**UNIVERSIDAD POLITÉCNICA DE PUEBLA**

**Programa Académico de Ingeniería en Informática**

# "Sistema Distribuido de Video-vigilancia para dispositivos móviles"

Ignacio Huitzil Velasco

Protocolo de Tesis NUMERO

Comité Evaluador

Dr. Eduardo López Domngez
Asesor

Dr. Jorge de la Calleja Mora
Coasesor

Dr. o MC. Nombre completo del segundo sinodal
Sinodal

Juan C. Bonilla, Puebla. Mayo 2012

# Índice

# Capítulo 1

# Planteamiento del problema de investigación unooo

## 1.1. Introducción

El término seguridad puede ser definido desde distintos puntos de vista y contextos, sin embargo, considerando sólo al que compete al ser humano, puede ser concebido como salvaguardar sus intereses fundamentales y hasta su propia vida. Actualmente para salvaguardar tales intereses y reducir riesgos, se han propuesto desarrollar sistemas de seguridad.

Los sistemas de seguridad con base en lo propuesto por Maloof *et al.* [3] deben llevar a cabo tres funciones principales: en primer lugar, la *protección* la cual consiste en prevenir hechos indeseables y que puedan ocurrir. En segundo lugar, la *detección* que consiste en determinar el momento exacto en el que ocurrió el hecho y que es auxiliado por los mecanismos de protección. Finalmente, después de la detección, se realiza la *respuesta*, en este paso se toman acciones tales como activar alarmas, generar avisos al centro de seguridad, entre otros.

Debido a la gran cantidad de información útil que puede ser obtenida de una secuencia de video, los sistemas de *video-vigilancia* se han convertido en una herramienta efectiva para brindar servicios de seguridad a las personas en diversos contextos [14, 19, 27]. Los sistemas de video-vigilancia se pueden definir como un sistema de cámaras colocadas dentro de un área (pública o privada en el interior o exterior) con el fin de monitorear y vigilar la actividad que se lleva a cabo [20]. Las imágenes obtenidas de este sistema se visualizan o se archivan para análisis futuros. Una gran demanda de este tipo de sistemas es visible en la seguridad de supermercados, edificios inteligentes, en el hogar, el uso militar y gubernamental [16, 29].

Los sistemas de video vigilancia, con base en el tipo de técnicas para detección de objetos y tecnología implementada en función de la escalabilidad y la comunicación, se dividen en tres generaciones [25, 24, 27]:

La primera generación está constituida por *sistemas analógicos* de Circuito Cerrado de Televisión (CCTV) análogicos. Algunos ejemplos donde se usan estos sistemas son en países como el Reino Unido, considerado el país con el mayor número de cámaras de vigilancia instaladas (alrededor de 4 millones); Francia en el 2011 instalará 60,000 cámaras en la vía pública y México que incursiona en este tipo de seguridad, colocando cámaras en lugares con alto tránsito de personas e índices delictivos [20, 36].

La segunda generación es la combinación del aprendizaje automático y visión por computadora con los sistemas CCTV con el fin de procesar imágenes y señales, llamándoles *sistemas semiautomáticos*. Por ejemplo, en [29] se propone un sistema de video vigilancia para la detección, clasificación de autos y peatones en movimiento. Su proceso de análisis se basa en tres niveles, 1) la extracción del objeto en movimiento, 2) el re-

conocimiento del mismo en movimiento y finalmente 3) su rastreo.

Los sistemas de video-vigilancia de tercera generación, también llamados *sistemas inteligentes o automáticos*, tienen como principal meta incorporar el enfoque distribuido y hacerlos más heterogéneos. Estos aspectos permiten una extensión larga de las capacidades de procesamiento de las computadoras sobre la red y la integración de diversos dispositivos que procesan señales, dando solución de escalabilidad y robustez a los sistemas de vigilancia. A continuación se mencionan algunos elementos que se incorporan y combinan en los sistemas automáticos: sensores, cámaras (fijas, IP, Pan Tilt Zomm PTZ), dispositivos móviles, redes alámbricas e inalámbricas de tipo LAN o WAN, dispositivos de red, uso de base de datos, servidores Proxys de procesamiento, entre otros [27]. Otros aspectos importante de estos sistemas son la diversidad temporal (solicitudes y respuestas en diferentes tiempos, asíncronos, paralelos), la distribución en la implementación de una arquitectura y la integración en tiempo real que usualmente son de tipo asíncrono.

En general, un *sistema inteligente o automático* debe estar compuesto por elementos de visión por computadora (uso de algoritmos para la detección y rastreo de algún objeto), poseer un control, contar con módulos de almacenamientos y recuperación, ser diseñados de manera secuencial y síncrona usando un módelo orientado a objetos y adaptarse en diferentes contextos [25].

El presente trabajo propone el diseño y desarrollo de un sistema distribuido de video-vigilancia automático que permita detectar movimiento y reconocer personas en tiempo real para integrarlo en dispositivos móviles con sistema operativo Android. El sistema de video-vigilancia distribuido propuesto llevará a cabo las fases principales del ciclo de

seguridad definido en [3]. En este caso, la fase de protección se implementará mediante el uso de un conjunto de cámaras localizadas en ciertas zonas, las cuales capturarán los hechos actuales. La fase de detección se llevará a cabo a través de la implementación de un algoritmo de detección de movimiento y reconocimiento de personas en tiempo real. Finalmente, en la fase de respuesta se considera la generación de alarmas si un intruso está dentro del área. La alarma es el aviso al usuario final mediante un mensaje multimedia o de texto a su dispositivo móvil. Además, el usuario desde su dispositivo podrá monitorear en tiempo real las zonas que abarcan el campo de visión de las cámaras.

## 1.2. Objetivo general

Diseñar y desarrollar un sistema distribuido de video-vigilancia que lleve a cabo la detección de movimiento y el monitoreo en tiempo real en zonas específicas a través de dispositivo móviles.

## 1.3. Objetivos específicos

- Diseñar una arquitectura de software en capas para el sistema de video-vigilancia propuesto en este trabajo de investigación.
- Detectar y reconocer movimiento de personas mediante el análisis de las imágenes en tiempo real para identificar situaciones anormales.
- Monitorear en tiempo real zonas específicas y generar alarmas a través de mensajes de texto y mensajes multimedia en caso de detectar situaciones anormales.

## 1.4. Justificación

El ser humano tiene la necesidad y el derecho a la seguridad así como a la protección. Sin embargo, este derecho se ve afectado actualmente con los altos índices de inseguridad y actos delictivos que atacan constantemente a la sociedad [8, 20, 36]. En consecuencia, las personas buscan la manera de cubrir esta necesidad con diversos medios, principalmente en la seguridad del individuo, familiar y la protección de sus propiedades. Algunos ejemplos de sistemas de seguridad son: las alarmas para autos y casas, sistemas de monitoreo mediante sensores y luces, sistemas de control satelital y sistemas de video vigilancia que permiten monitorear una área en especial, entre otros.

Actualmente se requieren sistemas de seguridad que logren el objetivo de reconocer, identificar y generar alarmas de ciertas situaciones anormales tales como personas no autorizadas en zonas específicas, objetos abandonados, conglomerados de masas, autos robados, entre otros [14]. En este contexto, los sistemas de video-vigilancia se han convertido en una herramienta efectiva para brindar estos servicios de seguridad a las personas [8, 16, 19, 26, 31]. Diversos trabajos han propuesto sistemas de *video-vigilancia* distribuidos [4, 14, 15, 19, 20, 22, 26, 27]. Estos trabajos se clasifican en analógicos, semiautomáticos y automáticos. Los sistemas analógicos propuestos funcionan con CTTV y señales analógicas lo cual genera ruido en el flujo de video evitando dotarlos de inteligencia (uso de algoritmos de visión por computadora para la detección) [24]. Por otra parte, los sistemas semiautomáticos solucionan las deficiencias de los analógicos; sin embargo, carecen de escalabilidad, velocidad de video, poder de procesamiento y algoritmos robustos para la detección [27]. Por otro lado, los sistemas automáticos propuestos [4, 15, 22] presentan deficiencias en la integración y comunicación de las diversas plataformas (multisensores, dispositivos móviles) y el diseño de una metodología que

definen [19, 24].

La meta de este trabajo es diseñar y desarrollar un sistema de video vigilancia distribuido VVD que realice el monitoreo y detecte el movimiento de personas no autorizadas (intrusos) en zonas específicas eliminando desventajas de trabajos propuestos previamente. En esta investigación, se propone para superar dichas desventajas el desarrollo de una arquitectura de software en capas. Esta arquitectura permitirá ocultar la heterogeneidad del sistema en términos de tipos de dispositivos de captura de video y redes de comunicación. En este caso, la integración de dispositivos móviles (Smartphone) al sistema distribuido propuesto permitirá, desde cualquier sitio y en cualquier momento, visualizar en tiempo real los sucesos que ocurren en una zona, así como administrar las cámaras correspondientes. Además, se plantea la ejecución de un algoritmo de visión por computadora sobre un móvil para realizar la detección (movimiento del intruso) de forma local, con el fin de informar al usuario final de lo sucedido por medio del envío de mensajes de texto o multimedia. En este proyecto se opta por dispositivos móviles tipo Smartphone debido a la demanda que menciona [37], indicando que afinales del 2011 se habrán vendido alrededor de 472 millones de tales dispositivos en el mundo comparado con la cantidad aproximada de 305 millones de unidades vendidas en 2010 y esperando que para el 2015 la cifra sea cercana a los 982 millones de nuevos usuarios que adquirirán un equipo de este tipo. Por otro lado, el dispositivo móvil seleccionado para el desarrollo del proyecto usa el sistema operativo Android evitando así el uso de licencias.

## 1.5. Contribuciones esperadas

El presente trabajo al concluirse habrá logrado aportar los siguientes aspectos:

- Arquitectura de software para el desarrollo de sistemas de video-vigilancia distribuidos sobre redes inalámbricas con infraestructura tales como las redes celulares.

- Implementación del algoritmo de detección de movimiento y detección de personas (cascadas de Haar) sobre un dispositivo móvil con sistema operativo Android.

- Monitoreo en tiempo real de zonas específicas y generación de alarmas a través de mensajes multimedia en caso de detectar situaciones anormales.

## 1.6. Cronograma de actividades

Tabla 1.1: Cronograma de actividades primer cuatrimestre

| | **Septiembre** | | | | **Octubre** | | | | **Noviembre** | | | | **Diciembre** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Actividades** | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 |
| Planteamiento del problema | * | * | * | | | | | | | | | | | |
| Edición de protocolo | | | * | * | * | | | | | | | | | |
| Presentación de protocolo | | | | | * | | | | | | | | | |
| Elaboración de marco teórico | | | | | | * | * | * | * | | | | | |
| Diseño y desarrollo de la arquitectura del SVVD. | | | | | | | | | * | * | | | | |
| Edición de Metodología. | | | | | | | | | | | * | * | | |
| Presentación de avances. | | | | | | | | | | | | | * | |
| Revisión de literatura. | * | * | * | * | * | * | * | * | * | * | * | * | * | * |

* SVVD Sistema de Video-Vigilancia Distribuido

Tabla 1.2: Cronograma de actividades segundo cuatrimestre

| | Dic. | Ene. | Ene. | Ene. | Ene. | Feb. | Feb. |
|---|---|---|---|---|---|---|---|
| **Primer bloque de entrega Actividades** | S 15-21. | S 2-8 | S 9-15 | 16-22 | S 23-29 | S 30-5 | S 6-12 |
| **Etapa uno PUDS (Servidor-Cámaras) Monitoreo.** Mecanismo de captura de imágenes (flujo de video) en tiempo real y envió de este flujo al usuario final (Servidor de flujo de video). | | | | | | | |
| Investigar funcionamiento de Servidor Streaming y protocolos (Generar comparativo). Modelo de Análisis (Casos de Uso y Diagrama Preliminar de Clases). | * | | | | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | * | | | | | | |
| Modelo de implementación (Diagramas de Componentes). | * | | | | | | |
| +Instalación de servidor Streaming (Darwin o Wowza) | * | | | | | | |
| +Captura de video de una cámara web | | * | | | | | |
| +Uso del servidor de Streaming para enviar flujos de video a un usuario móvil | | * | | | | | |
| Modelo de pruebas. | | | * | | | | |
| Documentar resultados. | | | * | | | | |
| **Etapa dos PUDS (Servidor-Cámaras) Gestionar sistema.** Desarrollo de una aplicación Web para usuario móvil final donde este es capaz de realizar altas, bajas, actualizaciones y eliminar la configuración de vigilancia (activación de los algoritmos de VC, tiempos), usuarios y evidencias (fotos, videos). . | | | | | | | |
| Investigar funcionamiento del framework Struts bajo el patrón MVC (Model View Control) para desarrollo de aplicaciones Web (Servlets y JSP). Modelo de Análisis para la configuración de vigilancia, usuarios y evidencias (Casos de Uso y Diagrama Preliminar de Clases). | | | | * | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | | | | * | * | * | |
| Modelo de implementación (diagramas de componente) 1.Instalación de servidor web, 2.Instalación del Framework, 3. Programación. . | | | | | * | * | |
| Modelo de pruebas. . | | | | | | | * |
| Documentar resultados. . | | | | | | | *P |
| Revisión de literatura. | * | * | * | * | * | * | * |

* S=Semana P=Presentación de Avances

PUDS = Proceso Unificado de Desarrollo de Software

Tabla 1.3: Cronograma de actividades segundo cuatrimestre

| | Feb. | Feb. | Mar. | Mar. | Mar. | Mar. | Abr. |
|---|---|---|---|---|---|---|---|
| **Segundo bloque de entrega Actividades** | S 13-19 | S 20-25 | S 27-3 | S 5-11 | S 12-18 | S 19-25 | S 26-1 |
| **Etapa tres PUDS (Servidor-Cámaras).** Algoritmo de detección de movimiento (motion template) y detección de personas (clasificador de cascadas de Haar) utilizando JavaCV además de permitir almacenar evidencias (imagen y secuencia de video). . | | | | | | | |
| Investigar el almacenamiento de material multimedia a la base de datos, funcionamiento y uso de JavaCV para programar los algoritmos de detección de movimiento y de personas (generar Base de datos de imágenes de personas para entrenar el algoritmo de detección de personas y obtener un nuevo archivo XML ). Modelo de análisis (Casos de Uso y Diagrama Preliminar de Clases). | * | | | | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | * | | | | | | |
| Modelo de implementación (diagramas de componente). 1.Instalación de JavaCV, 2.Implementar Algoritmos e integrarlos . | * | * | * | | | | |
| Modelo de pruebas. | | | * | | | | |
| Documentar resultados. | | | * | | | | |
| **Etapa cuatro PUDS (Servidor-Cámaras)** Generar Alarmas de tipo mensaje de texto y mensaje multimedia. | | | | | | | |
| Investigar servidores para administrar mensajes (recepción y envió) de tipo MSM y MMS. (clickatell o Kannel) Modelo de análisis (Casos de Uso y Diagrama Preliminar de Clase). | | | | * | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | | | | | * | | |
| Modelo de implementación (diagramas de componente) 1.Instalación de servidor de mensajes, 2.Envió y recepción de mensajes tipo texto y multimedia . | | | | | | * | |
| Modelo de pruebas. | | | | | | | * |
| Documentar resultados. | | | | | | | *P |
| Revisión de literatura. | * | * | * | * | * | * | * |

