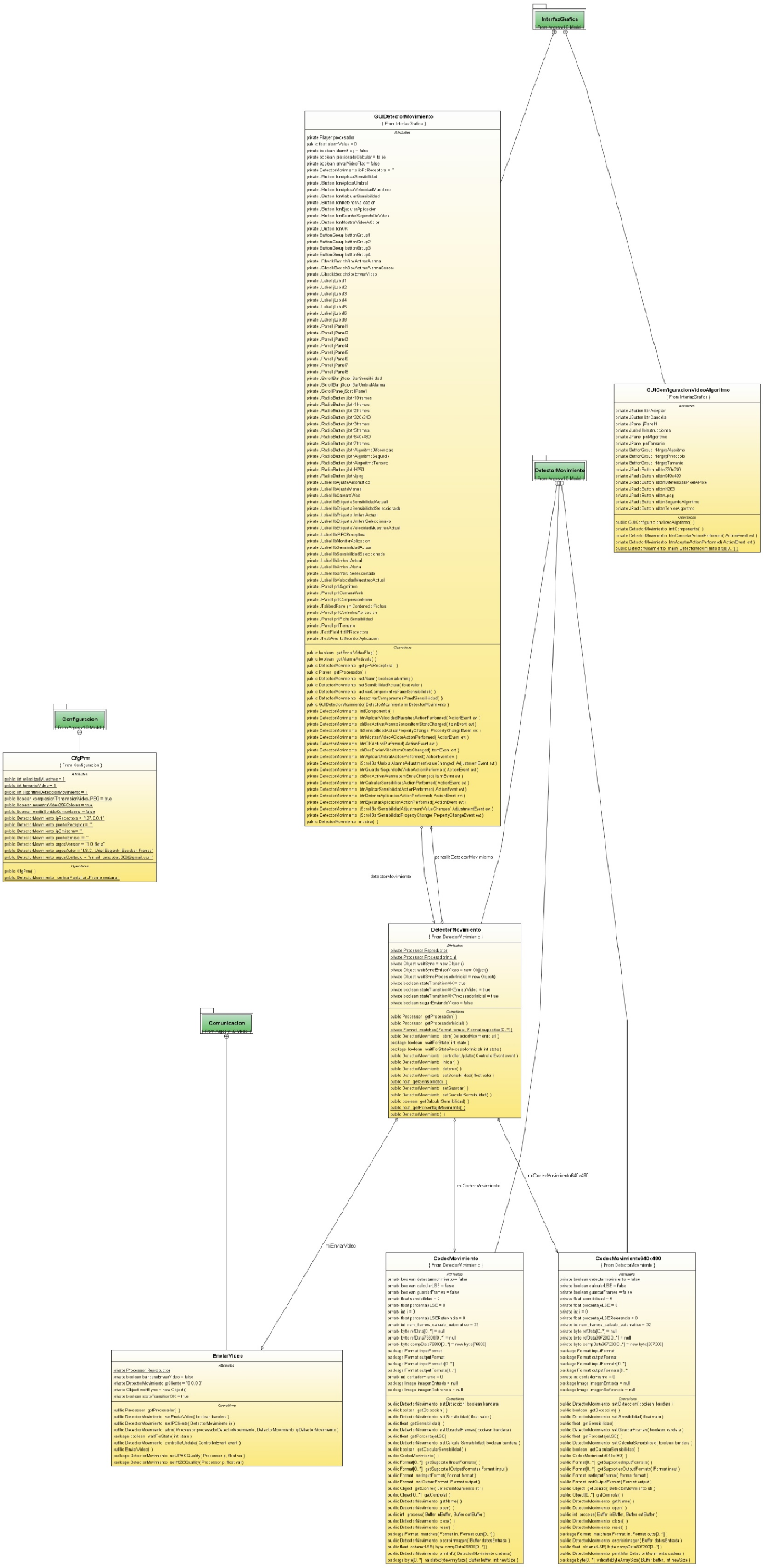

GUIDetectorMovimiento
CfgPrm
GUIConfiguracionVideoAlgoritmo
detectorMovimiento
pantallaDetectorMovimiento
CodecMovimiento
DetectorMovimiento
CodecMovimiento640x480
miCodecMovimiento
miCodecMovimiento640x480
miEnviarVideo
EnviarVideo

Aplicación Cliente

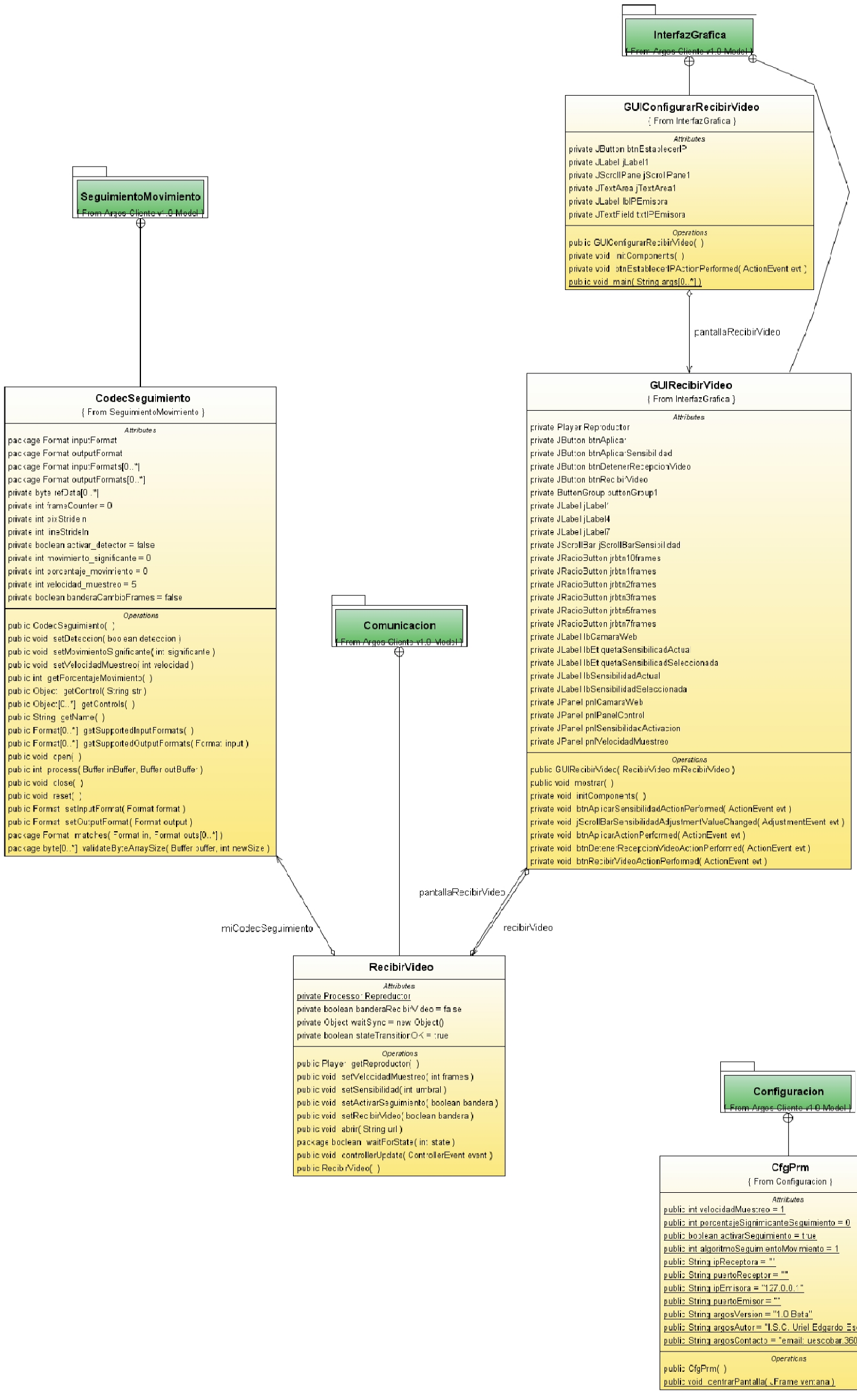

RecibirVideo
{ From Comunicacion }
Attributes
private Processor Reproductor
private boolean banderaRecibirVideo = false
private Object waitSync = new Object()
private boolean stateTransitionOK = true
Operations
public Player getReproductor( )
public void setVelocidadMuestreo(int frames)
public void setSensibilidad(int umbral)
public void setActivarSeguimiento(boolean bandera)
public void setRecibirVideo(boolean bandera)
public void abrir(String url)
package boolean waitForState(int state)
public void controllerUpdate(ControllerEvent event)
public RecibirVideo( )
GUIConfigurarRecibirVideo
Attributes
private JButton btnEstablecerIP
private JLabel jLabel1
private JScrollPane jScrollPane1
private JTextArea jTextArea1
private JLabel lblIPEmisora
private JTextField txtIPEmisora
Operations
public GUIConfigurarRecibirVideo( )
private void initComponents( )
private void btnEstablecerIPActionPerformed(ActionEvent evt)
public void main(String args[0..*])
pantallaRecibirVideo
pantallaRecibirVideo
recibirVideo
miCodecSeguimiento
CfgPrm
{ From Configuracion }
Attributes
public int velocidadMuestreo = 1
public int porcentajeSignimicanteSeguimiento = 0
public boolean activarSeguimiento = true
public int algoritmoSeguimientoMovimiento = 1
public String ipReceptora = ""
public String puertoReceptor = ""
public String ipEmisora = "127.0.0.1"
public String puertoEmisor = ""
public String argosVersion = "1.0 Beta"
public String argosAutor = "I.S.C. Uriel Edgardo Escobar Franco"
public String argosContacto = "email: uescobar.360@gmail.com"
Operations
public CfgPrm( )
public void centrarPantalla(JFrame ventana)
GUIRecibirVideo
Attributes
private Player Reproductor
private JButton btnAplicar
private JButton btnAplicarSensibilidad
private JButton btnDetenerRecepcionVideo
private JButton btnRecibirVideo
private ButtonGroup buttonGroup1
private JLabel jLabel1
private JLabel jLabel4
private JLabel jLabel7
private JScrollBar jScrollBarSensibilidad
private JRadioButton jrbtn10frames
private JRadioButton jrbtn1frames
private JRadioButton jrbtn2frames
private JRadioButton jrbtn3frames
private JRadioButton jrbtn5frames
private JRadioButton jrbtn7frames
private JLabel lbCamaraWeb
private JLabel lbEtiquetaSensibilidadActual
private JLabel lbEtiquetaSensibilidadSeleccionada
private JLabel lbSensibilidadActual
