Universidad
Carlos III de Madrid

Trabajo Fin de Grado

Detección de eventos en secuencias con multitudes

Fernando de la Calle Silos
Grado en Ingeniería de Sistemas Audiovisuales

Tutor:
Iván González Díaz

Índice general

Índice de Figuras

Índice de Tablas

1

Introducción y objetivos del proyecto

1.1. Marco y motivación del proyecto

Cada día los sistemas de vídeovigilancia son más imprescindibles en lo que a seguridad concierne, por lo que el número de cámaras y, por tanto, de horas de grabación han aumentado de forma significativa en los últimos años. De este hecho surge la necesidad de crear sistemas de reconocimiento automático que permitan detectar diferentes situaciones o eventos, que resulten significativos desde el punto de vista de la seguridad. La creación de estos sistemas y su evolución permitirá, por una parte, reducir el personal de vigilancia encargado de la visualización de los vídeos y, por otra, aumentar el número de cámaras sin preocuparse por el visionado humano de estas grabaciones.

La seguridad en lugares públicos en los que hay presentes grandes multitudes, como pueden ser aeropuertos, conciertos, estaciones, etc, es de gran interés en la actualidad. Por ello resulta necesaria la aparición de sistemas de reconocimiento de eventos anómalos, que informen rápidamente cuando están ocurriendo situaciones críticas, permitiendo así la actuación de los cuerpos de seguridad en caso necesario.

Se han realizado múltiples estudios sobre vídeo vigilancia, adquiriendo este campo de investigación una gran importancia. Aunque cada estudio aporta nuevos métodos a aplicar, aún es necesario realizar grandes esfuerzos en este ámbito, que permitan conseguir sistemas con aplicaciones reales. Como se podrá ver a lo largo del presente documento, se ha realizado una extensa revisión bibliográfica de todos los artículos y estudios que pudiesen ser relevantes para la realización de este trabajo.

Para poder abordar el tema de la detección de eventos en secuencias audiovisuales, será necesario recurrir a diversas técnicas de procesado de imagen y aprendizaje máquina. Estos dos ámbitos también resultan de especial interés hoy en día para la comunidad investigadora.

Por último, la motivación de la realización de este proyecto también se debe a intentar mejorar los sistemas propuestos hasta el momento, aportando nuevas técnicas y procesos que aumenten el rendimiento sobre el estado del arte.

1.2. Objetivos del proyecto

El objetivo de este proyecto es la implementación de un sistema de reconocimiento automático de eventos anómalos en secuencias de vídeo donde se vean involucradas un gran número de personas.

Para desarrollar y probar el sistema se ha utilizado la base de datos *PETS Dataset S3, High Level* [62], que contiene siete secuencias de vídeo en las aparecen los siguientes eventos:

- *Walking:* Representa a un número significativo de personas desplazándose lentamente.
- *Running:* Representa a un número significativo de personas desplazándose rápidamente.

- *Evacuation:* Representa la dispersión rápida en diferentes direcciones de una multitud.
- *Crowd Formation:* Representa la unión en un grupo de un gran número de individuos provenientes de diferentes direcciones.
- *Crowd Splitting:* Representa la división de un grupo de individuos, en dos o más grupos que toman diferentes direcciones.
- *Local Dispersion:* Representa la dispersión de un pequeño grupo de individuos de una multitud.

Debido a la necesidad de analizar el movimiento para detectar los distintos eventos, la mayoría de los sistemas propuestos en la literatura ([25], [18], [9]) se organizan en varias etapas, tal y como se representa en la figura 1.1.

Fig. 1.1: Etapas del sistema de reconocimento de eventos implementado.

Estas etapas son:

1. **Detección:** El objetivo de la etapa de detección consiste en discriminar los objetos de interés (personas o grupos de personas) del fondo de la imagen, empleando distintos algoritmos de sustracción de fondo.
2. **Tracking:** El objetivo de esta etapa consiste en realizar un seguimiento de cada objeto detectado durante la secuencia, obteniendo sus vectores de movimiento.
3. **Reconocimiento de Eventos:** En esta etapa se extraen las diferentes características que modelan los distintos eventos para, posteriormente, usando técnicas de aprendizaje maquina determinar cuando estos ocurren.

Por lo tanto, el objetivo de este proyecto es el diseño e implementación de soluciones para cada una de las etapas identificadas. En algunos casos, la literatura proporcionará soluciones válidas para el problema mientras que, en otros, será necesario desarrollar nuevas técnicas que mejoren el estado del arte.