* S=Semana P=Presentación de Avances

PUDS = Proceso Unificado de Desarrollo de Software

Tabla 1.4: Cronograma de actividades segundo y tercer cuatrimestre

| | Abr. | Abr. | Abr. | Abr. | May. | May. | Mar. |
|---|---|---|---|---|---|---|---|
| **Segundo bloque de entrega Actividades** | S 2-8 | S 9-15 | S 16-22 | S 23-29 | S 30-6 | S 7-13 | S 14-20 |
| **Etapa cinco PUDS (dispositivo móvil).** Algoritmo de detección de movimiento (motion template) y detección de personas (clasificador de cascadas de Haar) utilizando Android además de permitir almacenar evidencias (imagen y secuencia de video) en el servidor. | | | | | | | |
| Investigar el almacenamiento de material multimedia a la base de datos en el servidor desde el dispositivo móvil. Funcionamiento y uso de Android para programar los algoritmos de detección de movimiento y de personas además de utilizar el archivo XML ya generado anteriormente para detectar personas. Modelo de análisis (Casos de Uso y Diagrama Preliminar de Clases). | * | | | | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado) (diagrama detallado y secuencias detallado). | | * | | | | | |
| Modelo de implementación (Diagramas de Componente) 1.Instalación de eclipse y Android, 2.Implementar Algoritmos e integrarlos, 3.Almacenar evidencias obtenidas por el móvil hacia el servidor. | | * | * | * | | | |
| Modelo de pruebas. | | | | * | | | |
| Documentar resultados. | | | | * | | | |
| **Etapa seis PUDS (dispositivo móvil).** Transmisión de video (entre dispositivos móviles o usando el servidor de Streaming) para el monitoreo. | | | | | | | |
| Investigación y análisis de la viabilidad para la transmisión de video entre dos dispositivos móviles o uso del servidor de Streaming. Modelo de (Casos de Uso y Diagrama Preliminar de Clases). | | | | | * | * | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | | | | | | * | |
| Modelo de implementación (diagramas de componente) . | | | | | | | *P |
| Revisión de literatura. | * | * | * | * | * | * | * |

* S=Semana P=Presentación de Avances

PUDS = Proceso Unificado de Desarrollo de Software

Tabla 1.5: Cronograma de actividades tercer cuatrimestre

| | May. | Jun. | Jun. | Jun. | Jun. | Jul. | Jul. |
|---|---|---|---|---|---|---|---|
| **Tercer bloque de entrega Actividades** | S 21-27 | S 28-3 | S4-10 | S 11-17 | S 18-24 | S 25-1 | S 2-8 |
| Modelo de pruebas. . | * | | | | | | |
| Documentar resultados. . | * | | | | | | |
| **Etapa siete PUDS (dispositivo móvil).** Generar alarmas de tipo mensaje de texto y mensaje multimedia, emitidos desde un dispositivo móvil. | | | | | | | |
| Usar el servidor para administrar mensajes (recepción y envió) de tipo MSM y MMS que interactúa entre dos dispositivos móviles. Modelo de análisis (Casos de Uso y Diagrama Preliminar de Clases). | | * | | | | | |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | | | * | | | | |
| Modelo de implementación (Diagramas de Componente). | | | | * | * | | |
| Modelo de pruebas. | | | | | * | | |
| Documentar resultados. | | | | | * | | |
| **Etapa ocho PUDS (dispositivo móvil).** Administración de la cámara del dispositivo móvil a partir del servidor Web que ha procesado la petición de un cliente móvil. (Configuración de vigilancia y evidencias) y el desarrollo de una aplicación nativa para la comunicación entre el servidor Web y el móvil de detección de movimiento y de personas. | | | | | | | |
| Investigar el desarrollo de aplicaciones nativas en Android y la comunicación de servidor-móvil mediante la red con el objetivo de administrar la configuración de vigilancia y evidencias. Modelo de análisis (Casos de Uso y Diagrama Preliminar de Clases). | | | | | | * | * |
| Revisión de literatura. | * | * | * | * | * | * | * |

* S=Semana P=Presentación de Avances

PUDS = Proceso Unificado de Desarrollo de Software

Tabla 1.6: Cronograma de actividades tercer cuatrimestre

| | Jul | Jul. | Jul. | Agos. | Agos. |
|---|---|---|---|---|---|
| **Tercer bloque de entrega Actividades** | S 9-15 | S 16-22 | S 23-29 | S 30-5 | S 6-31 |
| Modelo de diseño (Diagrama de secuencias y de Clases Detallado). | * | | | | |
| Modelo de implementación (Diagramas de Componente) . | | * | * | | |
| Modelo de pruebas. | | * | * | | |
| Documentar resultados. | | | * | | |
| Entrega de documento final. | | | | * | |
| Edición de un artículo y presentanción final. | | | | * | *P |

* S=Semana P=Presentación de Avances

PUDS = Proceso Unificado de Desarrollo de Software

# Capítulo 2

# Marco teórico

## 2.1. Sistemas distribuidos

### Definición de sistema distribuido

“ Un sistema distribuido es una colección de entidades independientes que cooperan para resolver un problema que no pueden solucionarlo de manera individual”, definido de manera general por [41].

En relación a la computación, los avances tecnológicos de las computadoras y de las redes de comunicación a partir de la mitad de la década de 1980, logró que las computadoras (con cierto poder de procesamiento) junto con la ayuda de las redes (LAN, MAN y WAN) pudieran comunicarse entre sí (transferir información), surgiendo así los sistemas distribuidos (SD) para los sistemas computacionales. El principal propósito es comunicar y coordinar una gran cantidad de estas entidades con la finalidad de compartir recursos (hardware HW y/o software SW), además de que los usuarios perciban al sistema como uno sólo y de manera trasparente. Tres ejemplos comunes de los sistemas distribuidos son:

- Internet

- Una intranet (una red privada que forma parte de internet y que es administrada por una organización)
- Red de telefonía móvil

Por lo tanto, un sistema distribuido en computación se define como un conjunto de componentes HW o SW autónomos conectados a través de una red, que se comunican y coordinan sólo por el paso de mensajes [5, 41, 2]. Un SD tiene las siguientes características [5, 41]:

1. **Carecen de una referencia global de tiempo:** No hay un tiempo global en un SD, debido a que los relojes en los diferentes componentes no tienen necesariamente el mismo tiempo entre ellos, así que es necesario de métodos y algoritmos para la comunicación y coordinación.
2. **No hay memoria compartida:** En un SD cada elemento de la red cuenta con una memoria local.
3. **Tolerancia a fallos independientes:** El SD es capaz de mantenerse en funcionamiento a pesar de que éste haya tenido un avería o fallo en alguno de sus componentes sin que ésto afecte a todo el sistema.
4. **Concurrencia:** El SD permite compartir recursos de tal manera que más de un usuario (componente) lo requiera en determinado tiempo.

Finalmente, un SD permite mayor flexibilidad que una computadora personal o sistema centralizado debido a que permite distribuir la carga de trabajo entre los elementos que lo componen, estos componentes no siempre son computadoras (heterogeneidad), además de que poseen diferentes características y capacidades (unidades de procesamiento y memoria local) [5]. La Figura 2.1 ilustra un SD integrando por diversas plataformas con el objetivo de compartir recursos y distribuir tareas..

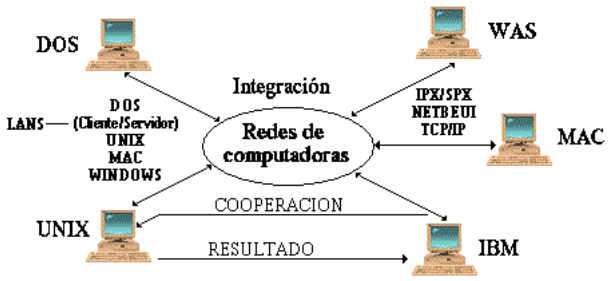



Figura 2.1: Ejemplo de un SD.

### Motivación

Los aspectos de mayor importancia para la implementación de un SD son los siguientes [41]:

**Distribución de tareas a procesar**: En un SD las aplicaciones (SW o servicios) necesitan repartir o distribuir la carga de trabajo entre muchos componentes (las tareas se procesan de manera distribuida, en relación al contexto de la tarea o lógica del negocio, ejemplo bancos, supermercados, etc.).

**Recursos compartidos:** El SD permite tener diversos recursos (HW, SW, base de datos) disponibles para poder ser accedidos o utilizados por las entidades que no cuentan con ellos, (una opción para evitar los cuellos de botella por el uso de recursos compartidos son las réplicas).

**Acceso remoto a datos y recursos:** El SD integra una gran gama de elementos o entidades conectados mediante una red, ejemplo internet (WAN), lo que permite una gran dispersión geográfica. Por lo tanto, los elementos pueden accederse o solicitarse de manera remota.

**Fiabilidad:** En el SD se observa fiabilidad en la réplica de recursos y datos, así también en la ejecución remota de acuerdo a los siguientes aspectos:

- **Disponibilidad:** El recurso compartido debe permitir el acceso todo el tiempo.

- **Integridad:** Los valores /estados de los recursos y datos deben ser correctos (no debe sufrir alteraciones o corrupción).
- **Tolerancia:** Si un elemento del SD falla, éste no debe afectar el funcionamiento del sistema, debe recuperase y continuar su tarea.

**Escalabilidad:** Los sistemas distribuidos operan efectiva y eficientemente en muchas escalas (intranet o internet). Se dice que un sistema es escalable si conserva su efectividad aún cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.

**Modularidad y extensibilidad:** En el SD se pueden añadir nuevos elementos de procesamiento o servicios para compartir recursos, permitiendo un desarrollo gradual según las necesidades.

### 2.1.1. Modelos de un sistema distribuido

#### Modelos arquitecturales

El modelo arquitectónico de un SD, simplifica y abstrae las funciones de sus componentes individuales, enseguida define la ubicación de éstos dentro de la red y la interrelación (el papel que tiene cada uno de ellos y la comunicación) [5].

#### Capas de software

La *arquitectura de software* se concibe para una única computadora como la estructuración del SW en capas o módulos, cuando se tiene un conjunto de computadoras con procesos locales y remotos que hace solicitudes y en donde se obtienen respuestas de ciertos servicios, se expresan o definen también como capas de servicio [5]. La Figura 5.1 presenta un conjunto de capas de servicio de HW y SW para un sistema distribuido.

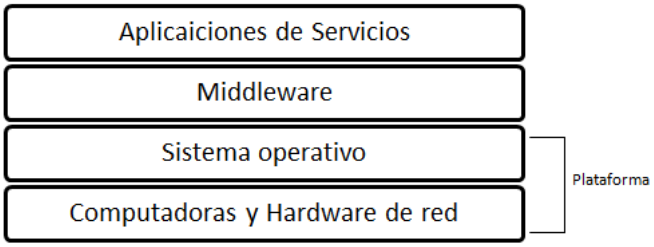



Figura 2.2: Capas de servicio de SW y HW en los sistemas distribuidos [5].

Dos aspectos de interés para este trabajo son la plataforma y el middleware, que son descritos enseguida:

**Plataforma:** Las dos primeras capas de la Figura 5.1 forman la plataforma para un SD y proporcionan servicios a las capas continuas, tanto el SO como la arquitectura de HW son independientes de cada entidad (computadora, dispositivo móvil), ejemplos Windows, Linux y Solaris para intel X86, Sun OS para Sun SPARC y Mac OS para Power PC.

**Middleware:** Es una capa de software (estrato de software) de conectividad que hace posible que las aplicaciones distribuidas puedan ejecutarse sobre distintas plataformas (heterogeneidad), es decir sobre diferentes arquitecturas de HW, sistemas operativos (SO), protocolos de red (redes) y lenguajes de programación [5, 7]. Ejemplos de middleware orientados a objetos son CORBA (llamadas a procedimientos remotos) y Java RMI (invocación de objetos remotos). Uno de los objetivos de un middleware es ofrecer un acuerdo entre las interfaces y los mecanismos de interoperabilidad, debido a las diferencias de hardware, sistemas operativos, protocolos de redes, y lenguajes de programación.

**Arquitecturas del sistema**

En los sistemas distribuidos el aspecto de diseño que se refiere a la división de responsabilidades entre los componentes (aplicaciones, servidores, computadoras, procesos, entre otros) y a la ubicación de los componentes sobre la red, tomando en cuenta la presentación, la fiabilidad y la seguridad del sistema [5], enseguida se definene cuatro arquitecturas:

- **Modelo cliente-servidor:** Esta arquitectura consiste en tener procesos servidores y clientes. Los clientes realizan solicitudes (peticiones) al servidor éste último toma la petición realiza o procesa cierta tarea solicitada y regresa una respuesta al cliente.

- **Servicio proporcionado por múltiples servidores:** Los servicios se implementan en distintos procesos servidores (distribuye o se replica el servicio en diferentes computadoras) que se encuentran separados e interconectados a través de la red, este conjunto de servidores proporcionaran más de un servicio a los procesos clientes.

- **Servidores proxy y caches**: Una caché es un almacén de objetos de datos recientemente utilizados y que se encuentran próximos a los procesos. Si un objeto nuevo es recibido en una computadora, se añade al almacén caché o se remplaza si es necesario. Cuando un proceso cliente requiere un objeto, el servicio caché comprueba inicialmente en su caché y proporcionandole el objeto actualizado, de lo contrario buscaría una copia del objeto actualizado. Las cachés pueden estar ubicadas en el cliente (ejemplo en los navegadores web) o en un servidor proxy que se comparte entre varios clientes. Los servidores proxy para web proporcionan una caché compartida de recursos entre las entidades clientes de uno o más sitios.

- **Proceso de igual a igual (*peer to peer*):** En esta arquitectura todos los procesos se desempeñan cooperativamente como iguales o semejantes para realizar una actividad distribuida (no hay una distinción entre cliente o servidor, ambos papeles son desempeñados por cada proceso).

### Modelos fundamentales

De manera general, un modelo contiene sólo aquellos elementos necesarios para comprender y razonar sobre algunos aspectos del comportamiento del sistema [5].
El objetivo de un modelo es:

1. Hacer claras todas las premisas relevantes sobre los sistemas que se encuentra modelando.

2. Hacer generalizaciones respecto a lo que es posible o no, tomando en cuenta las premisas anterioresm,(las generalizaciones pueden ser algoritmos de propósito general o particular).

En los siguientes apartados se mencionará sólo dos modelos para SD, el modelo de iteración y el de fallos.

### Modelos de interacción

La *interacción* proviene principalmente del cómputo que ocurre entre los procesos. Es decir, los procesos interactúan sólo por el paso de mensajes, dando lugar a la comunicación (flujo de información) y a la coordinación (sincronización y ordenamiento de las actividades) entre procesos [5]. Existen dos variantes de modelos de interacción:

1. **Síncronos (fuerte restricción del tiempo).** Un SD es síncrono si cumple lo siguiente:

- El tiempo de ejecución de cada etapa de un proceso tiene ciertos límites conocidos (límite inferior y superior).
- Cada mensaje transmitido por un canal se recibe en un tiempo límite.
- Cada proceso tiene un reloj local, cuya tasa de deriva sobre el tiempo real tiene un límite conocido.

Se sugiere un tiempo límite superior e inferior (un *timeout*) en la ejecución de un proceso, los retardos de mensajes y las tasas de deriva de los relojes.

2. **Asíncronos.** En un sistema distribuido asíncrono no es necesario colocar intervalos de tiempos para que funcione adecuadamente por lo tatno no exigen limitanción en la Velocidad de procesamiento, Velocidad de procesamiento, Retardos de trasmisión de mensajes y Tasa de deriva de reloj.

### Modelos de fallos

Los fallos en un sistema distribuido se pueden dar tanto en los canales de comunicación como en los proceso. Hadzilacos y Toueg en [5] señalan tres maneras de conceptualizar a los fallos; por 1) omisión, 2)arbitrarios y 3)de temporalización. Se presenta cada uno enseguida:

- **Fallos por omisión:** Se refiere a los casos donde los procesos o canales de comunicación no consiguen realizar las acciones que pueden hacer o tendrían que hacer.