private JLabel lbSensibilidadSeleccionada
private JPanel pnlCamaraWeb
private JPanel pnlPanelControl
private JPanel pnlSensibilidadActivacion
private JPanel pnlVelocidadMuestreo
Operations
public GUIRecibirVideo(RecibirVideo miRecibirVideo)
public void mostrar( )
private void initComponents( )
private void btnAplicarSensibilidadActionPerformed(ActionEvent evt)
private void jScrollBarSensibilidadAdjustmentValueChanged(AdjustmentEvent evt)
private void btnAplicarActionPerformed(ActionEvent evt)
private void btnDetenerRecepcionVideoActionPerformed(ActionEvent evt)
private void btnRecibirVideoActionPerformed(ActionEvent evt)
CodecSeguimiento
{ From SeguimientoMovimiento }
Attributes
package Format inputFormat
package Format outputFormat
package Format inputFormats[0..*]
package Format outputFormats[0..*]
private byte refData[0..*]
private int frameCounter = 0
private int pixStrideIn
private int lineStrideIn
private boolean activar_detector = false
private int movimiento_significante = 0
private int porcentaje_movimiento = 0
private int velocidad_muestreo = 5
private boolean banderaCambioFrames = false
Operations
public CodecSeguimiento( )
public void setDeteccion(boolean deteccion)
public void setMovimientoSignificante(int significante)
public void setVelocidadMuestreo(int velocidad)
public int getPorcentajeMovimiento( )
public Object getControl(String str)
public Object[0..*] getControls( )
public String getName( )
public Format[0..*] getSupportedInputFormats( )
public Format[0..*] getSupportedOutputFormats(Format input)
public void open( )
public int process(Buffer inBuffer, Buffer outBuffer)
public void close( )
public void reset( )
public Format setInputFormat(Format format)
public Format setOutputFormat(Format output)
package Format matches(Format in, Format outs[0..*])
package byte[0..*] validateByteArraySize(Buffer buffer, int newSize)

APÉNDICE B. DIAGRAMAS DE PAQUETES

Aplicación Servidor


InterfazGrafica
GUIDetectorMovimiento
{ From InterfazGrafica }
Attributes
private Player procesador
public float alarmValue = 0
private boolean alarmFlag = false
private boolean presionadoCalcular = false
private boolean enviarVideoFlag = false
private DetectorMovimiento ipPcReceptora = ""
private JButton btnAplicaSensibilidad
private JButton btnAplicaUmbral
private JButton btnAplicaVelocidadMuestreo
private JButton btnCalcularSensibilidad
private JButton btnDetenerAplicacion
private JButton btnEjecutarAplicacion
private JButton btnGuardarSegundoDeVideo
private JButton btnMostrarVideoAColor
private JButton btnOK
private ButtonGroup buttonGroup1
private ButtonGroup buttonGroup2
private ButtonGroup buttonGroup3