Asimismo, en los capítulos correspondientes se expondrán los objetivos particulares a alcanzar en cada una de las etapas del sistema.

1.3. Estructura de la Memoria

La memoria de este proyecto está organizada en siete capítulos. A continuación se presenta una breve descripción de cada uno:

- **Capítulo 1** Se presenta la motivación del proyecto y la situación actual de los temas a tratar, definiendo además los objetivos que se pretenden alcanzar y la estructura de la memoria.
- **Capítulo 2** Se describe la etapa de detección, explicando detalladamente distintos algoritmos de sustracción de fondo y realizando una comparación y combinación de los mismos que permita obtener el mejor rendimiento.
- **Capítulo 3** Se describe la etapa de *tracking*, detallando todos elementos de ésta, como son la extracción de puntos característicos, la descripción de estos y el cálculo de los vectores de movimiento.

- **Capítulo 4** Desarrollo del sistema de detección de eventos a partir de los resultados de las dos etapas anteriores. Se detallan los descriptores de bajo nivel que han sido implementados, así como el proceso de selección de los más adecuadas para cada evento. Por último se realiza una descripción del sistema de clasificación empleado.
- **Capítulo 5** Se presentan los resultados obtenidos por el sistema implementado, comparándose estos con los sistemas presentados en otros artículos, los cuales constituyen el estado del arte en este problema.
- **Capítulo 6** En este capítulo se presentan las conclusiones más destacadas del estudio realizado, y se identifican las limitaciones y debilidades más notables del sistema propuesto. Además, se plantean algunas de las líneas futuras que se podrían seguir a partir de este proyecto.
- **Capítulo 7** Se detalla el presupuesto económico de la realización del proyecto.
- **Apéndice A** Descripción detallada de la base de datos PETS 2010.
- **Apéndice B** Formulación del modelo de mezcla de Gaussianas utilizado en el capítulo 2.
- **Apéndice C** Descripción del algoritmo RANSAC utilizado en el capítulo 4.

2

Detección

2.1. Introducción y objetivos

El objetivo de la etapa de detección consiste en discriminar los objetos de interés (personas o grupos de personas) del fondo de la imagen, para después realizar un *tracking* de estos durante la secuencia de vídeo. El método de detección empleado se basa en la sustracción del fondo, *background substraction* [54], permitiendo extraer de cada imagen los objetos de interés mediante una serie de algoritmos que modelan la apariencia del fondo.

Esta etapa es una de las más complejas del sistema. Al no ser las secuencias de vídeo ideales, surgen diferentes problemas como cambios de iluminación en la escena, sombras, objetos del fondo que aparecen y desaparecen, etc.

Para desarrollar el sistema de detección se han implementado diferentes métodos de sustracción de fondo, que han sido evaluados con el fin de utilizar en el sistema final el que mejores resultados proporcione. En la sección 2.2 se explican los métodos de sustracción de fondo más utilizados en la literatura, en la 2.3 se detallan los distintos métodos implementados. El método de evaluación utilizado para comparar las distintas técnicas se explica en la sección 2.4, los resultados obtenidos se detallan en la 2.5, y en la 2.6 se explica cómo se ha tratado la salida de estos métodos y la combinación de varios de ellos.

2.2. Estado del Arte

Los algoritmos de sustracción de fondo se pueden dividir en varias etapas, tal y como se observa en la figura 2.1.

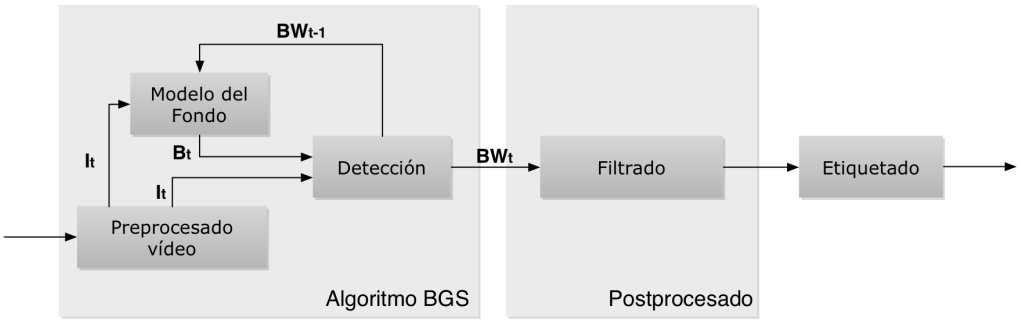


Fig. 2.1: Diagrama de flujo de un sistema típico de substracción de fondo en aplicaciones de vídeo vigilancia, se observa el bloque del algoritmo de sustración de fondo BGS (*BackGround Sustraction*) y los bloques de postprocesado y etiquetado.