  1. **Fallos por omisión de procesos:** Se refiere cuando un proceso tiene una avería accidental del procesamiento (*crash*), esto quiere decir que el proceso se ha parado y no ejecutará ningún paso más del programa. La solución es

el uso de *timeouts* para sistemas síncronos, para los asíncronos se desconoce si el proceso está parado, lento o roto.

2. **Fallos por omisión de comunicaciones:** El canal de comunicación produce un fallo entre dos procesos $p$ y $q$, si no se trasporta el mensaje desde el búfer de mensajes de salida de $p$ al búfer de mensajes de entrada de $q$, a este fallo se le denomina *pérdida de mensajes*. La causa puede ser la falta de espacio en el buffer de cualquiera de los dos procesos. Aquellos fallos en donde el mensaje se pierde entre el proceso emisor y el búfer de salida se denomina *fallo por omisión de envío*, por otro parte la pérdida de mensajes entre el búfer de mensajes de estrada y el proceso receptor se llama *fallo por omisión de recepción* y la perdida de mensajes entre los búfers como *fallos por omisión de canal*.

- **Fallos arbitrarios:** Este fallo ocurre por errores de semántica que da cavida a que pueda ocurrir cualquier otro tipo de error. Por ejemplo, un proceso puede escribir o responder mensajes con datos equivocados para una solicitud o una respuesta. Los canales de comunicación también sufren de estos fallos, por ejemplo, un mensaje puede ser corrupto o pueden repetirse mensajes inexistentes.

- **Fallos de temporalización:** Este tipo de fallo es exclusivo para los SD síncronos, ya que en éstos definen límites de tiempo para los procesos, relojes y el reparto de mensajes. Este fallo se presenta cuando se excede el tiempo límite.

**Enmascaramiento de fallos:** El conocimiento de las características de un fallo de un componente puede permitir crear un servicio nuevo diseñado para enmascarar ese fallo (ocultar o generar un fallo aceptable).

**Fiabilidad y comunicación uno a uno:** La comunicación fiable consiste en definir términos de validez e integridad. El primer concepto establece que cualquier mensaje

enviado por su emisor se debe garantizar la entrega al receptor (sin pérdidas). El segundo concepto menciona que si un mensaje es recibido, debe ser idéntico al enviado y sin duplicados.

### 2.1.2. Comunicación interprocesos

#### Características de la comunicación interprocesos

El paso de mensajes entre dos proceso se basa en las operaciones *envía* y *recibe*. Si un proceso desea comunicarse con otro, el proceso emisor envía un mensaje (secuencia de bytes) a un destino y el proceso destino (receptor) recibe el mensaje. Esta actividad implicaría la comunicación y la coordinación de los dos procesos, por lo tanto se define lo siguiente [5]:

**Comunicación síncrona y asíncrona.** Cada proceso receptor asocia una cola para los mensajes, cuando el proceso emisor genera mensajes éstos serán añadidos a la cola, por lo que la comunicación entre estos dos procesos puede ser síncrona y asíncrona.

- **Síncrona.** Tanto el proceso receptor como emisor se sincronizan con cada mensaje, por lo tanto las operaciones *envía* y *recibe* son operaciones *bloqueantes.* Es decir, que para cada *envía* producido por el emisor, éste se bloquea hasta que se produce el evento recibe. Cuando se invoca, *recibe* el proceso se bloque hasta la llegada del mensaje.

- **Asíncrona.** Utiliza una operación *envía* de tipo no bloqueante de modo que el proceso emisor continua sus actividades después de que el mensaje es copiado en el búfer local. La operación recibe tiene la variante *bloqueante* y *no bloqueante*, en esta última el proceso receptor continua sus tareas después de invocar la operación

recibe que proporciona un búfer que será llenado, si éste es llenado, el proceso debe ser avisado (método de encuesta o interrupción).

**Destinos de los mensajes.** En los protocolos de internet, los mensajes son enviados a las direcciones construidas por pares (*dirección de internet, puerto local*), un puerto local (un número entero) es el destino de un mensaje dentro de una computadora. Los procesos pueden usar varios puertos desde los cuales se reciben los mensajes, entonces cualquier proceso que conozca el número de puerto apropiado puede enviar un mensaje.

**Fiabilidad y orden.** La fiabilidad consiste en aspectos de validez e integridad que fueron tratado en los modelos de fallos, en relación al orden, diversas aplicaciones requieren que la entrega de mensajes sea de acuerdo a la manera en que se realizó su emisión.

### Arquitectura de comunicación con ICE

Un middleware que ha revolucionado a los sistemas distribuidos a partir de los 90´s y hasta nuestros días es CORBA, que estableció una arquitectura base para la comunicación de los sistemas distribuidos y el modelo orientado a objetos. CORBA tuvo una implementación rápida en la industria, en el ámbito profesional y científico, pero los resultados obtenidos fueron limitados debido a los aspectos siguientes [23]:

- Complicado de aprender
- Alta complejidad para usarse
- Los protocolos implementados ineficientes que no permiten un rediseño
- Falta de soporte
- Falta de interfaces múltiples

Retomando las fortalezas y evitando las debilidades de CORBA, surgió el middlaware ICE (Internet Communication Engine), que posee una facilidad de uso, alto desempeño, una mejora en la escalabilidad entre otras ventajas. Por lo tanto, ICE es un middleware orientado a objetos que proporciona herramientas, APIs (*Application Programming Interface*) y el soporte de bibliotecas para construir aplicaciones cliente-servidor orientadas a objetos [7, 23].

Una aplicación ICE se emplea en ambientes heterogéneos, en donde los clientes-servidores pueden programarse en diferentes lenguajes de programación, además de poseer distintos sistemas operativos y diferentes arquitecturas capaces de comunicarse empleando diferentes tecnologías de red [7, 23, 30].
Los propósitos de ICE son los siguientes:

- Proporcionar un middleware listo para usarse en sistemas heterogéneos.
- Proveer un conjunto completo de características que soporten el desarrollo de aplicaciones distribuidas reales en un amplio rango de dominios.
- Evitar una complejidad innecesaria haciendo que ICE sea fácil de aprender y de usar.
- Proporcionar una implementación eficiente en ancho de banda, en uso de memoria y en carga de CPU.
- Proporcionar una implementación basada en la seguridad (hilos seguros, encapsulación), de forma que se pueda usarse sobre redes no seguras.

**Concepto proxy y esqueleto de ICE**

Generalmente los middlewares tratan y visualizan el modelo de programación de forma

local, enmascarando la llamada a los procedimientos remotos. Para ello se crea un proxy en la parte del cliente (que actúa como intermediario con el servidor) y un esqueleto en lado del servidor (que traduce los eventos de la red a invocaciones sobre un objeto).

**Objeto ICE**

Un objeto ICE es una entidad conceptual o una abstracción que mantiene una serie de características de red [23]:

- Un objeto ICE es una entidad en el espacio de direcciones remoto o local capaz de responder a las peticiones de los clientes.
- Un único objeto ICE puede instanciarse en un único servidor o de manera redundante (múltiples servidores).
- Cada objeto ICE tiene una o más interfaces. Una interfaz es una colección de operaciones soportadas por un objeto. Los clientes invocan dichas operaciones (peticiones).
- Una operación tiene cero o más parámetros y un valor de retorno. Los parámetros y los valores de retorno tienen un tipo específico. Además, los parámetros tienen una determinada dirección: los parámetros de entrada se inicializan en la parte del cliente y se pasan al servidor y los parámetros de salida se inicializan en el servidor y se pasan a la parte del cliente.
- Un objeto ICE tiene una interfaz distinta del resto y conocida como la interfaz principal, ICE puede proporcionar cero o más interfaces alternativas, conocidas como facetas (*facets*).
- Cada objeto ICE tiene una identidad de objeto única. La identidad de un objeto es un valor que distingue a un objeto del resto de los objetos.

**Proxies**

Para que un cliente sea capaz de comunicarse con un objeto ICE, debe acceder a un proxy. Un proxy es un componente local al espacio de direcciones del cliente que representa al cliente que requiere el objeto ICE de la red [7, 23].

El proxy tiene el rol de embajador local de un objeto ICE, de manera que cuando un cliente invoca una operación en el proxy, el núcleo de ejecución de ICE realiza un proceso para localizar el objeto ICE (en el servidor), trasmitir los parámetros de entrada, esperar que la operación se complete y devolver el parámetro de salida (valor de retorno para el cliente o un error). La información de este proceso es encapsulada por el proxy.

**Replicación**

La replicación en ICE implica hacer disponibles en direcciones múltiples a los adaptadores de objetos. El objetivo principal de la replicación es proporcionar redundancia ejecutando el mismo servidor en distintas máquinas. Si en una de ellas se produce un error, el servidor permanece disponible en el resto (las aplicaciones desarrolladas deben estar bajo este mismo concepto).

**Sirvientes *(Servants)***

El componente en el lado del servidor que proporciona el comportamiento asociado a la invocación de las operaciones se denomina sirviente. Un sirviente representa a uno o más objetos ICE.

**Semántica *at-most-once***

Las solicitudes ICE tienen una semántica *at-most-once*: el núcleo de ejecución de ICE

hace todo lo posible para entregar una solicitud al destino correcto y dependiendo de las circunstancias, puede volver a intentar una solicitud en caso de fallo.

ICE garantiza que entregará la solicitud o en el caso de no realizarlo, informará al cliente con una determinada excepción. Bajo ninguna circunstancia una solicitud se entregará dos veces, es decir, los reintentos se generarán sólo si se conoce con certeza que un intento previo falló.

La invocación de métodos en ICE son de dos formas:

1. Invocación de métodos síncrona: Se comporta como una llamada a un procedimiento remoto. Ejemplo el hilo del cliente se suspende mientras dura la llamada y se reactiva cuando la llamada se ha completado.

2. Invocación de métodos asíncrona (AMI en inglés): Un cliente puede invocar operaciones de manera asíncrona. Ejemplo un cliente usa un proxy para invocar una operación y hace uso de un objeto retrollamada (*callback object*).

El servidor no es capaz de diferenciar entre una invocación síncrona de una asíncrona, para ello realiza un tratamiento de métodos, para síncrono el servidor se libera (desbloquea) cuando la operación se ha completado, en el tratamiento de métodos asíncronos consiste de una gran cantidad de clientes que podrían bloquearse en la misma invocación sin estar vinculados a los hilos del servidor (el tratamiento síncrono o asíncrono es trasparente para el cliente y no sabe si el servidor utiliza alguno de estos tratamientos).

**SLICE (Lenguaje de especificación para ICE)**

La comunicación o acuerdos entre los componentes (cliente-servidor) la provee ICE a través de las interfaces definidas bajo el lenguaje SLICE, que define la manera de comunicación (parámetros, operaciones y tipos de datos) usados entre componentes de

red [7, 23]. SLICE permite establecer el acuerdo entre el cliente y el servidor de forma independiente del lenguaje de programación empleado.

**Mapeo de lenguajes**

Las reglas que rigen cómo se traslada cada construcción SLICE en un lenguaje de programación específico se conocen como mapeop (*mappings*) de lenguajes.

Con el objetivo de determinar qué aspecto tiene la API generada para un lenguaje determinado, sólo se necesita conocer la especificación en SLICE y las reglas de mapping hacia dicho lenguaje. Actualmente, ICE proporciona mappings para los lenguajes C++, Java, C♯, Visual Basic .NET, Python, PHP y Ruby [7, 23].

**Protocolo ICE**

ICE proporciona un protocolo RPC que puede utilizar tanto TCP/IP (Streams) como UPD (datagramas). Además, de permitir usar SSL, de forma que todas las comunicaciones entre el cliente y el servidor se encuentren encriptados [7, 23]. El protocolo ICE define:

- Un determinado número de tipos de mensajes, ejemplo mensaje de solicitud y respuesta.
- Una máquina de estados qué determina que secuencia siguen los distintos tipos de mensajes que se intercambian entre el cliente y el servidor.
- Reglas de codificación que determinan cómo se representa cada tipo de datos en la red.
- Una cabecera para cada tipo de mensaje que contiene detalles como el tipo del mensaje, el tamaño del mensaje, el protocolo y la versión de codificación emplea-

dos.

- ICE también soporta compresión en la red: ajustando un parámetro de configuración se pueden comprimir todos los datos asociados al tráfico de red para reducir el ancho de banda utilizado. Esta propiedad resulta útil si la aplicación intercambia grandes cantidades de datos entre el cliente y el servidor.

El protocolo ICE también soporta operaciones bidireccionales.

### 2.1.3. Sistemas distribuidos de video-vigilancia

La seguridad es una de las necesidades del hombre (seguridad individual y de sus propiedades) que durante siglos ha tratado de satisfacer y resolver mediante la tecnología y diversos métodos, uno de ellos son los sistemas de video-vigilancia que utilizan la valiosa información proporcionada de una secuencia de video.

A continuación se mencionan tres generaciones de los sistemas de video-vigilancia de acuerdo a los avances tecnológicos y las necesidades de tener sistemas más inteligentes.

#### Primera generación

También conocida como *sistemas analógicos* (SA), debido a la implementación de la tecnología de circuitos cerrados de televisión CCTV de tipo analógico. Los SA consisten de un número de cámaras colocadas en múltiples regiones remotas y conectadas a un conjunto de monitores a través de interruptores automáticos (switches), usualmente los monitores se localizan en un único centro de control (monitoreo y almacenamiento de video) [24, 25, 27]. La Tabla 2.1 presenta las ventajas, desventajas y los retos de estos sistemas.

Tabla 2.1: Ventajas, desventajas y retos de los sistemas analógicos

| Ventajas | Desventajas | Retos |
|---|---|---|
| -Monitoreo a regiones remotas y de riesgo (volcanes, en industrias)<br><br>- Uso de cámaras digitales | -La existencia de un único punto de control o monitoreo<br><br>-Demasiado caro e ineficiente debido a la gran cantidad de personal (supervisores) cuando el sistema crece.<br><br>-La conversión de datos digitales a analógicos generan ruido y degradación de las imágenes. | -Compresión de video para CCTV.<br><br>-Usos de la tecnología digital enel sistema. |

### Segunda generación

Como mejora de la primera generación surgen los *sistemas semiautomáticos* ( SS o segunda generación) los cuales combinan a la visión por computadora con los sistemas de CCTV con el propósito de procesar imágenes y señales. Estos sistemas se basan en la búsqueda y creación de algoritmos de detección y rastreo de objetos de interés en tiempo real, también de la necesidad de desplegar mensajes relevantes (alarmas, recuadros de color) de la existencia de los objetos de interés sobre el monitor, esto es de gran ayuda para el centro de monitoreo [24, 25, 27]. La Tabla 2.2 presenta las ventajas, desventajas y retos de la segunda generación.

Tabla 2.2: Ventajas, desventajas y retos de los sistemas semiautomáticos

| Ventajas | Desventajas | Retos |
|---|---|---|
| -Combinación de visión por computadora y los CCTV.<br><br>-Aumento en la eficiencia de vigilancia en los sistemas CCTV. | - Escalabilidad limitada.<br><br>-Procesamiento de los algoritmos es caro. | -Búsqueda de algoritmos robustos de visión por computadora (VC) y aprendizaje automático (AA) en tiempo real.<br><br>-Reconocimiento de patrones y comportamientos |

### Tercera generación

Tras necesidad de diseñar sistemas que cubran el monitoreo de áreas amplias, de tipo distribuido y heterogéneos (redes diferentes, cámaras, sensores etc.) , además que cumplan con las cuestiones semiautomáticas (algoritmos de VC y AA para el procesamiento de imágenes) surgen los *sistemas automáticos e inteligentes* SA (tercera generación). Estos sistemas se caracterizan por ser concurrentes, distribuidos, integrados y de tiempo real. Su diseño generalmente está orientado a objetos y de tipo asíncrono, integran módulos de control de dispositivos, almacenamiento y recuperación de video, entre otros, [19, 24, 25, 27].