private ButtonGroup buttonGroup4
private JCheckBox chkNoActivarAlarma
private JCheckBox chkSolActivarAlarmaSeccionada
private JCheckBox chkSolEnviarVideo
private JLabel jLabel1
private JLabel jLabel2
private JLabel jLabel3
private JLabel jLabel4
private JLabel jLabel5
private JLabel jLabel6
private JLabel jLabel8
private JPanel jPanel1
private JPanel jPanel2
private JPanel jPanel3
private JPanel jPanel4
private JPanel jPanel5
private JPanel jPanel6
private JPanel jPanel7
private JPanel jPanel8
private JScrollBar jScrollBarSensibilidad
private JScrollBar jScrollBarUmbralAlarma
private JScrollPane jScrollPane1
private JRadioButton jbtr10Frames
private JRadioButton jbtr1Frames
private JRadioButton jbtr2Frames
private JRadioButton jbtr320x240
private JRadioButton jbtr3Frames
private JRadioButton jbtr5Frames
private JRadioButton jbtr640x480
private JRadioButton jbtr7Frames
private JRadioButton jbtrAlgoritmoDiferencias
private JRadioButton jbtrAlgoritmoSegundo
private JRadioButton jbtrAlgoritmoTercero
private JRadioButton jbtrH263
private JRadioButton jbtrJpeg
private JLabel lblAjusteAutomatico
private JLabel lblAjusteManual
private JLabel lblCamaraWeb
private JLabel lblEtiquetaSensibilidadActual
private JLabel lblEtiquetaSensibilidadSeleccionada
private JLabel lblEtiquetaUmbralActual
private JLabel lblEtiquetaUmbralSeleccionado
private JLabel lblEtiquetaVelocidadMuestreoActual
private JLabel lblPPCReceptora
private JLabel lblMonitorAplicacion
private JLabel lblSensibilidadActual
private JLabel lblSensibilidadSeleccionada
private JLabel lblUmbralActual
private JLabel lblUmbralAlarma
private JLabel lblUmbralSeleccionado
private JLabel lblVelocidadMuestreoActual
private JPanel pnlAgoritmo
private JPanel pnlCamaraWeb
private JPanel pnlCompresionEmisi
private JTabbedPane pnlContenedorFichas
private JPanel pnlControlesAplicacion
private JPanel pnlFichaSensibilidad
private JPanel pnlTamanio
private JTextField txtIPReceptora
private JTextArea txtMonitorAplicacion
Operations
public boolean getEnviarVideoFlag( )
public boolean getAlarmaActivada( )
public DetectorMovimiento getIpPcReceptora( )
public Player getProcesador( )
public DetectorMovimiento setAlarm(boolean alarming)
public DetectorMovimiento setSensibilidadActual(float valor)
public DetectorMovimiento activarComponentesPanelSensibilidad( )
public DetectorMovimiento desactivarComponentesPanelSensibilidad( )
public GUIDetectorMovimiento(DetectorMovimiento mDetectorMovimiento)
private DetectorMovimiento initComponents( )
private DetectorMovimiento btnAplicarVelocidadMuestreoActionPerformed(ActionEvent evt)
private DetectorMovimiento chkSolActivarAlarmaSeccionItemStateChanged(ItemEvent evt)