A partir de un vídeo, correspondiente a cada cámara estática, se hace un modelo de escena de dicha cámara, B_t asociado al plano I_t en el instante t. En la etapa de detección, utilizando el modelo de fondo, se deciden los píxeles que pertenecen al fondo y los que pertenecen al primer plano, dando como resultado una máscara binaria BW_t.

Esta máscara puede ser calculada de diversas maneras, dependiendo del tipo de modelo de fondo. En algoritmos simples basta con calcular la diferencia píxel a píxel del modelo de fondo con cada plano y aplicar un umbral. Si se cumple que:

$$|I_t(x,y) - B_t(x,y)| < Th \tag{2.1}$$

El píxel de la posición (x,y) se considera fondo. Este método presenta el problema de que el umbral Th es común para todos los píxeles.

En otros algoritmos el modelo del fondo esta definido por una función de probabilidad condicional. Aplicando un umbral podemos clasificar cada píxel:

$$p(I_t(x,y)|B_t(x,y)) < Th(x,y) \tag{2.2}$$

En algunos métodos se tiene en cuenta la estimación de la varianza de cada píxel $\sigma(x,y)$ de la siguiente manera:

$$\frac{p(I_t(x,y)|B_t(x,y))}{\sigma(x,y)} < Th(x,y) \tag{2.3}$$

lo que proporciona mejores resultados. Esto se verá más detalladamente en secciones posteriores, al haberse implementado métodos de este tipo.

Como se observa en la figura 2.1, pueden existir etapas en las que se realice un postprocesado de la salida BW_t para tratar de eliminar algunos píxeles que se consideren como fondo debido a pequeños cambios de iluminación o ruido. Por último se realiza un etiquetado de los objetos detectados.

Por tanto, las características que definen a un algoritmo de sustracción de fondo son: cómo se define y como se actualiza el modelo del fondo. Según esto se pueden dividir los distintos algoritmos en función de si utilizan técnicas recursivas o no.

A continuación se describen algunos algoritmos según esta división. Algunos han sido implementados en el proyecto por lo que están explicados de forma más detallada en la sección 2.3.

2.2.1. Técnicas recursivas

Las técnicas recursivas mantienen un modelo de fondo que se actualiza en cada plano. Estas técnicas son computacionalmente muy eficientes y presentan unos requisitos de memoria mínimos. A continuación se presenta brevemente algunos de estos métodos.

Resta plano anterior

En este método el modelo de fondo es simplemente el plano anterior del vídeo $B_t = I_{t-1}$.

Running Gaussian Average

Si se considera que el fondo es muy estático, y la mayor variación que existe es debida al ruido que introduce la cámara, se puede modelar cada píxel del modelo de fondo como una distribución Gaussiana [70]. Este modelo se explica detalladamente en la sección 2.3.1.

Gaussian Mixture Model

Para modelar fondos multimodales, en los que cada píxel de la imagen puede tomar diversos valores, en [63] se propone modelar cada canal de color de cada píxel como una mezcla de K Gaussianas. Modelar el fondo de esta manera ha sido explorado en numerosos artículos ([12], [55], [72]). Este aproximación se explica detalladamente en la sección 2.3.2.

GMM con selección adaptativa del número de Gaussianas (AGMM)

Una implementación interesante del modelado del fondo mediante GMM se expone en [72], donde el numero de Gaussianas se elige adaptativamente a partir de los datos de entrada, reduciendo las necesidades de memoria y mejorando el rendimiento en fondos que cambian muy frecuentemente.

2.2.2. Técnicas no recursivas

Las técnicas no recursivas mantienen un *buffer* de los n planos anteriores y estiman el modelo de fondo en base a las propiedades estadísticas de estos. Estas técnicas requieren unos requisitos de memoria mayores que las recursivas; sin embargo, al poder acceder a los n últimos planos, pueden modelar aspectos del fondo que las técnicas recursivas no pueden.