El objetivo de los SA es obtener la descripción de lo que sucede en una área monitoreada y automáticamente tomar una acción como las alertas que permiten dar aviso a un usuario final (centro de monitoreo o un individuo con dispositivo móvil).

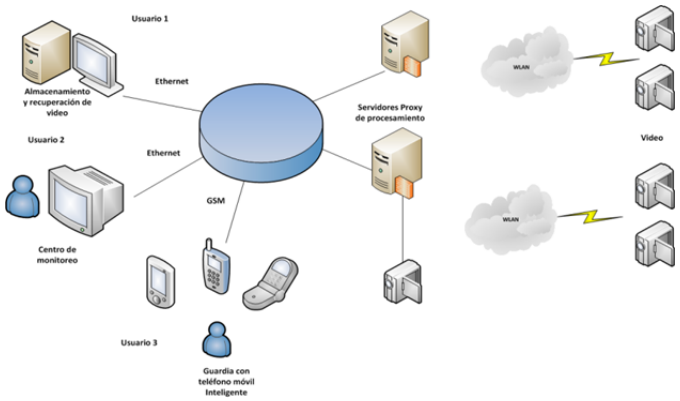



Figura 2.3: Representación de un sistema automático e inteligente.

La Figura 2.3 muestra un sistema distribuido de video-vigilancia SDV de tipo automático, el cual resalta la parte distribuida y heterogénea,se compone de distintos dispositivos como cámaras que captan el video que es transmitido a los servidores proxys que realizan su procesamiento e identifican situaciones anormales, si existe una de

ellas, se generan alarmas de aviso a sus distintos usuarios [27].

En la Tabla 2.3 se mencionan las ventajas, desventajas y retos del SA.

Tabla 2.3: Ventajas, desventajas y retos de los sistemas automáticos

| Ventajas | Desventajas | Retos |
|---|---|---|
| -Integración de VC, AA, sistemas distribuidos, heterogéneos.<br><br>- Escalabilidad alta.<br><br>-Robustez. | -Cuellos de botella y saturación de los dispositivos.<br><br>-Distribución de la información.<br><br>-Inexistencia de un middleware de uso general para su diseño. | -Seguridad del sistema.<br><br>-Fusión o integración de los datos |

### 2.1.4. Sistema operativo Android

*Android* es un conjunto de software para dispositivos móviles que incluye un sistema operativo, un middleware y un conjunto de aplicaciones clave. Cuenta con un kit de desarrollo denominado SDK Android que provee las herramientas y API´s necesarias para desarrollar aplicaciones usando el lenguaje de programación JAVA [30, 1].

#### Arquitectura de Android

La arquitectura de Android se basa en el *kernel* de Linux de la versión 2.6, que ofrece seguridad, administración de memoria, de procesos, un conjunto de redes y modelos de manejo. Este kernel trabaja sobre una capa de abstracción entre el HW y el resto del conjunto de SW (ver Figura 2.4 parte inferior). Android *Runtime* se basa en la máquina virtual de Java denominada *Dalvik VM* la cual está diseñada para requerir poca memoria y permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando en el sistema operativo el soporte de aislamiento de procesos, gestión de memoria

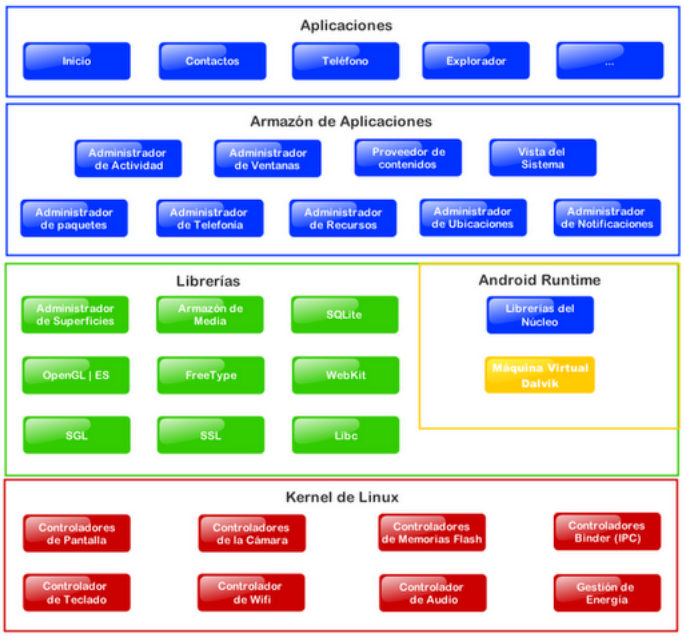



Figura 2.4: Arquitectura Android [30, 1]

.

e hilos (ver Figura 2.4 parte media). En este mismo nivel se encuentran las librerías del sistema (C/C++) que son utilizadas por varios componentes del sistema Android ver Figura 2.4parte media). Sobre esta capa se encuentra aplicaciones de estrutura (framework) programado en JAVA, que ofrece funcionalidad y acceso al gestor gráfico, gestor de contenidos, diferentes controladores de teléfono, localización y conectividad, listos para que los programadores puedan desarrollar sus aplicaciones de manera sencilla y puedan correr en un dispositivo móvil utilizando todos los niveles inferiores, (ver Figura 2.4 parte superior).

Dalvik VM es un intérprete que sólo ejecuta los archivos ejecutables con formato .dex (*Dalvik Executable*). Este formato está optimizado para el almacenamiento lo consigue delegando en el kernel la gestión de hilos (*multithreading*), memoria y procesos. La herramienta "dx" incluida en el SDK de Android permite transformar las clases

compiladas (.class) en formato .dex.

**Aplicaciones en Android y sus componentes**

Una característica fundamental de Android es que el tiempo y ciclo de vida de una aplicación no está controlado por la misma aplicación sino que lo determina el sistema a partir de una combinación de estados, que las aplicaciones están funcionando, que prioridades tienen para el usuario y cuánta memoria queda disponible en el sistema.

La plataforma de Android proporciona diferentes componentes para programar en función del objetivo de la aplicación provee cuatro tipos diferentes de componentes [1]:

**Activity:** Típicamente es una actividad llevada a cabo por una aplicación, se representa generalmente por una pantalla individual en la terminal como una interfaz gráfica de usuario. Este es el componente más usado en las aplicaciones en Android y permite su control mediante su ciclo de vida que son: activa, pausada, parada y reiniciada.

**Services:** Un componente servicio no tiene interfaz gráfica, pero puede ejecutarse en segundo plano por un tiempo indefinido mientras otras aplicaciones se encuentran activas en la pantalla del dispositivo.

**Broadcast receivers:** Este tipo de componentes no tienen interfaz gráfica y se emplean para recibir y reaccionar ante ciertas notificaciones broadcast.

**Intent:** Es una clase especial que usa Android para moverse de una pantalla a otra. Un Intent describe lo que una aplicación desea hacer. Cualquiera *activity* puede reutilizar funcionalidades de otros componentes con sólo hacer una solicitud en la forma

de *intent*.

**Content provider:** con este componente cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere, además la clase de este componente tiene métodos que permiten almacenar, recuperar, actualizar y compartir los datos de una aplicación.

### Estado de los procesos

Cada aplicación de Android se ejecuta dentro de su propio proceso, el cual se crea cuando se ejecuta y permanece hasta que la aplicación deja de trabajar o el sistema necesita memoria para otras aplicaciones. Android sitúa cada proceso en una jerarquía de importancia (ver Figura 2.5 ) basada en estados, como se puede ver a continuación [1]:

**Proceso en primer plano (*Active process*):** Es un proceso que aloja una Actividad en la pantalla y con la que el usuario está interactuando (su método *onResume()*). Este tipo de procesos serán eliminados como último recurso si el sistema necesitase memoria.

**Proceso visibles (*Visible process*):** Es un proceso que aloja una actividad pero no está en primer plano (su método *onPause()*). Esto ocurre en situaciones donde la aplicación muestra un cuadro de diálogo para interactuar con el usuario. Este tipo de procesos no será eliminado en caso que sea necesaria la memoria para mantener a todos los procesos del primer planoen ejecución.

**Proceso de servicio (*Started service process*):** Es un proceso que aloja un servicio que ha sido iniciado con el método *startService()*. Este tipo de procesos no son visibles y suelen ser importantes para el usuario (conexión con servidores, reproducción

de música).

**Proceso en segundo plano (*Bbackground process*):** Es un proceso que aloja una actividad que no es visible actualmente para el usuario (su método *onStop()* ha sido llamado). Normalmente la eliminación de estos procesos no suponen un gran impacto para la actividad del usuario.

**Proceso vacíos (Empty process):** Es un proceso que no aloja ningún componente. La razón de existir de este proceso es tener una caché disponible de la aplicación para su próxima activación. Es común que el sistema elimine este tipo de procesos con frecuencia para obtener memoria disponible.

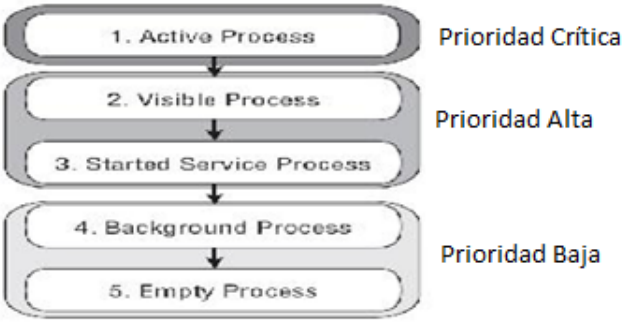



Figura 2.5: Jerarquía de importancia de los procesos.

De acuerdo con la jerarquía, Android prioriza los procesos existentes en el sistema y decide cuáles han de ser eliminados, con el fin de liberar recursos y poder lanzar la aplicación requerida.

Para los procesos en segundo plano, existe una lista llamada LRU (*Least Recently Used*), que implica que los primeros procesos que se eliminan son aquellos que llevan más tiempo sin usarse. Así, el sistema se asegura de mantener vivos los procesos que se

han usado recientemente.

#### Desarrollo de una actividad y el entorno de programación

Para programar una actividad (una pantalla) en Android, es necesario saber que la interfaz gráfica se define en un archivo XML y la lógica se programa en una clase JAVA. Android ofrece un conjunto de herramientas para el desarrollo de aplicaciones en dispositivos móviles, a continuación se enlistan [30].

1. Un SDK desde la versión 2.1 hasta la 4.0.

2. Un entorno de desarrollo de Eclipse que facilita el acceso al API.

3. Un emulador o pruebas directas sobre el dispositivo móvil.

## 2.2. Cómputo inteligente

### 2.2.1. Inteligencia artificial

El ser humano como ser inteligente, que razona y aprende por medio de la experiencia, ha tratado desde la segunda guerra mundial y hasta nuestros días de construir entidades inteligentes, dando lugar al campo de investigación de la Inteligencia Artificial (IA), que está conformada por áreas científicas y de ingeniería que le apoyan con su propósito [32]. Winston en [32] define a la IA como "El estudio de la computación que hace posible percibir, razonar y actua". Algunas disciplinas que la integran son: procesamiento de lenguaje natural, representación del conocimiento, razonamiento automático, aprendizaje automático,visión por computadora, robótica entre otras.

### 2.2.2. Aprendizaje automático

A partir de la invención de la computadora y de las ciencias computacionales, se ha intentado desarrollar y entender cómo aprenden los programas de computadora, además de que puedan mejorar su desempeño automáticamente con base a la experiencia adquirida, lo cual dio origen al área del Aprendizaje Automático (AA) [35].

Mitchell define al AA de la siguiente manera:"...un programa de computadora aprende, con base a la experiencia $E$ con respecto a alguna clase de tarea $T$ y la medida de desempeño $P$; si y sólo sí su funcionamiento en la tarea $T$, mejoran con la experiencia", la Tabla 2.4 contine un ejemplo de la definición dada por Mitchell.

Tabla 2.4: Ejemplo.

| |
|---|
| **Tarea ($T$):** Detectar rostros humanos.<br>**Medida de desempeño ($P$):** El porcentaje de rostros detectados correctamente .<br>**Experiencia de entrenamiento ($E$):** Un conjunto de imágenes de rostros. |

#### Tipos de aprendizaje automático

El aprendizaje automático se divide generalmente en dos grupos:

1) **Aprendizaje supervisado:** los algoritmos de este tipo de aprendizaje conocen la clase o el tipo del conjunto de datos y su aprendizaje se refiere en encontrar una hipótesis basándose en las clases para categorizar nuevos ejemplos.

2) **Aprendizaje no supervisado:** los algoritmos de este grupo no conocen la clase o tipo del conjunto de datos por lo tanto, su aprendizaje se basa en encontrar patrones en los datos y agruparlos de acuerdo a ciertas características localizadas.

**Aprendizaje supervisado.**

Formalmente, el aprendizaje supervisado se define de la siguiente manera: Dado un conjunto de entrenamiento de $M$ ejemplos de la forma:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \tag{2.1}$$

Donde $x_j \epsilon\ X$ que es el conjunto de entrenamiento y $y_j$ es la clase que es generada por una función desconocida $y = f(x)$, aplicada a $x_i$, (los valores de $x_i$ tiene la forma de vector $< x_{i1}, x_{i2}, \ldots, x_{in} >$), dado el conjunto de ejemplos, el propósito es descubrir una función $h$ que aproxime a una $f$ verdadera. A la función $h$ se denomina hipótesis, $x$ y $y$ pueden tener cualquier valor, no necesariamente números.

Entonces, el aprendizaje supervisado consiste en buscar una hipótesis que se comporte bien dentro del espacio de hipótesis $\mathcal{H}$, aún cuando nuevos ejemplos se encuentren fuera del conjunto de entrenamiento.

Por otra parte, cuando la salida (resultado) de $y$ es un conjunto finito de valores $\{1, \ldots, k\}$, se define al problema de aprendizaje como *clasificación*, también se le llama clasificación booleana o binaria en la cual solo existen dos valores (si/no). Si $y$ es un número, se dice que el problema de aprendizaje es de regresión.

**Algoritmos de clasificación**

Para poderse llevar a cabo el aprendizaje automático es necesario de algoritmos que realicen las tareas para aprender, ya sea de manera supervisada o no supervisada. En este apartado se hablará de aquellos que serán usados para este trabajo.

### Árboles de decisión

La estructura del árbol de decisión consiste de nodos de los cuales uno es la raíz (nodo inicial), en la parte final del árbol se tienen nodos hojas (la conclusión o clase) y entre la raíz y las hojas se forman caminos llamadas ramas que conectan a los nodos internos (forman subárboles).

Un árbol se construye partiendo del nodo raíz que es el atributo que mejor discrimina entre los ejemplos de entrenamiento, el cual se obtiene mediante una propiedad estadística llamada ganancia de información (ver ecuaciones entropía 2.2 y *ganacia de información* 2.3). Después se añadirán, al nodo raíz, otros nodos que corresponderán a los valores que puede tener este atributo. Este proceso se repite para cada uno de los nuevos nodos hasta haber analizado todos los atributos para clasificar objetos o instancias posteriormente. En la Figura 2.6 se muestra un árbol de decisión en el cual se establece una serie de condiciones meteorológicas para decidir si se puede realizar deporte a la intemperie.

$$Entropia(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{2.2}$$

donde $S$ es un ejemplo de entrenamiento, $p_{\oplus}$ es la proporción de ejemplos positivos en $S$, $p_{\ominus}$ es la propotción de ejemplos negativos en $S$ y Entropia es la medida de impuridad de $S$.

$$Ganancia(S, A) \equiv Entropia(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{2.3}$$

donde dada la Entropía como medida de impuridad en una colección de ejemplos de entrenamiento, se define una nueva medida de efectividad de un atributo en la clasificación de de los datos de entrenamiento llamada *GananciaGanada*.