private DetectorMovimiento lblSensibilidadActualPropertyChange(PropertyChangeEvent evt)
private DetectorMovimiento btnMostrarVideoAColorActionPerformed(ActionEvent evt)
private DetectorMovimiento btnOKActionPerformed(ActionEvent evt)
private DetectorMovimiento chkSolEnviarVideoItemStateChanged(ItemEvent evt)
private DetectorMovimiento btnAplicarUmbralActionPerformed(ActionEvent evt)
private DetectorMovimiento jScrollBarUmbralAlarmaAdjustmentValueChanged(AdjustmentEvent evt)
private DetectorMovimiento btnGuardarSegundoDeVideoActionPerformed(ActionEvent evt)
private DetectorMovimiento chkSolActivarAlarmaItemStateChanged(ItemEvent evt)
private DetectorMovimiento btnCalcularSensibilidadActionPerformed(ActionEvent evt)
private DetectorMovimiento btnAplicarSensibilidadActionPerformed(ActionEvent evt)
private DetectorMovimiento btnDetenerAplicacionActionPerformed(ActionEvent evt)
private DetectorMovimiento btnEjecutarAplicacionActionPerformed(ActionEvent evt)
private DetectorMovimiento jScrollBarSensibilidadAdjustmentValueChanged(AdjustmentEvent evt)
private DetectorMovimiento jScrollBarSensibilidadPropertyChange(PropertyChangeEvent evt)
public DetectorMovimiento main(String[] args)
DetectorMovimiento
GUIConfiguracionVideoAlgoritmo
{ From InterfazGrafica }
Attributes
private JButton btnAceptar
private JButton btnCancelar
private JPanel jPanel1
private JLabel lblInstrucciones
private JPanel pnlAlgoritmo
private JPanel pnlTamanio
private ButtonGroup rbtngrpAlgoritmo
private ButtonGroup rbtngrpProtocolo
private ButtonGroup rbtngrpTamanio
private JRadioButton rdbtn320x240
private JRadioButton rdbtn640x480
private JRadioButton rdbtnDiferenciasPixelAPixel
private JRadioButton rdbtnJPEG
private JRadioButton rdbtnJpeg
private JRadioButton rdbtnSegundoAlgoritmo
private JRadioButton rdbtnTercerAlgoritmo
Operations
public GUIConfiguracionVideoAlgoritmo( )
private DetectorMovimiento initComponents( )
private DetectorMovimiento btnCancelarActionPerformed(ActionEvent evt)
private DetectorMovimiento btnAceptarActionPerformed(ActionEvent evt)
public DetectorMovimiento main(DetectorMovimiento args[])
Configuracion
CfgPrm
{ From Configuracion }
Attributes
public int velocidadMuestreo = 1
public int tamanioVideo = 1
public int algoritmoDeteccionMovimiento = 1
public boolean compresionTransmisionVideo_JPEG = true
public boolean muestraVideo256Colores = true
public boolean emitirSonidoCorreoAlarma = false
public DetectorMovimiento ipReceptora = "127.0.0.1"
public DetectorMovimiento puertoReceptor = ""
public DetectorMovimiento ipEmisora= ""
public DetectorMovimiento puertoEmisor = ""