Filtro de mediana

En el filtro de mediada, cada píxel del modelo de fondo es la mediana del valor del píxel en el *buffer* de planos.En [8] se propone el siguiente algoritmo:

1. Ordenar el *buffer*: $E(x,y) = \left(I_{t-n}(x,y), ..., I_{t-2}(x,y), I_{t-1}(x,y), B_{t-1}(x,y)\right)$ en orden ascendente. El *buffer* incluye el valor del último modelo del fondo para ser más robusto al ruido cuando se utilizan *buffers* pequeños.
2. El modelo del fondo es la mediana del buffer: $B_t(x,y) = E_{\frac{n+1}{2}}(x,y)$
3. Estimar la varianza: $\sigma(x,y) = \lambda\left(E_{\frac{n+1}{2}+k}(x,y) - E_{\frac{n+1}{2}-k}(x,y)\right)$ donde k y λ son parámetros.
4. Umbralizar $I_t(x,y)$, teniendo en cuenta la estimación de la varianza para obtener la mascara binaria BW_t que indica los píxeles que pertenecen al fondo.

$$BW_t(x,y) = \begin{cases} 0 & \text{si} \quad |I_t(x,y) - B_t(x,y)| < Th \cdot \sigma_t(x,y) \\ 1 & \text{si} \quad |I_t(x,y) - B_t(x,y)| \geq Th \cdot \sigma_t(x,y) \end{cases} \tag{2.4}$$

Eigenbackgrounds (EigBG)

Todos los algoritmos propuestos hasta el momento modelan el fondo de forma independiente para cada píxel. En [51] se propone un método que intenta capturar correlaciones espaciales aplicando el análisis de componentes principales a un *buffer* de N planos. Este conjunto de planos da lugar a las funciones base que capturan la apariencia de la escena.

Un nuevo plano puede ser proyectado en estas funciones base (*Eigenbackgrounds*), las cuales modelan únicamente partes estáticas de la escena. Al volver a proyectar de nuevo en el espacio original la imagen, ésta no contendrá las zonas que no son estáticas. Por ello las funciones base pueden ser usadas como modelo de fondo. Una de las limitaciones del método es que, para calcular las funciones base, se requiere un conjunto de imágenes que no contenga los objetos que estamos intentando detectar.

2.3. Métodos de subtracción de fondo implementados

Puede realizarse una clasificación de los métodos implementados en dos tipos: los que se basan en modelos de actualización temporal y los modelos entrenados mediante un conjunto de secuencias de vídeo en las que no aparecen personas.

Estos métodos proporcionan una imagen binaria para cada plano: BW_t, en la que cada píxel, $BW_t(x,y)$, valdrá 0 si el píxel pertenece al fondo y 1, si es un objeto de interés (una persona o grupo de personas).

2.3.1. Modelos de actualización temporal

Resta del Plano Anterior

Los métodos más sencillos se basan en calcular la diferencia $D(x, y)$ entre la imagen de la que se quiere extraer el fondo $I_t(x, y)$, y el fondo estimado $B_t(x, y)$. Posteriormente se puede aplicar un umbral a esta diferencia $D(x, y)$ para obtener la mascara binaria $BW_t(x, y)$ que indica los píxeles que pertenecen al fondo.

Como fondo estimado de este método se toma el plano anterior $B_t(x, y) = I_{t-1}(x, y)$; por tanto, la imagen diferencia queda definida como:

$$D(x, y) = |I_t(x, y) - I_{t-1}(x, y)| \tag{2.5}$$

La imagen binaria $BW_t(x, y)$ se obtiene mediante:

$$BW_t(x, y) = \begin{cases} 0 & \text{si} \quad D(x, y) < T \\ 1 & \text{si} \quad D(x, y) \geq T \end{cases} \tag{2.6}$$

Donde T es un umbral elegido de forma empírica.

Media Móvil

Algunos autores ([44] y [15]) proponen estimar el fondo utilizando las imágenes anteriores. En [44], por ejemplo se propone usar como modelo de fondo la media de los n últimos planos. Así, el fondo se estimaría como:

$$B_t(x, y) = \frac{\sum_{i=t-n}^{t-1} I_i(x, y)}{n} \tag{2.7}$$

Esta forma de estimación requiere almacenar los n últimos valores que toma cada píxel. Para evitarlo, se puede realizar una actualización *on line* del modelo:

$$B_t(x, y) = \alpha \cdot I_{t-1}(x, y) + (1 - \alpha) \cdot B_(t - 1)(x, y) \tag{2.8}$$

Donde α es el parámetro de aprendizaje elegido de forma empírica, teniendo en cuenta que $0 \leq \alpha \leq 1$. Una vez calculado $B_t(x, y)$ obtendríamos la imagen diferencia:

$$D(x, y) = |I_t(x, y) - B_t(x, y)|$$

Se observa que si $\alpha = 1$, $B_t(x, y) = I_{t-1}(x, y)$ y $D(x, y) = |I_t(x, y) - I_{t,1}(x, y)|$, este método se corresponde con la resta del plano anterior.