Para realizar la clasificación de un objeto (instancia), el árbol toma los atributos del

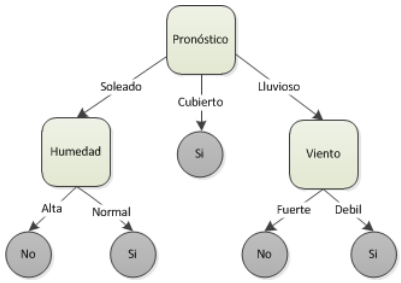



Figura 2.6: Árbol de decisión para pronosticar si es posible salir a realizar deporte. Un ejemplo de ello si tenemos (Pronóstico, soleado) $\wedge$ (Humedad, Normal) [35].

objeto $A_i$, que son evaluados en cada nodo, iniciando por la raíz hasta llegar a alguna hoja que provee su clasificación [32, 35].

Los árboles de decisión se construyen mediante diferentes algoritmos como IDA3, ASSITANT, C4.5 que pueden ser revisados en [35].

**Ensamble de clasificadores**

Dado un conjunto de entrenamiento del cual un algoritmo de aprendizaje produce un *clasificador* como salida, puede verse a esté como una hipótesis elegida del espacio de hipótesis, en donde nuevos ejemplos deben ser predichos [33]. Por otra parte, los clasificadores se denotan como $h_1, \ldots, h_L$.

Un ensamble de clasificadores es un conjunto de clasificadores donde sus predicciones individuales se combinadan de alguna manera (mediante pesos o sin éstos) para clasificar nuevos ejemplos [32, 33, 34, 9].

El propósito del ensamble es obtener mayor exactitud en lugar de un sólo clasificador que lo conforma (más detalles revisar [34]) y esta afirmación se muestra en Figura 2.7,

en donde tres árboles de decisión buscan aproximarse a la diagonal, en el inciso (a) lo realizan de manera individual, mientras (b) es una combinación (un ensamble) y por medio del voto simple obtienen un resultado mejor (mayor exactitud). Las condiciones

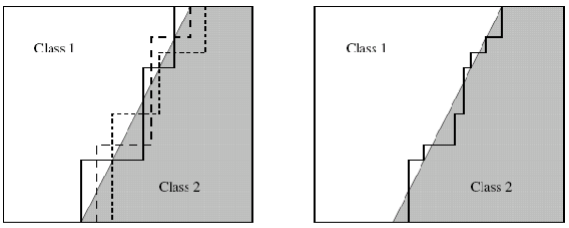



Figura 2.7: (a) es la representación de la clasificación individual de tres árboles de decisión y (b) es el resultado de un ensamble de estos tres clasificadores [34].

necesarias y suficientes para que un ensamble posea una mayor exactitud, consiste en que los clasificadores individuales tengan un buen desempeño y que sean diversos. Un buen desempeño significa tener errores menores al 50 % y diverso significa que cometan errores diferentes (deben ser independientes) [9].

### Boosting y AdaBoost

*Boosting* es la combinación de reglas simples para formar un ensamble de tal manera que el desempeño de un miembro (*clasificador*) del ensamble mejore [33]. Dado un conjunto de hipótesis o clasificadores $h_1, \ldots, h_T$ se consideran como el ensamble de hipótesis.

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{2.4}$$

donde $\alpha_t$ es el coeficiente (voto con cierta cantidad de peso) con el cual el ensamble miembro $h_t$ se combina, tanto $\alpha_t$ como $h_t$ deben ser aprendidos por el procedimiento de *Boosting* para que al finalizar el proceso se tome una decisión.

*Boosting* es un meta algoritmo de tipo ensamble que trabaja sobre pesos, de tal modo que es necesario un conjunto de entrenamiento donde cada ejemplo de entrenamiento está asociado a un peso $w_j \leq 0$. El peso es importante para el proceso de aprendizaje de una hipótesis.Para construir un ensamble *Boosting*, éste manipula los ejemplos de entrenamiento (pesando los datos) con el fin de generar múltiples hipótesis y combinarlas [32].

Existen diferentes variantes de *Boosting* principalmente por la manera de ajustar los pesos y de combinar las hipótesis, una de ellas es:

*AdaBoost*(en inglés *Adapting Boosting*) que utiliza el mismo ejemplo de entrenamiento

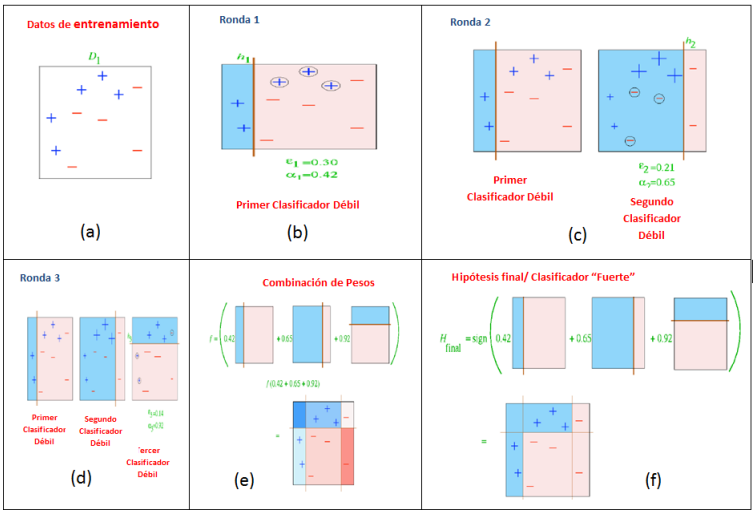



Figura 2.8: Representación del funcionamiento del algoritmo AdaBoost [38]

una y otra vez; también combina un número arbitrario de clasificadores y los genera de manera secuencial, AdaBoost es una adaptación de *Boosting* y es conocida por ser un clasificador binario {0, 1}.

El funcionamiento de este algoritmo es el siguiente:en un principio inicializa todos

los pesos del conjunto de entrenamiento (1/m) (ver Figura 2.8 (a)); de este conjunto se genera una $h_1$ (primer clasificador débil), el cual clasifica a algunos ejemplos de entrenamiento de manera correcta e incorrecta (ver Figura 2.8(b)). En una siguiente hipótesis el desempeño debe mejorar sobre los ejemplos mal clasificados, por lo tanto incrementa, su peso y decrementa el de aquellos que fueron clasificados correctamente, obteniendo $h_2$ (ver Figura 2.8 (c)). Este proceso es iterativo (nuevamente se actualizan los pesos para los nuevos clasificadores, el objetivo es minimizar el error esperado) y continua hasta haber generado $T$ hipótesis ($T$ es un número de entradas al algoritmo) (ver Figura 2.8(d)). Finalmente las hipótesis o clasificadores se ensamblan, lo que significa que se combinan los pesos de todas las hipótesis de $T$ (ver Figura 2.8 (e)), finalmente se obtiene un último clasificador (fuerte) mediante la mayoría de votos (pesos) de las predicciones, que depende del desempeño de cada clasificador en su conjunto de entrenamiento (ver Figura 2.8 (f))[12, 38].

### 2.2.3. Visión por computadora

"Visión es el proceso de descubrir de las imágenes lo que está presente en el mundo y en donde se encuentra" de acuerdo con David Marr [21]. La Visión por Computadora (VC) es una rama de la inteligencia artificial que tiene como meta generar programas de computadora que puedan interpretar imágenes y video de manera similar a la percepción del ser humano, es decir, desarrollar modelos matemáticos y algoritmos para crear un representación del mundo real a partir de las imágenes de acuerdo a la percepción visual humana por medio de la computación [12, 21].

Un humano percibe su mundo a través de diversos sentidos, entre ellos la vista.

Para un humano resulta hasta cierto punto muy sencillo reconocer a los objetos en su entorno gracias a las diferentes capacidades desarrolladas por el cerebro, sin embargo, una computadora no tiene esta capacidad de interpretación, ésta necesita de hardware especializado (como una cámara (sensor)) el cual capta radiación (luz) que ha interactuado con ciertos objetos físicos y de esta forma proporcionarinformación sobre su entorno. Estos valores no tiene ninguna interpretación ni significado para la computadora, entonces son necesarias las técnicas, métodos y los algoritmos para darse a la tarea de interpretar las imágenes del mundo real (La figura 2.9 se representa un ejemplo de la información obtenida por una cámara). Algunas de las tareas que estudia la

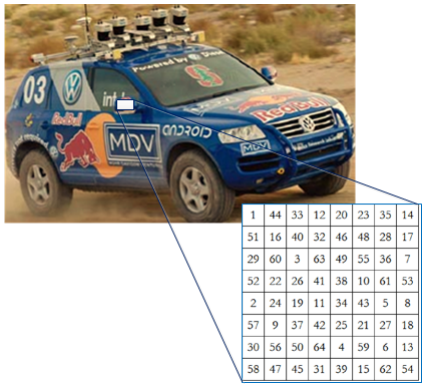



Figura 2.9: Representación de los datos (matriz de números) captados por una cámara los cuales son trasmitidos y almacenados en una computadora para su futuro análisis [12].

VC son la detección de movimiento y el reconocimiento de objetos que a continuación se describen.

### Detección de movimiento

El análisis de la detección de movimiento puede obtenerse de maneras diversas en este apartado se habla en especial del modelo de movimiento de plantilla (MP) (*motion template*), que fue inventado en el *MIT Media Lab* por Bobick, Davis y Bradski

*OpenCV* [12, 18].

MP usa imágenes (secuencia de imágenes o acumulación de siluetas de objetos) en las cuales permite determinar rápidamente en dónde ha ocurrido un movimiento, como se produjo y en qué dirección ocurrió [12, 18]. Para determinar el movimiento, MP requiere:

1. Silueta (s) (partes de la silueta) de un objeto.
2. Historial de movimientos de imágenes.
3. Cálculo de gradientes (orientación global).

**1. Siluetas**

Para obtener la silueta de una persona o de un objeto puede obtenerse por métodos diversos como lo son: la diferencia de imágenes (*frames*), el uso de un color de fondo estático para extraer el objeto que no coincide con este, el uso de una luz infrarroja (cámara especial) mediante imágenes térmicas y técnicas de segmentación [12] (ver Figura 2.10).

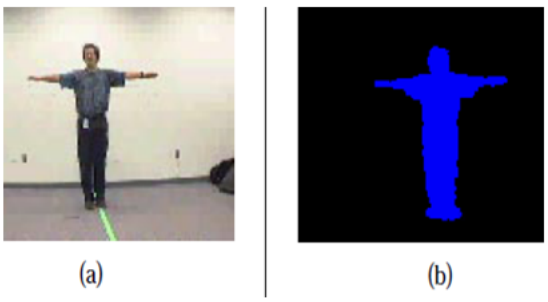



Figura 2.10: Método de substracción de fondo usado por Davis y Bradski, (a) la imagen de entrada, (b) Extracción de la silueta del video [18].

**2. Historial de movimiento de imágenes**

La representación compacta y en tiempo real de la captura de una secuencia de movimientos (siluetas sucesivas) en una imagen estática se llamada historial de movimientos de imágenes (HMI). El HMI se construyeo por capas sucesivas seleccionadas de las regiones de la imagen sobre el tiempo (se captura el movimiento reciente de los pixeles en función de un historial temporal de posición en el punto) usando una regla de actualización (un operador de duración y remplazo en base al tiempo de estampado)[17]:

$$HMI_{\delta}(x,y) = \begin{cases} \tau & \text{si } \Psi(I(x,y)) \neq 0 \\ 0 & \text{de lo contrario si } HMI_{\delta}(x,y) < \tau - \delta \end{cases}$$

donde cada pixel $(x, y)$ en el HMI es marcado con un tiempo de estampado actual $\tau$, si la función $\Psi$ señala la presencia de un objeto (o movimiento) en la imagen actual del video $I(x, y)$; el resto del tiempo de estampado en el HMI es removido si los anteriores son previos al valor de $\tau - \delta$, ($\delta$ tiempo máximo de duración asociado a la plantilla), esta función de actualización es llamada cada vez que una nueva imagen (frame) se analice en la secuencia.

La función $\Psi$ (función de selección del objeto, método de substracción de fondo y diferencia de imágenes) selecciona la localización del pixel en la imagen de entrada para incluirse dentro del HMI y pueda ser especificado arbitrariamente (la plantilla contendrá el historial de la posición y el historial del tiempo).

Para representar el propósito del tiempo de estampado del HMI, los valores de los pixeles en la platilla son mapeados linealmente por intensidades del color gris del valor 0-255. Los pixeles con mayor brillo (blancos) corresponden a los más recientes (tiempo actual) y aquellos con una escala de gris mayor intensidad son los pixeles previos o más antiguos (ver Figura 2.11). (Computacionalmente, el HMI se define como un vector de características en relación al tiempo global actual y en base a las intensidades de las

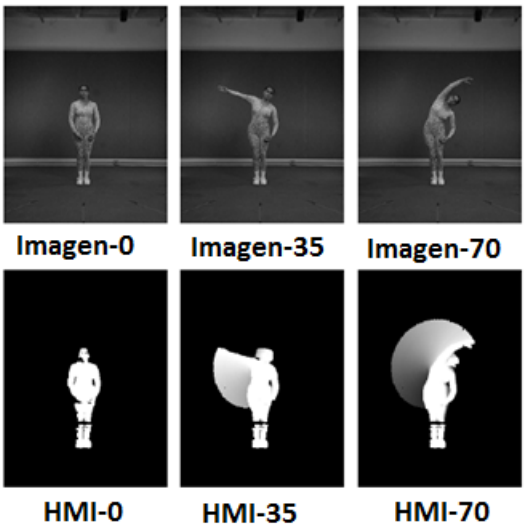



Figura 2.11: Representación del HMI y las intensidad del color gris en relación a la actualización del movimiento [17].

imágenes que se etiquetan en relación al tiempo).

**3. Gradientes**

Las capas de diferencia del HMI permiten determinar la forma del movimiento (dirección) del contorno de las siluetas y puede ser percibida de la intensidad de los gradientes de los valores de los pixeles del HMI (de los pixeles obscuros a los claros ver Figura 2.12 (a)). Davis y Bridski implementaron un método simple de deriva para extraer el componente de movimiento y determinar la dirección [18].

La intensidad local de los gradientes de HMI, producen directamente la orientación del contorno del movimiento de la silueta. Por lo tanto la orientación del movimiento local puede ser extraido por la convolución clásica de la máscara de intensidades de gradientes de la imagen a través del HMI. Para calcular la convolución es implementado la máscara de gradiente Sobel definida enseguida:

$$F_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad F_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Los gradientes de las imágenes $F_x(x, y)$ y $F_y(x, y)$ calculado del HMI es una manera simple de obtener el gradiente (movimiento) de orientación para un pixel por medio de la aproximación de un arctan que calculará la orientación.

$$\phi(x, y) = \arctan \frac{F_y(x,y)}{F_x(x,y}$$

Sólo el HMI de las siluetas interiores se utilizan para ser examinados dejando fuera aquellos gradientes que se encuentren en el contorno de las capas de las siluetas, aquellas gradientes de los pixeles de HMI que tengan un contraste demasiado bajo o demasiado alto no son utilizados (ver Figura 2.12 (b)). A partir de lo anterior, se pueden extraer las características de movimiento de la varianza de escalas y se genera un histograma radial de orientaciones de movimiento (ver Figura 2.12 (c) líneas verdes) dando como resultado una orientación espacial.