public DetectorMovimiento argosVersion = "1.0 Beta"
public DetectorMovimiento argosAutor = "I.S.C. Uriel Elizardo Escobar Franco"
public DetectorMovimiento argosCorreoAutor = "ismael_uriel_eeu@hotmail.com"
Operations
public CfgPrm( )
public DetectorMovimiento centrarPantalla(JFrame ventana)
Comunicacion
DetectorMovimiento
{ From Area v4.0 Model }
DetectorMovimiento
{ From DetectorMovimiento }
Attributes
private Processor Reproductor
private Processor ProcesadorInicial
private Object waitSync = new Object()
private Object waitSyncEmisorVideo = new Object()
private Object waitSyncProcesadorInicial = new Object()
private boolean stateTransitionOK = true
private boolean stateTransitionOKEmisorVideo = true
private boolean stateTransitionOKProcesadorInicial = true
private boolean seguirEmitiendoVideo = false
Operations
public Processor getProcesador( )
public Processor getProcesadorInicial( )
private Format matches(Format format, Format supported[0..*])
public DetectorMovimiento abrir(DetectorMovimiento d)
package boolean waitForState(int state)
package boolean waitForStateProcesadorInicial(int state)
public DetectorMovimiento controllerUpdate(ControllerEvent event)
public DetectorMovimiento iniciar( )
public DetectorMovimiento detener( )
public DetectorMovimiento setSensibilidad(float valor)
public float getSensibilidad( )
public DetectorMovimiento setGuardar( )
public DetectorMovimiento setCalcularSensibilidad( )
public boolean getCalcularSensibilidad( )
public float getPorcentajeMovimiento( )
public DetectorMovimiento( )
EnviarVideo
Attributes
private Processor Reproductor
private boolean banderaEnviarVideo = false
private DetectorMovimiento pCliente = "0.0.0.0"
private Object waitSync = new Object()
private boolean stateTransitionOK = true
Operations
public Processor getProcesador( )
public DetectorMovimiento setEnviarVideo(boolean bandera)
public DetectorMovimiento setIPCliente(DetectorMovimiento ip)
public DetectorMovimiento abrir(Processor procesadorDetectorMovimiento, DetectorMovimiento ipDetectorMovimiento)
package boolean waitForState(int state)
public DetectorMovimiento controllerUpdate(ControllerEvent event)
public EnviarVideo( )
package DetectorMovimiento setJPEGQuality(Processor p, float val)
package DetectorMovimiento setH263Quality(Processor p, float val)
CodecMovimiento
{ From DetectorMovimiento }
Attributes
private boolean detectarMovimiento = false
private boolean calcularLSE = false
private boolean guardarFrames = false
private float sensibilidad = 0
private float porcentajeLSE = 0
private int i = 0
private float porcentajeLSEReferencia = 0
private int num_frames_calculo_automatico = 32
private byte refData[0..*] = null