A partir de $D(x, y)$ se obtendría $BW_t(x, y)$ de una manera similar a lo anterior:

$$BW_t(x, y) = \begin{cases} 0 & \text{si} \quad D(x, y) < T \\ 1 & \text{si} \quad D(x, y) \geq T \end{cases} \tag{2.9}$$

Un forma de mejorar este método consiste en actualizar el modelo en función de cómo ha sido clasificado el píxel:

$$B_t(x, y) = BW_{t-1}(x, y) \cdot B_{t-1}(x, y) + (1 - BW_t(x, y)) \cdot (\alpha \cdot I_{t-1}(x, y) + (1 - \alpha) \cdot B_{t-1}(x, y)) \tag{2.10}$$

En caso de que el píxel de la posición (x, y) sea un objeto de interés, el modelo no se actualizará $B_t(x, y) = B_{t-1}(x, y)$, pero si éste se clasificó como fondo, se aplicará el modelo anteriormente descrito: $B_t(x, y) = \alpha \cdot I_{t-1}(x, y) + (1 - \alpha) \cdot B_{(t-1)}(x, y)$.

Modelo simple de Gauss

En [71] se propone realizar un modelo de fondo independiente para cada píxel $I_t(x, y)$ de la imagen. Éste se basará en realizar un ajuste de una función de probabilidad Gaussiana en los n últimos valores que toma el mismo.

Para evitar tener que ajustar la Gaussiana en cada nuevo plano, se calculan los parámetros (μ, σ) de forma acumulativa, para cada píxel. La media $\mu_t(x, y)$ para cada uno será la siguiente:

$$\mu_{t+1}(x, y) = \alpha \cdot I_t(x, y) + (1 - \alpha) \cdot \mu_t(x, y) \tag{2.11}$$

donde $I_t(x, y)$ es el valor de cada píxel en la imagen actual, $\mu_n(x, y)$ la media calculada anteriormente y α es el parámetro de aprendizaje elegido de forma empírica con $0 \leq \alpha \leq 1$. De forma similar se calcula la desviación tipica:

$$\sigma^2_{t+1}(x, y) = \alpha \cdot (I_t - \mu_t(x, y))^2 + (1 - \alpha) \cdot \sigma^2_t(x, y) \tag{2.12}$$

Los requisitos de memoria para este método son pequeños, ya que sólo necesita almacenar dos valores $(\mu(x, y), \sigma(x, y))$ para cada píxel, en vez de los n últimos valores que toma éste. Cada píxel $I_t(x, y)$ de la imagen se clasifica como fondo u objeto de interés de la siguiente manera:

$$BW_t(x, y) = \begin{cases} 0 & \text{si} \quad |I_t(x, y) - \mu_t(x, y)| < Th \cdot \sigma_t(x, y) \\ 1 & \text{si} \quad |I_t(x, y) - \mu_t(x, y)| \geq Th \cdot \sigma_t(x, y) \end{cases} \tag{2.13}$$

donde Th se calcula de forma empírica.

El modelo del fondo se actualiza independientemente de que el píxel haya sido clasificado como objeto de interés o como fondo. En [37] se propone modificar la actualización del modelo dependiendo de la clasificación del píxel:

$$\mu_{t+1}(x, y) = BW_t(x, y) \cdot \mu_t(x, y) + (1 - BW_t(x, y)) \cdot (\alpha \cdot I_t(x, y) + (1 - \alpha) \cdot \mu_t(x, y)) \tag{2.14}$$

Si éste se clasificó como objeto de interés $BW_t(x, y) = 1$ el modelo no se actualiza: $\mu_{t+1}(x, y) = \mu_n(x, y)$, en caso contrario se aplicará la actualización descrita anteriormente. También [71] propone utilizar este método en imágenes en color.

La aplicación de estos modelos sobre la base de datos utilizada presenta un claro inconveniente. Al comienzo de cada vídeo aparece un gran número de personas que se quiere detectar, teniendo además una escasa duración, por lo que ajustar *on line* una Gaussiana no proporciona buenos resultados. Es por ello que se han estudiado otro tipo de modelos que se presentan en el siguiente apartado.