Obtenida orientación espacial el propósito es obtener una medida única u orientación

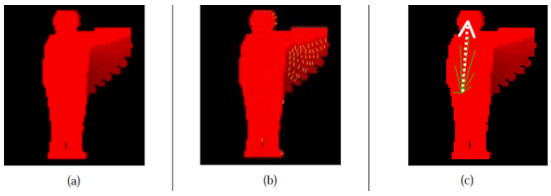



Figura 2.12: Gradientes. (a)HMI del movimietno del brazo izquierdo, (b)Orientación del movimiento local usando la máscara de convolución Sobel, (c) Historial radial de la orientación del movimiento y la orientacion global $\bar{\phi} = 90°$ [18].

global del movimiento. Una manera es a partir de la normalización de los valores del

HMI tomando en cuenta la mayor influencia del movimiento más repetido dentro de la plantilla y se calcula de la siguiente manera.

$$\bar{\phi} = \phi_{ref} + \frac{\sum_{x,y} angDiff(\phi(x,y), \phi_{ref}) \times norm(\tau, \delta, HMI_{\delta}(x,y)))}{norm(\tau, \delta, HMI_{\delta}(x,y))}$$

donde $\bar{\phi}$ es la orientación del movimiento global, $\phi_{ref}$ es la base del ángulo de referencia (máximo valor en el histograma de orientaciones), $\phi(x,y)$ es la orientación de movimiento en el mapa encontrado de la covolución del gradiente, $norm(\tau, \delta, HMI_{\delta}(x,y))$ es el valor normalizado de HMI (normalización lineal de HMI de 0-1 usando el tiempo de estampado actual $\tau$ y la duración $\delta$) y $angDiff(\phi(x,y), \phi_{ref})$ es el mínimo, es el signo de la diferencia angular de una orientación del ángulo de referencia. Un histograma base del ángulo de referencia ($\phi_{ref}$) es requerido debido al problema asociado con una medida de una distancia circular.

**Reconocimiento de objetos usando clasificadores en cascada de Haar**

Generalmente los trabajos para hacer reconocimiento de objetos basan su análisis sobre los pixeles de la imagen, pero resulta costoso en tiempo y en procesamiento. Cuando se hace la detección de un objeto se procesan los pixeles y se hallan variaciones en las características de éstos (color, reflexión de la luz, movimientos y otros elementos), aunque se trate del mismo objeto. Viola y Jones [40] propusieron un método para el reconocimiento de objetos en tiempo real, utilizando un clasificador de cascada *AdaBoost* que se basa en las características de Haar y no sobre el tratamiento de pixeles.

El *clasificador de Haar* de detección de objetos, no hace uso de los valores de intensidad de pixeles, se basa principalmente en *características*, que consisten en utilizar el cambio de contraste entre grupos de pixeles que forma rectángulos adyacentes en una

imagen (da como resultado una manera simple y barata de cómputo) estas son similares a las características de *Haar Wavelets*.

Las varianzas de contraste entre los grupos de pixeles se usa para determinar áreas

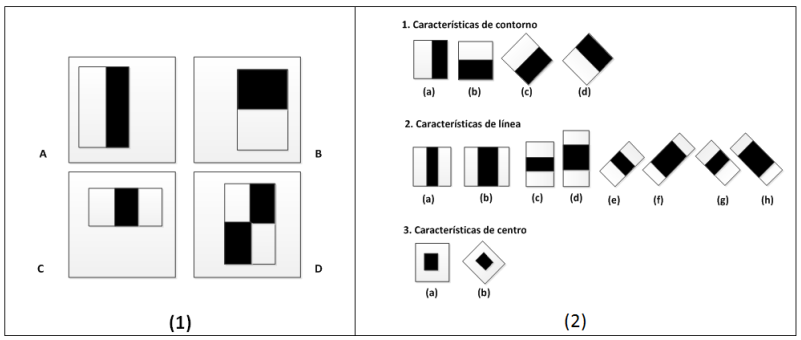
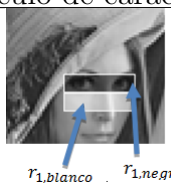



Figura 2.13: (1) Contiene cuatro características, A y D son características de dos rectángulos, C tres rectángulos y finalmente D con cuatro rectángulos. (2) contiene tres clasificaciones de las anteriores características de manera extendida .

relativas de luz u obscuridad. Dos o tres grupos adyacentes con un contraste relativo (rectángulos negros y blancos) forman una característica de Haar también conocidas en inglés como *Haar-like features* [40, 39]. En la Figura 2.13 (1) se muestra un conjunto básico de las características de Haar [40], mientras tanto en la Figura 2.13 (2) representa un conjunto extendido de este grupo [39].

Matemáticamente, el valor de las características de Haar se obtiene como la diferencia de la suma de las regiones (rectángulos blancos y negros) como se muestra en la Tabla 2.5: Las características de Haar pueden ser escaladas fácilmente es decir, incrementar o

Tabla 2.5: Cálculo de característica de Haar




$$F_{Haar} = \sum(r_{i,blanco}) - \sum(r_{i,negro})$$

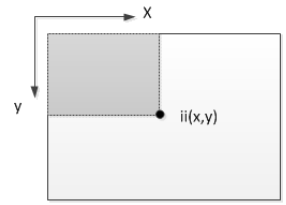
Donde $r_i$ establece el número de rectángulos negros o blancos.

decrementar el tamaño de los grupos de pixeles a ser examinados, esto permite usarlas para la detección de objetos que se encuentren en varios tamaños, como resultado una clasificación mucho más rápida [28].

Una manera rápida de calcular o procesar las características en una imagen de dos dimensiones es mediante la representación intermedia de una imagen llamada *imagen integral* [13]. La imagen integral se denota como *ii(x,y)* en la posición *(x,y)*, *ii(x,y)* contiene la suma de los valores de intensidad de los pixeles localizados a la izquierda y directamente sobre la posición de *(x,y)* (ver Tabla 2.6).

Tabla 2.6: Representación de la imagen integral.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$




La imagen integral puede ser calculada en un paso sobre la imagen original [6] (ver Figura 2.14) :

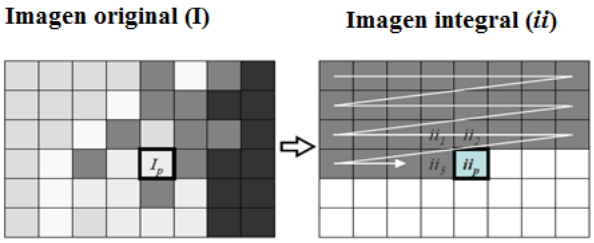
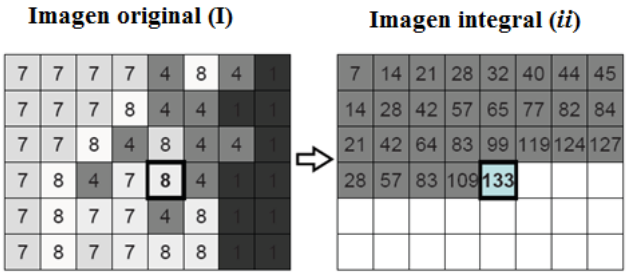



Figura 2.14: Calculo de la imagen integral.

$$ii_p = ii_2 + ii_3 - ii_1 + I_p$$

Ejemplificando el calculo de la imagen integral $ii_p$(ver Figura 2.15):




$$133 = 99 + 109 - 83 + 8$$

Figura 2.15: Encontrar los valores de la imagen integral

Dada la imagen integral la, suma de los valores de los pixeles dentro de una región rectangular de la imagen alineada con los ejes de coordenadas puede ser calculada con cuatro arreglos de referencia. Un ejemplo de ello se representa en la Tabla 2.7 donde se calcula la región D siguiendo las referencias requeridas: $D = ii_4 + ii_1 - (ii_2 + ii_3)$.

Por otra parte, Viola y Jones proponen un conjunto de características (Figura 2.13 inciso (1) y (2) segunda clasificación característica (b)) para la detección de objetos haciendo uso de la imagen integral antes vista, cada característica puede ser calculada en un tiempo constante. Para aquellas características orientadas a 45° el cálculo de la imagen integral sufre una variación, para ello consultar la referencia [13].

### Cascada de clasificadores

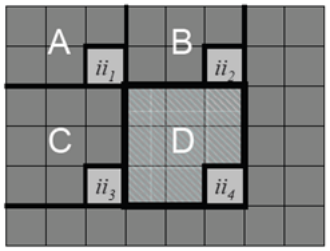
Para realizar la tarea de la detección de objetos, los árboles de decisión son necesarios para construir la "cascada de clasificadores"(cascada de reacción de nodos *boost*), su propósito es incrementar el desempeño en la detección y reducir radicalmente el tiempo de cómputo [12, 40]. La clave consiste en la implementación de grupos de clasificadores

Tabla 2.7: Calculo de la region D dentro da la imagen original a partir de la imagen integral [6].

$$ii_1 = \sum(A)$$
$$ii_2 = \sum(A) + \sum(B)$$
$$ii_3 = \sum(A) + \sum(C)$$
$$ii_4 = \sum(A) + \sum(B) + \sum(C) + \sum(D)$$

$$\sum(D) = ii_4 + ii_1 - (ii_2 + ii_3)$$

Imagen Integral ($ii$)




*Adaboost* en cada uno de los nodos de la cascada. Estos clasificadores debe rechazar una gran cantidad de subventanas, mientras detectan al menos todas las instancias positivas (al igual que los árboles, la cascada permite entrenar a los clasificadores usando los ejemplos de entrenamiento e ir ajustando el umbral para reducir los falsos negativos).

El objetivo de los clasificadores *AdaBoost´s* en cada nodo de la cascada consiste en aprender y tener una alta tasa de detección en el costo de una baja tasa de ejecución [12]. Cuando la clasificación dentro de cada nodo es negativa, finaliza el cómputo y el algoritmo declara la inexistencia del objeto a buscar, de lo contrario, si el resultado es positivo, la detección del objeto es declarada, si y solo sí se han procesado todos los nodos de la cascada. Una cascada de clasificadores se muestra en la Figura 2.16.

El funcionamiento de la cascada es el siguiente:

1. El primer clasificador débil (nodo raíz) elimina un gran número de subventanas negativas y deja pasar al menos todas las subventanas positivas (éstas contiene

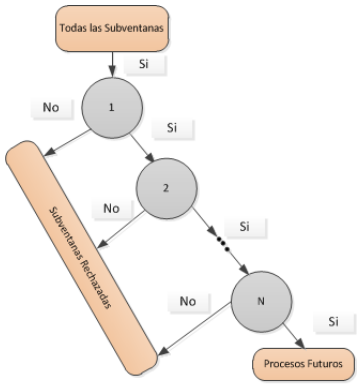



Figura 2.16: Representación de una cascada de clasificadores.

aun una tasa alta de falsos positivos y detecta todas las instancias positivas) al siguiente nodo.

2. El siguiente nivel (nodo), elimina las subventanas negativas adicionales (que fueron pasadas por el anterior nodo). Para ello requiere procesamiento adicional, el resultado que se obtiene es el aumento de la tasa de detección, lo que permite ejecutar el siguiente nivel.

3. Después de varios niveles (nodos o clasificadores complejos) el número de subventanas negativas es reducido radicalmente.

4. Finalmente, el último nodo (llamado proceso futuro), posee el sistema de detección del objeto el cual concluye la existencia del objeto buscado.

### 2.2.4. OpenCV

OpenCV (*Open source Computer Vision library*) es un conjunto de librerías de código abierto escritas en C y C++, que permiten el desarrollo de aplicaciones de visión por computadora en tiempo real, creadas por los laboratorios de Intel [12]. Estas librerías proporcionan una gran cantidad de funciones para la captura (video de webcams), el

análisis y manipulación de datos visuales (imágenes y video).

Las librerías proveen funciones simples y complejas que permite a los programadores crear aplicaciones poderosas en el dominio de la visión por computadora. Además, OpenCV ofrece varios tipos de datos de alto nivel como estructuras dinámicas, árboles, gráfos, matrices, etc. También provee diversos algoritmos para la detección de rostros, rastrear movimiento y análisis de características. Otros algoritmos se relacionan con la aplicación en la industria (supervisar la producción), la seguridad (video-vigilancia), procesamiento de imágenes médicas, calibración de cámaras, interfaces de usuarios, aplicaciones militares, robótica, entre otras.

La visión por computadora y el aprendizaje automático se relacionan entre sí en diversas tareas, en OpenCV se integra librerías para el aprendizaje automático, las cuales están enfocadas al reconocimiento de patrones estáticos y su agrupación.

El algoritmo clasificador de Haar propuesto por Viola y Jones para la detección de rostros (conocido como detector Viola-jones), se encuentra implementado en OpenCV. Este clasificador necesita de las librerías de aprendizaje automático y de visión por computadora, con el propósito de construir un clasificador que tenga buen desempeño en tiempo real (por medio de *boosting*), que permita realizar el reconocimiento de tareas (objetos diferentes) como el propósito de este trabajo, en el que se implementará la detección de una persona en un sistema de video-vigilancia.

OpenCV se encuentra estructurado en cuatro componentes y se muestra en la Figura 2.17 , El componente CV contiene el procesamiento básico para las imágenes y algoritmos de alto nivel de visión por computadora. El componente MLL contiene las librerías

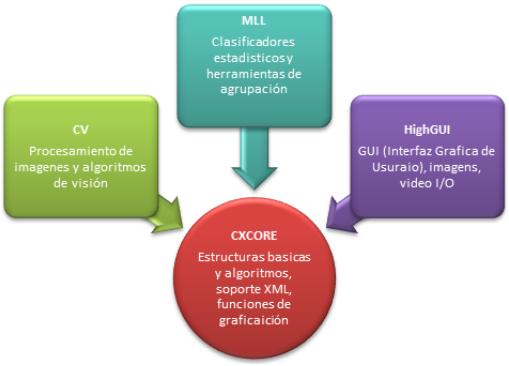



Figura 2.17: Componentes que forman OpenCV [12].

de aprendizaje automático, HighGUI contiene funciones y rutinas de entrada y salida para la lectura y el almacenamiento de video e imágenes. Por último el componente CXCore posee las estructuras básicas y el contenido.

[1]

[1] OnpenCV disponible en http://sourceforge.net/projects/opencvlibrary/

# Capítulo 3

# Estado del arte

En este capítulo se describen diversos trabajos que proponen sistemas de video-vigilancia de tercera generación, los cuales se clasificaron de acuerdo a los siguientes aspectos : monitoreo en tiempo real, detección de movimiento y reconocimiento de objetos.

## 3.1. Monitoreo en tiempo real

En [4] se propone un modelo de capas para un sistema de video-vigilancia móvil en tiempo real basado en la red inalámbrica LAN (WAN), en donde el usuario final puede monitorear una región y controlar las cámaras desde un asistente personal digital PDA. Este trabajo propone dos capas para el desarrollo de su esquema, enseguida se define cada una de ellas:

1. *Capa de arquitectura general*: Esta capa está constituida por los módulos de cámaras, servidor principal de control, terminales de monitoreo y componentes WLAN.

2. *Capa de video-vigilancia móvil*: Esta capa incluye módulos de captura de video móvil, transmisión y codificación de video, protocolos de transporte de red, mul-

ticast IP, transmisión y recepción de señales de control, funciones de la terminal móvil PDA.