private byte refData7680[0..*] = null
private byte compData7680[0..*] = new byte[76800]
package Format inputFormat
package Format outputForma
package Format inputFormats[0..*]
package Format outputFormats[0..*]
private int contadorFrame = 0
package Image imagenEntrada = null
package Image imagenReferencia = null
Operations
public DetectorMovimiento setDeteccion(boolean bandera)
public boolean getDeteccion( )
public DetectorMovimiento setSensibilidad(float valor)
public float getSensibilidad( )
public DetectorMovimiento setGuardarFrames(boolean bandera)
public float getPorcentajeLSE( )
public DetectorMovimiento setCalcularSensibilidad(boolean bandera)
public boolean getCalcularSensibilidad( )
public CodecMovimiento( )
public Format[0..*] getSupportedInputFormats( )
public Format[0..*] getSupportedOutputFormats(Format input)
public Format setInputFormat(Format format)
public Format setOutputFormat(Format output)
public Object getControl(DetectorMovimiento str)
public Object[0..*] getControls( )
public DetectorMovimiento getName( )
public DetectorMovimiento open( )
public int process(Buffer inBuffer, Buffer outBuffer)
public DetectorMovimiento close( )
public DetectorMovimiento reset( )
package Format matches(Format in, Format outs[0..*])
public DetectorMovimiento escribirImagen(Buffer datosEntrada)
public float obtenerLSE(byte compData7680[0..*])
public DetectorMovimiento printInfo(DetectorMovimiento cadena)
package byte[0..*] validateByteArraySize(Buffer buffer, int newSize)
CodecMovimiento640x480
{ From DetectorMovimiento }
Attributes
private boolean detectarMovimiento = false
private boolean calcularLSE = false
private boolean guardarFrames = false
private float sensibilidad = 0
private float porcentajeLSE = 0
private int i = 0
private float porcentajeLSEReferencia = 0
private int num_frames_calculo_automatico = 32
private byte refData[0..*] = null
private byte refData307200[0..*] = null
private byte compData307200[0..*] = new byte[307200]
package Format inputFormat
package Format outputForma
package Format inputFormats[0..*]
package Format outputFormats[0..*]
private int contadorFrame = 0
package Image imagenEntrada = null
package Image imagenReferencia = null
Operations
public DetectorMovimiento setDeteccion(boolean bandera)
public boolean getDeteccion( )
public DetectorMovimiento setSensibilidad(float valor)
public float getSensibilidad( )
public DetectorMovimiento setGuardarFrames(boolean bandera)
public float getPorcentajeLSE( )
public DetectorMovimiento setCalcularSensibilidad(boolean bandera)
public boolean getCalcularSensibilidad( )
public CodecMovimiento640x480( )