El funcionamiento del sistema [4] se lleva a cabo de la siguiente manera: el video captado por una serie de cámaras se transmite a un servidor central, el video se visualiza en monitores, se codifica, se almacena y se transmiten flujos de video a las terminales móviles (el video se codificado en MPG-4 ). Cuando el flujo de video llega al móvil, se descodifica y reproduce. Finalmente, el usuario tiene la capacidad de monitorear la zona y controlar las cámaras.

Por otra parte, un sistema de video-vigilancia de monitoreo, que visualiza una diversificación de hardware y software se propone en [16], que está basado en el sistema de procesamiento ARM9, compuesto por terminales móviles, terminales de captura móvil, un centro de monitoreo y el uso de la red inalámbrica 3G.

## 3.2. Detección de movimiento

En la investigación de [14], propone un sistema de video-vigilancia que permite visualizar y controlar 16 cámaras mediante el uso de dispositivos móvil (un celular), con el fin de activar luces remotamente, sirenas u otros dispositivos. El sistema está constituido de alarmas con cámaras instaladas para realizar la detección de movimiento de intrusos. Al detectar un hecho anormal, el sistema envía un SMS de aviso al celular del usuario final, permitiéndole visualizar la zona en tiempo real desde su dispositivo móvil y hacer acciones correspondientes como capturar y enviar imágenes vía correo electrónico.

El desarrollo de un sistema de video-vigilancia a bajo costo en hardware, integrando teléfonos celulares con cámaras, en donde se procesa un algoritmo de substracción de *background* (una técnica de segmentación para detección de movimiento) se propone por [31, 22]. El funcionamiento del algoritmo utilizado consiste en comparar la imagen observada con una imagen estimada (previa) indicando el movimiento de un objeto. Al activarse la detección de movimiento, el centro de control o el usuario final se notifica mediante una llamada o un SMS. Es importante mencionar que ambos trabajos incluye un modelo de tres capas: 1) la adquisición de video, 2) la detección de movimiento y 3) el envío de alarmas al usuario final.

De la misma manera, una arquitectura de red inteligente llamada *FACET* es definida por [26], la base de esta propuesta es el uso de teléfonos celulares con cámaras para la video-vigilancia reduciendo los costos, además esta red inteligente de celulares tiene la capacidad de detectar movimiento (eventos) de las zonas monitoreadas. Este sistema trabaja bajo eventos distribuidos, es decir, cómo se fueron generando los eventos, se definen dos tipos de eventos uno de entrada y uno de salida (el ingreso y la partida de un intruso), este procedimiento es procesado en cada uno de los teléfonos celulares.

La arquitectura *FACET* en [26], se encuentra estructurada por los módulos: cámaras, captura y análisis de imágenes, tiempo de sincronización, sistema de eventos distribuidos, red de comunicación y calibración. El módulo de calibración consiste en tener las condiciones internas y externas de una cámara para la video-vigilancia como la posición, la orientación y un algoritmo de calibración. En cuanto al módulo de análisis del video, se utiliza el método de substracción de *background* que presenta desventajas en aspectos de iluminación, color de ropa, velocidad del objeto en movimiento; estos retos se tratan en los trabajos de video-vigilancia *Knight* y *Whitenight* de [11] y en la investigación

de detección de movimiento y rastreo de humanos que propone [10], en donde utiliza un candidato *foreground* y filtros morfológicos para reducir el ruido para la detección y clasificación de una persona o grupos de personas (siluetas humanas) y finalmente, analizar actividades como caminar y correr.

## 3.3. Reconocimiento de objetos

La propuesta [29], posee el objetivo de identificar y clasificar vehículos y peatones en movimiento. Este trabajo tiene un proceso de tres fases: la primera fase es la extracción del objeto en movimiento, en la cual propone la aplicación de una substracción de background que segmenta el movimiento de cada pixel del fondo, usando una mezcla de la distribución Gaussiana K. La segunda fase, consiste en la identificación del objeto, en la que incorpora la herramienta SUSAN que realiza la detección de vehículos en las esquinas de las calles. Finalmente, la tercera fase es rastreado el objeto, que implementa filtros para reducir ruidos, un método híbrido, una función de seguimiento y el factor de Kalman. Esta última fase forma rectángulos (alarmas) que encierra al objeto rastreado para la identificación visual.

Del mismo modo, una arquitectura de análisis de video distribuida llamada *DiVA* propuesta en [19], la cual desarrolla un modelo físico y lógico para el análisis de video en tiempo real, que se describen enseguida.
El modelo físico está constituido por una red física que se divide en dos subredes, la primera referente al control de monitoreo y la segunda encargada de la ejecución del algoritmo para el análisis del video y su almacenamiento. Para el monitoreo utiliza tres modelos de cámaras (IEE1394, IP y GigE), siguiendo un modelo cliente-servidor.

El modelo lógico, plantea cuatro capas: adquisición de video, comunicación, procesamiento y gestión de datos. La adquisición de video se refiere al video captado por los tres tipos de cámaras, el cual será transmitido punto a punto mediante el protocolo TCP. La capa de procesamiento consiste en analizar el contenido del video; el objetivo detectar objetos olvidados. En cada escenario (3 cámaras) se extrae el *foreground* de los objetos y se clasifican las regiones estáticas del *foreground* como un objeto olvidado. Esta capa se divide en dos módulos uno en el que se segmenta el *foreground* y otro que extrae el objeto candidato. Al lograse la identificación de un objeto sobre el fotograma (*frame*) se generan las alarmas sobre el escenario correspondiente. Este sistema no incorpora el uso de dispositivos móviles para visualizar lo que se está monitoreando de manera remota. Por último, la capa de gestión de datos se encarga de almacenar y distribuir la información requerida para un futuro análisis, a su vez esta capa se divide en dos módulos uno de análisis de resultados (uso del formato MPG-7 para almacenar video) y otro de información contextual.

Por otro lado, [25] presenta un método de diseño para implementar sistemas concurrentes de tiempo real de gran escala llamada MASCOT, este incluye el diseño de un sistemas de video-vigilancia distribuido e inteligente formado por dos módulos: el primero denominado Módulo de Procesamiento de Datos (MPD) y la segundo llamado Módulo de Control (MC). El primer módulo MPD se encarga de las funciones de adquisición de video, procesamiento de imágenes (segmentación, detección de movimiento, reconocimiento, rastreo) y el almacenamiento de evidencias (base de datos). El módulo MC tiene dos tareas importantes: controlar las funciones de MPD y mostrar lo monitoreado al usuario final.

Finalmente en la Tabla 3.1, presenta las funcionalidades principales de los trabajos

que fueron tratados con anterioridad de acuerdo a los aspectos de monitoreo en tiempo real (color azul), detección de movimiento (color naranja) y reconocimiento de objetos (color rojo).

Tabla 3.1: Funcionalidades que presentan las propuestas diversas.

| Ref. | Detección de movimiento implementado en: | | | Reconocimiento de objetos implementado en: | | | Notificación a usuario final (alarmas) | Monitoreo en tiempo real desde un dispositivo móvil | Software de desarrollo |
|---|---|---|---|---|---|---|---|---|---|
| | Cámara (hardware especial ) | Servidor | Dispositivo móvil | Cámara (hardware especial ) | Servidor | Dispositivo móvil | | | |
| [4] | No | No | No | No | No | No | No | Si | Visual C++ y Video for Windows (VFW). Windows CE SO. |
| [16] | No | No | No | No | No | No | No | Si | Linux |
| [14] | Si | No | No | No | No | No | Si | Si | J2ME Java Microedition |
| [22] | No | No | Si | No | No | No | Si | No | J2ME, WMA, MIDLest CBS |
| [26] | No | No | Si | No | No | No | No | No | Java APIs |
| [30] | No | Si | No | No | No | No | Si | Si | Android |
| [31] | No | No | Si | No | No | No | Si | No | J2ME Java Microedition, Media Movil API y JRS120 |
| [19] | No | Si | No | No | Si | No | No | No | Windows SO |
| [29] | No | Si | No | No | Si | No | No | No | Linux |

## 3.4. Discusión

En la sección anterior se han descrito los trabajos más relevantes en cuanto al monitoreo, detección de movimiento e identificación de objetos, en algunos casos estas tareas se resuelven con la ayuda e implementación de dispositivos (HW especial) o algoritmos ejecutándose en servidores o dispositivos móviles. A partir de estas propuestas, el trabajo actual propone cubrir los aspectos de monitoreo en tiempo real a través del uso de un dispositivo móvil (*Smart phone*), la detección de movimiento mediante la implementación del algoritmo de movimiento de plantilla (*motion termplate*) que accionara a un segundo algoritmo que realiza la detección de una persona (algoritmo clasificador de cascadas de Haar). Al detectarse una persona, el sistema enviara alarmas a un usuario final (alarmas de mensajes de texto y multimedia dando aviso del hecho anormal).

Ambos algoritmos se encuentran implementados en un servidor que controla dos cámaras (webcam e IP) y en un dispositivo móvil (Smartphone con cámara integrada), todas estas cámaras se encuentran adquiriendo video dentro de una área controlada permitiendo de esta manera cubrir los tres aspectos ya mencionados, además de generar alarmas que informan al usuario final de un intruso. El lenguaje de programación de desarrollo es Java implementado bajo Andorid.

# Capítulo 4

# Metodología

La metodología a seguir en este proyecto de tesis consiste en utilizar el Proceso de desarrollo Unificado de Software PUDS, este proceso se basa en componentes de software interconectados a través de interfaces bien definidas.

El PUDS está dirigido por *casos de uso*, centrado en la *arquitectura*, es *iterativo* e *incremental* y se describen a continuación:

1. **Casos de uso**: Es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante .

2. **Arquitectura**: Describe una vista del diseño con las características más importantes del sistema en construcción

3. **Iterativo e incremental**: Dividide al sistema en miniproyectos o prototipos, cada uno de estos es una iteración que resulta un incremento.

El PUDS consta de un ciclo de vida que se divide en cuatro fases: 1)el *inicio* (Definir el alcance del proyecto), 2)la *elaboración* (planificar el proyecto, elaborar una arquitectura base), 3) la *construcción* (construir el sistema) y 4) la *transición* (el paso hacia los usuarios). Cada una de estas fases se subdivide en iteraciones (prototipos), en cada

iteración se desarrolla una secuencia o flujos de trabajo como el *modelo de análisis*, el *modelo de diseño*, el *modelo de implementación* y el *modelo de pruebas* como se muestra en la Figura 4.1.

Los siguientes puntos presentan cada uno de los elementos del flujo de trabajo y los diagramas que conllevan:

- **Modelo de análisis** (Diagrama de casos de uso y de clases preliminares).
- **Modelo de diseño** (Diagrama de clases detallado, de secuencia e iteración, estados, de actividades).
- **Modelo de implementación** (Modelo de implementación, crear el plan de integración, implementar componentes, validar componentes implementados, integrar subsistemas, validar subsistemas implementados e integrar el sistema)
- **Modelo de pruebas** (verificar la iteración e integración del componente, diseño de pruebas)

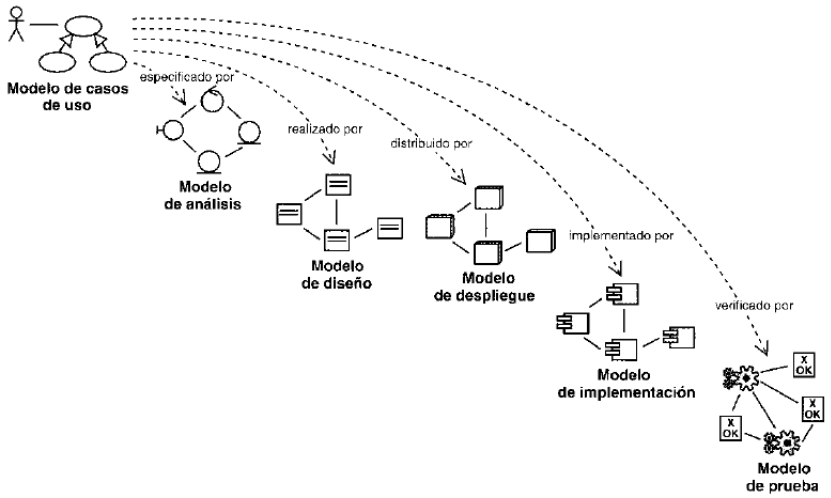



Figura 4.1: Flujo de trabajo del modelo Proceso de desarrollo Unificado de Software PUDS.

En la Tabla 4.1 se definen ocho prototipos que se llevaran a cabo en el desarrollo del Sistema Distribuido de Video-vigilancia para Dispositivos Móviles.

Tabla 4.1: Miniproyectos para el Sistema Distribuido de Video-vigilancia para Dispositivos Móviles (incluidos en el cronograma de activiadades)

| |
|---|
| **Etapa uno PUDS (Servidor-Cámaras) Monitoreo.**<br>Mecanismo de captura de imágenes (flujo de video)en tiempo real y envió de este flujo al usuario final (Servidor de flujo de video). |
| **Etapa dos PUDS (Servidor-Cámaras) Gestionar sistema.**<br>Desarrollo de una aplicación Web para usuario móvil final donde este es capaz de realizar altas, bajas, actualizaciones y eliminar la configuración de vigilancia (activación de los algoritmos de VC, tiempos), usuarios y evidencias (fotos, videos). |
| **Etapa tres PUDS (Servidor-Cámaras).**<br>Algoritmo de detección de movimiento (motion template) y detección de personas (clasificador de cascadas de Haar) utilizando JavaCV además de permitir almacenar evidencias (imagen y secuencia de video). |
| **Etapa cuatro PUDS (Servidor-Cámaras)**<br>Generar Alarmas de tipo mensaje de texto y mensaje multimedia. |
| **Etapa cinco PUDS (dispositivo móvil).**<br>Algoritmo de detección de movimiento (motion template) y detección de personas (clasificador de cascadas de Haar) utilizando Android además de permitir almacenar evidencias (imagen y secuencia de video) en el servidor. |
| **Etapa seis PUDS (dispositivo móvil).**<br>Transmisión de video (entre dispositivos móviles o usando el servidor de Streaming) para el monitoreo. |
| **Etapa siete PUDS (dispositivo móvil).** Generar alarmas de tipo mensaje de texto y mensaje multimedia, emitidos desde un dispositivo móvil. |
| **Etapa ocho PUDS (dispositivo móvil).**<br>Administración de la cámara del dispositivo móvil a partir del servidor Web que ha procesado la petición de un cliente móvil. (Configuración de vigilancia y evidencias) y el desarrollo de una aplicación nativa para la comunicación entre el servidor Web y el móvil de detección de movimiento y de personas. |

# Capítulo 5

# Implementación

## 5.1. Arquitectura de software en capas

El presente trabajo de tesis propone una arquitectura en capas para el desarrollo de sistemas distribuidos de video-vigilancia inteligentes. La arquitectura propuesta está estructurada en capas. Cada capa proporciona ciertos servicios a las capas superiores, ocultando los detalles de los servicios ofrecidos por las capas inferiores. Para el diseño de la arquitectura se consideraron principalmente dos aspectos: las fases del ciclo de seguridad definidas en [3] y los servicios que un sistema de video-vigilancia de tercera generación debe cumplir.

La arquitectura cumple con las fases del ciclo de seguridad que son: 1) la fase de protección que comprende el uso cámaras localizadas en ciertas zonas las cuales capturarán los hechos actuales, 2) la fase de detección que incluye la implementación de algún medio hardware o software que permita identificar hechos anormales y finalmente, 3) la fase de respuesta que considera la activación de alarmas que advierte la identificación del suceso anormal.

La Figura 5.1 muestra la arquitectura en capas de este trabajo que se diseñó a partir de los trabajos de diva-2008,face,aswin-2009, cisp-2008,real-2004 que sólo incluyeron ciertos elementos (visión por computadora, controla del sistema, gestión e información y evidencias, diseño secuencial y síncrono, además de adaptarse en diferentes contextos) y en algunos otros casos se excluyeron.

Es importante mencionar que la arquitectura propuesta cuenta con cuatro capas: interconexión, protección, análisis e identificación de sucesos anormales y respuesta, que se encuentran colocadas de manera ascendente. Este orden se debe respetar en el desarrollo y funcionamiento de cualquier sistema de video-vigilancia.

A continuación se describe cada una de estas capas y los módulos que las integran.

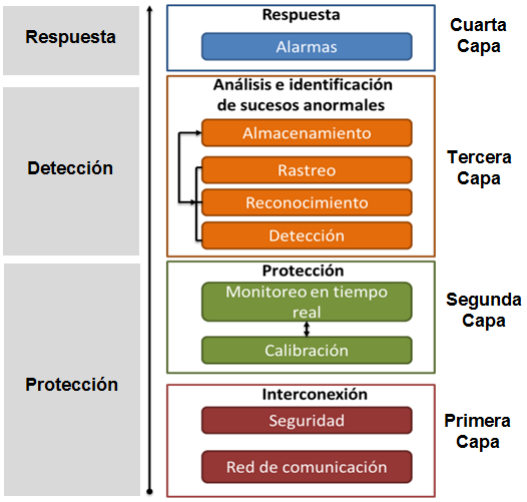



Figura 5.1: Arquitectura de software en capas para sistemas de video-vigilancia de tercera generación, (los rectángulos en gris representan las fases del ciclo de seguridad [3].

### 5.1.1. Capa de interconexión

Es la capa base ya que su función es lograr la comunicación de todos los elementos que intervienen con el sistema de video-vigilancia. Esta capa está integrada por los módulos de red de comunicación y de seguridad, los cuales son descritos enseguida.

#### Módulo Red de Comunicación

El módulo considera aquellos elementos físicos y lógicos que intervienen en la comunicación para satisfacer los objetivos que propone el sistema de video-vigilancia y es necesario definir los elementos y características de cada uno de ellos. Por ejemplo en los elementos físicos el número y tipo de cámaras (IP, Webcam, celulares, robotizadas), los dispositivos de red (routers, switches, servidores), la red (cableada o inalámbrica), los dispositivos o terminales de usuario final (computadoras, móviles, televisores). Por otra parte, en los elementos lógicos que consideran el modelo de comunicación (modelo cliente-servidor, punto a punto, etc.) y los protocolos de comunicación (TCP/IP o UDP, RTSP, entre otros).

#### Módulo Seguridad

El propósito de este módulo se refiere a la seguridad del sistema en dos aspectos: flujos de datos y usuarios que son descritos a continuación:

- Flujos de datos (video). Todo flujo emitido y recibido debe llegar sin afectaciones ni pérdidas, además de evitar que usuarios externos y no autorizados intenten acceder y manipular los contenidos.

- Usuarios. El acceso a los servicios del sistema deben validarse (ejemplo el nombre de usuario y contraseña), además de delimitar permisos en relación al tipo de

usuario (administrador o cliente). La infraestructura, programas, protocolos y planes de seguridad son incluidos en esta capa con el fin de minimizar y evitar los usos y ataques indebidos del sistema.

### 5.1.2. Capa de protección

El objetivo de la capa de protección es brindar el servicio de vigilancia y control remoto de las cámaras en tiempo real, a un número determinado de usuarios. En esta capa es necesario definir en primera instancia el tipo de ambiente donde se colocarán las cámaras, (por ejemplo; en el interior o exterior de un edificio), en un ambiente controlado (debe cumplir con ciertas características) o dinámico (no hay restricciones) o combinación de éstos. El definir el tipo de ambiente ayudará a implementar con éxito los dos módulos que forman esta capa: la calibración y el monitoreo estos se describen enseguida:

#### Módulo de calibración

Consiste en determinar los parámetros de una cámara con el fin de realizar las tareas de monitoreo y posteriormente el análisis de video. Los parámetros se dividen en internos (distancia focal, factores de distorsión y puntos centrales del plano imagen) y externos (posición y orientación) [14, 15].

Para la calibración de la cámara se determina la geometría, los parámetros internos y externos que normalmente son calculados de dos formas: 1) mediante un patrón de calibración (marco de referencia del mundo) y 2) mediante un método de autocalibración, el cual se basa en el movimiento de la cámara observado una escena estática, tomando el desplazamiento y usando la imagen que restringe los parámetros.

Existen diferentes técnicas y algoritmos de calibración que pueden emplearse en este módulo para determinar los parámetros de cada una de las cámaras que integran al sistema [15].

**Módulo monitoreo en tiempo real**

Este módulo ofrece a los usuarios remotos visualizar lo que sucede actualmente. Para lograr el servicio, se debe establecer la manera en que las cámaras capturan el flujo de video y lo codifican mediante un encoder, el cual convierte y comprime el flujo a un formato del video (ejemplo mp4, h.264, etc.) para que posteriormente se trasmita y visualice en el cliente.

El servicio de monitoreo le permite al usuario acceder en cualquier instante de tiempo que él disponga para supervisar las zonas. Para el usuario final es trasparente la arquitectura física del sistema distribuido y la captura de video.

Dentro de este módulo también se incluye el aspecto de manipular las cámaras de manera remota, es decir el usuario tiene el control de las características propias de las cámaras como ampliar o reducir la visualización de la zona (zoom), encender luces, sirenas, sistema de voz, cambiar la posición y orientación.

### 5.1.3. Capa de análisis e identificación de sucesos anormales

Consiste en identificar y almacenar sucesos anormales ocurridos en el área monitoreada. Esta capa consta de cuatro módulos: detección, reconocimiento, seguimiento de objetos y el almacenamiento.

Los módulos detección, reconocimiento y seguimiento de objetos se encargan de

medir las variaciones en el ambiente e identificar situaciones anormales. Dos formas en las que pueden desempeñar estas tareas son: 1) mediante el análisis o tratamiento de imágenes del flujo de video captada por las cámaras (ejemplo el uso de algoritmos de visión por computadora) y 2) mediante dispositivos electrónicos como los sensores de movimiento, de sonido, de humo, de temperatura, luz infrarroja, etc. Cada uno de estos módulos tiene un objetivo en particular que se describe a continuación.

**Módulo de detección**

Consiste en localizar una situación anormal, ejemplo: detectar humo, cambios de temperatura, movimiento (algoritmos background, motion template entre otros [13,14]).

**Módulo de reconocimiento**

La función de este módulo es exclusivamente definir o identificar de qué situación anormal se trata, por ejemplo el reconocimiento de objetos animado e inanimado en particular (objetos olvidados, personas) así también, de comportamientos sospechosos de los humanos como hechos violentos y conglomeraciones.

**Módulo de rastreo**

El objetivo de este módulo consiste en que una vez identificado el foco del suceso anormal éste no debe perderse de vista por ejemplo, un objeto sospechoso en movimiento.

Los módulos detección, reconocimiento y seguimiento de objetos se deben ejecutar de manera distribuida es decir para cada una de las cámaras sin afectar el monitoreo.

#### Módulo almacenamiento

La tarea de este módulo consiste en realizar la grabación de evidencias (fotografía, video y variaciones en el ambiente) y el almacenamiento en una base de datos de tipo relacional o distribuida, estableciendo los datos de la cámara proveniente, el tipo de evidencia, fecha y hora de su captura, con el fin de hacer un análisis futuro de las evidencias.

Concluida la grabación y el registro en la base de datos, el sistema tiene la capacidad de proveer servicios como visualizar las evidencias (recuperación de los datos), hacer consultas desde una aplicación web o desde un móvil a la galería de multimedia (fotos o videos bajo demanda) de los últimos días monitoreados.

### 5.1.4. Capa de respuesta

Esta capa ofrece el servicio de informar en tiempo real sobre los hechos anormales identificados. Comprende el módulo de alarmas que se describe enseguida.

#### Módulo de alarmas

En este módulo se incluyen aquellos elementos o dispositivos que permitan advertir sobre la situación anormal, algunos ejemplos de alarmas son las siguientes:

- Llamadas telefónicas.
- Mensaje de texto (SMS) que incluyen una leyenda sobre el hecho anormal.
- Mensaje multimedia (MMS) contiene una imagen / un pequeño video de lo ocurrido.
- Sirenas y luces.

Dentro del módulo de alarmas pueden ser incluidos opcionalmente elementos que permitan realizar acciones para tratar de resolver la situación anormal por ejemplo, un detector de humo advierte sobre un incendio y activa un sistema contraincendios (rociadores).

# Bibliografía


[1] Android 4.0. what is android?, 2011. Copyright Android 4.0,Revisado 14 de noviembre, 2011. Disponible en http://developer.android.com/guide/basics/what-is-android.html.

[2] Tanenbaum S. A. *Sistemas Operativos Distribuidos*. Prentice Hall, United States of America, 1996.

[3] Maloof A.M. *Machine Learning and Data Mining for Computer Science*. Springer, United States of America, 2006.

[4] Wang C., Mao Z., Zhnag Y., and Luo H. The mobile video surveillance system based on wireless lan [ol], 2003.

[5] dollimore J. Coulouris G. and Kindberg T. *Sistemas Distribuidos Conceptos y Diseño*. Addison Wesley, España, 2003.

[6] Gerónimo D. Haar-like feature and integral image representation, 2009. Çomputer Vision Center, Universidad de Barcelona,.

[7] Vallejo F. D. Zeroc ice, 2006. Copyright Vallejo F. D, Universidad de Castilla-la Mancha, Escuela Superior de Informática.

[8] Periódico El Economista S.A. de C.V. México, atractivo para el negocio de la videovigilancia, 2010. Copyright © 1994-2011. Revisado 4 de septiembre, Disponible en http://eleconomista.com.mx/seguridad-publica/2010.

[9] Morales E. Ensambles de clasificadores, 2009. Copyright Morales E., Revisado 31 de octubre, 2011. Disponible en http://ccc.inaoep.mx/ emorales/Cursos/Aprendizaje2/node4.html.

[10] Gedikli E. et al. Human motion detection, tracking and analysis for automated surveillance. *Department of Computer EngineeringKaradeniz Technical University.*

[11] Miller A. et al. Person and vehicle tracking in surveillance video. *Evaluation Campaign and Workshop CLEAR*, 174-178 2007.

[12] Bradski G. and Kaehler A. *Learning OpenCV.* O´reilly, United States of America, 2008.

[13] Derpanis K. G. Integral image-based representations, 2007. "York University, USA.

[14] Gonzales G. Sistema de vigilancia ip vía teléfono móvil, 2001. Revisado 1 de septiembre, 2001. Disponible en www.rnds.com.ar, www.flexisoft.com.

[15] Guevara C. H. and Real E. Sistema de control y monitoreo integrado con wireless application protocol (wap). Reporte técnico, Universidad Peruana de Ciencias Aplicadas, Lima, Perú., 2000.

[16] Song H., Feng X., Zhao Q., Chi Y., and Yeng H. Mobil video surveillance system of 3g network base on arm9. *IEEE e ICCASM (International Conference on Computer Application and System Modeling)*, pages 400–403, 2010.

[17] Davis J. Representing and recognizing human motion: From motion templates to movement categories. *Digital Human Modeling Workshop, International Conference on Intelligent Robots and Systems*, octubre 2001.

[18] Davis J. and Bradski G. Real-time motion template gradients using intel cvlib. *IEEE ICCV Workshop on Frame-rate Vision*, septiembre 1999.

[19] San Miguel J.C, Bescós J., Martínez J., and García A. Diva: A distributed video analysis framework applied to video-surveillance systems. *IEEE computer society. Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, pages 207–210, 2003.

[20] Davis K., Kelsey J., Langellier D., Mapes M., and Rosendahl J. Security cameras, 2003. CTER Program at UIUC. Revisado 31 de agosto, 2011. Disponible en http://students.ed.uiuc.edu/jkelsey/surveillance.

[21] Shafique H. K. Computer vision, 2003. Copyright UCF, Revisado 3 de noviembre, 2011. Disponible en http://www.cs.ucf.edu/courses/cap6411/cap5415/.

[22] Xu L., Wang Z., Wang H., Shi A., and Li C. A j2me-based intelligent video surveillance system using moving object recognition technology. *Congress on Image and Signal Processing CISP. 2008*, 2:281–285, 2008.

[23] Henning M. A new approach to object-oriented middleware. *IEEE Internet Computing*, 66-75 2004.

[24] Valera M. and Valastin S. A. Intelligent distributed surveillance systems: a review. *IEEE Processing Vision, image and signal. Image signal Process*, 152(2), 192-204 2005.

[25] Valera M. and Velatin S.A. Real-time architecture for a large distributed surveillance system. *IEEE Intelligent Distributed Surveilliance Systems*, pages 41–45, Feb 2004.

[26] Bolliger P., Köhler M., and Römer K. Facet: Towards a smart camera network of mobile phones. *AUTONOMICS. 1st InternationalConference on Autonomic Computing and Communication Systems ICST*, May 2007.

[27] Kurman P., Miltan A., and Kurman P. Study of robust and intelligence in visible and multi-modal framework. *Informatica (Slovenia)*, pages 63–77, 2008.

[28] Cehn Q. Hand detection with a cascade of boosted classifiers using haar-like features, 2006. Discover Lab, SITE, University of Ottawa.

[29] Zang Q. and Klette R. Object classification and tracking in video surveillance. In *Lecture Notes in Computer Science*, volume 2756, pages 198–205. Springer Berlin, 2003.

[30] Calvo P. R. Sistema distribuido de vídeo-vigilancia basado en android, 2010. Copyright Calvo P. R, Universidad Rey Juan Carlos.

[31] Aswin S., Snthiyan Sethuram A., Varun A., and Vasanth P. A j2me-based wireless automated video surveillance system using motion detection method. Reporte técnico, Departament of Electronics and Communication Engineering, AMRITA School of Engenrering, Coimbatore, India, Abril 2009.

[32] Rusell S and Norving P. *Artificial Intelligence a Modern Approach*. Pearson, United States of America y Canadá, 2010.

[33] Dietterich G. T. Machine-learning research four current directions. *AI Magazine (© AAAI)*, 18(4), 1997.

[34] Dietterich G. T. Ensamble methods in machine learning. *International Workshop on Multiple Classifier Systems*, 2000.

[35] Michell M. T. *Machine Learning.* McGraw-Hill, United States of America, 1997.

[36] Unomásuno. Demanda al d.f. colocar cámara de video vigilancia en eventos masivos del df, 2011. Copyright © 2011. Revisado 5 de septiembre 2011, Disponible en http://www.unomasuno.com.mx/notimomento.

[37] International Data Corporation IDC Corporate USA. Worldwide smartphone market expected to grow 55 % in 2011 and approach shipments of one billion in 2015, 2011. Copyright © IDC 2011, Revisado 22 de septiembre, 2011. Disponible en-http://www.idc.com/getdoc.jsp?containerId=prUS22871611.

[38] Verman V., Thrun S., Matas J., and Sochman J. Machine learning: Boosting. *Copyright Stefan Roth, Departament of Computer Science Technische Univeritat Darmtadt*, 2009.

[39] Wilson I. P. y Fernandez J. Facial feature detection using haar classifiers. *Journal of Computing Sciences in Colleges*, pages pp 127–133, 2000.

[40] Viola P. y Jones M. Rapid object detection using boosted cascade of simple features. *Computer Vision and Pattern Recognition*, 2001.

[41] Kshemkalyani D. A. y Singhal M. *Distributed computing principles, algorithms, and systems.* Cambridge, United States of America, 2008.