public Format[0..*] getSupportedInputFormats( )
public Format[0..*] getSupportedOutputFormats(Format input)
public Format setInputForma(Format format)
public Format setOutputFormat(Format output)
public Object getControl(DetectorMovimiento str)
public Object[0..*] getControls( )
public DetectorMovimiento getName( )
public DetectorMovimiento open( )
public int process(Buffer inBuffer, Buffer outBuffer)
public DetectorMovimiento close( )
public DetectorMovimiento reset( )
package Format matches(Format in, Format outs[0..*])
public DetectorMovimiento escribirImagen(Buffer datosEntrada)
public float obtenerLSE(byte compData307200[0..*])
public DetectorMovimiento printInfo(DetectorMovimiento cadena)
package byte[0..*] validateByteArraySize(Buffer buffer, int newSize)
pantallaDetectorMovimiento
detectorMovimiento
miCodecMovimiento
miCodecMovimiento640x480
miEnviarVideo

Aplicación Cliente


SeguimientoMovimiento
{ From Argos Cliente v1.0 Model }
InterfazGrafica
{ From Argos Cliente v1.0 Model }
GUIConfigurarRecibirVideo
{ From InterfazGrafica }
Attributes
private JButton btnEstablecerIP
private JLabel jLabel1
private JScrollPane jScrollPane1
private JTextArea jTextArea1
private JLabel lblIPEmisora
private JTextField txtIPEmisora
Operations
public GUIConfigurarRecibirVideo( )
private void initComponents( )
private void btnEstablecerIPActionPerformed(ActionEvent evt)
public void main(String args[0..*])
pantallaRecibirVideo
CodecSeguimiento
{ From SeguimientoMovimiento }
Attributes
package Format input:Format
package Format outputFormat
package Format input:Formats[0..*]
package Format outputFormats[0..*]
private byte refData[0..*]
private int framsCounter = 0
private int pixStrideIn
private int lineStrideIn
private boolean activar_detector = false
private int movimiento_significante = 0
private int porcentaje_movimiento = 0
private int velocidad_muestreo = 5
private boolean banderaCambioFrames = false
Operations
public CodecSeguimiento( )
public void setDeteccion(boolean deteccion)
public void setMovimientoSignificante(int significante)
public void setVelocidadMuestreo(int velocidad)
public int getPorcentajeMovimiento( )
public Object getControl(String str)
public Object[0..*] getControls( )
public String getName( )
public Format[0..*] getSupportedInputFormats( )
public Format[0..*] getSupportedOutputFormats(Format input)
public void open( )
public int process(Buffer inBuffer, Buffer outBuffer)
public void close( )
public void reset( )
public Format setInputFormat(Format format)
public Format setOutputFormat(Format output)
package Format matches(Format in, Format outs[0..*])
package byte[0..*] validateByteArraySize(Buffer buffer, int newSize)
Comunicacion
{ From Argos Cliente v1.0 Model }
GUIRecibirVideo
{ From InterfazGrafica }
Attributes
private Player Reproductor
private JButton btnAplicar
private JButton btnAplicarSensibilidad
private JButton btnDetenerRecepcionVideo
private JButton btnRecibirVideo
private ButtonGroup buttonGroup1
private JLabel jLabel1
private JLabel jLabel4
private JLabel jLabel7
private JScrollBar jScrollBarSensibilidad
private JRadioButton jrbtn10frames
private JRadioButton jrbtn1frames
private JRadioButton jrbtn2frames
private JRadioButton jrbtn3frames
private JRadioButton jrbtn5frames
private JRadioButton jrbtn7frames
private JLabel lbCamaraWeb
private JLabel lbEtquetaSensibilidadActua
private JLabel lbEtquetaSensibilidadSeleccionada
private JLabel lbSensibilidadActual
private JLabel lbSensibilidadSeleccionada
private JPanel pnlCamaraWeb
private JPanel pnlPanelControl
private JPanel pnlSensibilidadActivacion
private JPanel pnlVelocidadMuestreo
Operations
public GUIRecibirVidec(RecibirVideo miRecibirVideo)
public void mostrar( )
private void initComponents( )
private void btnAplicarSensibilidadActionPerformed(ActionEvent evt)
private void jScrollBarSensibilidadAdjustmentValueChanged(AdjustmentEvent evt)
private void btnAplicarActionPerformed(ActionEvent evt)
private void btnDetenerRecepcionVideoActionPerformed(ActionEvent evt)
private void btnRecibirVideoActionPerformed(ActionEvent evt)
pantallaRecibirVideo
miCodecSeguimiento
recibirVideo
RecibirVideo
Attributes
private Processor Reproductor
private boolean banderaRecibirVideo = false
private Object waitSync = new Object()
private boolean stateTransitionOK = true
Operations
public Player getReproductor( )
public void setVelocidadMuestreo(int frames)
public void setSensibilidad(int umbral)
public void setActivarSeguimiento(boolean bandera)
public void setRecibirVideo(boolean bandera)
public void abrir(String url)
package boolean waitForState(int state)
public void controllerUpdate(ControllerEvent event)
public RecibirVideo( )
Configuracion
{ From Argos Cliente v1.0 Model }
CfgPrm
{ From Configuracion }
Attributes
public int velocidadMuestreo = 1
public int porcentajeSignificanteSeguimiento = 0
public boolean activarSeguimiento = true
public int algoritmoSeguimientoMovimiento = 1
public String ipReceptora = ""
public String puertoReceptor = ""
public String ipEmisora = "127.0.0.1"
public String puertoEmisor = ""
public String argosVersion = "1.0 Beta"
public String argosAutor = "I.S.C. Uriel Edgardo Escobar Franco"
public String argosContacto = "email: uescobar.360@gmail.com"
Operations
public CfgPrm( )
public void centrarPantalla(JFrame ventana)