2.3.2. Modelos entrenados

Los métodos de actualización temporal funcionan correctamente cuando existe movimiento de los objetos a detectar; sin embargo, cuando la secuencia presenta escaso movimiento (las personas están hablando o se mueven lentamente), el rendimiento es bastante pobre.

La base de datos PETS 2010 (más información en el apéndice A) dispone de secuencias de entrenamiento donde no hay presencia de individuos, pudiendose realizar un modelo del fondo usando estas imágenes para, posteriormente, utilizarlo en secuencias donde hay personas y, por tanto, mejorar la extracción.

Modelo simple de Gauss con datos de entrenamiento

Se puede utilizar una función de probabilidad Gaussiana para cada píxel, lo que permitirá modelar el fondo mediante los datos de entrenamiento de forma sencilla. Esta distribución queda definida:

$$\phi(I(x, y)|\mu(x, y), \Sigma(x, y)) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma(x, y)|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (I(x, y) - \mu(x, y)) \Sigma^{-1}(x, y) (I(x, y) - \mu(x, y))^\top\right)$$

Los parámetros del modelo (media $\mu(x,y)$ y varianza $\Sigma(x,y)$ de cada píxel) se estiman a partir de los datos de entrenamiento:

$$\mu(x,y) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} I_i(x,y) \tag{2.15}$$

$$\Sigma(x,y) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(I_i(x,y) - \mu(x,y)\right)\left(I_i(x,y) - \mu(x,y)\right)^T \tag{2.16}$$

Es de esperar que los valores tomados por los píxeles pertenecientes a las personas (objetos de interés), no pertenezcan a la Gaussiana estimada mediante los datos de entrenamiento. En la figura 2.2 se observa que las personas pertenecen a una de las Gaussianas (amarilla) y el fondo pertenece a la otra (azul).

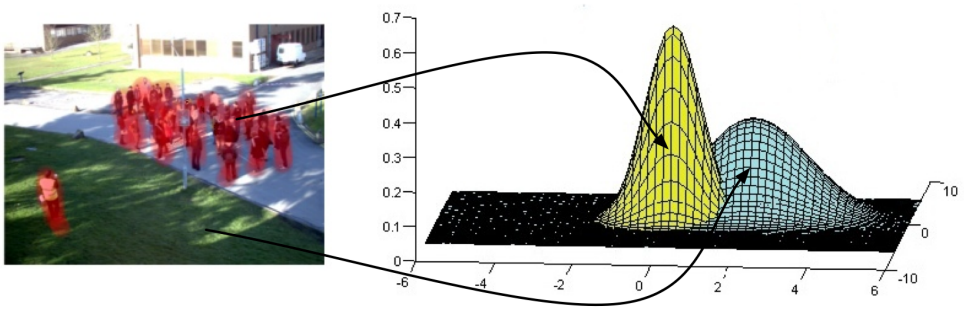


Fig. 2.2: Modelando el fondo mediante una Gaussiana estimada apartir de los datos de entrenamiento (en la figura la Gaussiana azul). Es de esperar que las personas no pertenezcan a esta distribución, sino a otra (en la figura la Gaussiana amarilla).

Se comienza con las imágenes en escala de grises, obteniendo para cada píxel de la imagen su media $\mu(x,y)$ y su varianza $\sigma(x,y)$ en el conjunto de entrenamiento (utilizando las fórmulas 2.15 y 2.16).

Para estimar si un píxel cuyo valor en escala de gris es $I_t(x,y)$ pertenece o no al fondo, se comprueba si éste pertenece a la Gaussiana obtenida mediante los datos de entrenamiento o si, por el contrario, está fuera de esta. El píxel será considerado como fondo cuando su valor $I_t(x,y)$ este comprendido entre $|\mu(x,y) - Th \cdot \sigma(x,y)|$ y $|\mu(x,y) + Th \cdot \sigma(x,y)|$, y como objeto de interés en caso contrario. Th es un umbral que permite elegir la selectividad del método, elegido su valor mediante pruebas experimentales. Matemáticamente se puede expresar como:

$$BW_t(x,y) = \begin{cases} 0 & \text{si} \quad I_t(x,y) < |Th \cdot \sigma(x,y) - \mu(x,y)| \\ 1 & \text{si} \quad I_t(x,y) \geq |Th \cdot \sigma(x,y) - \mu(x,y)| \end{cases} \tag{2.17}$$

Simplificando:

$$BW_t(x,y) = \begin{cases} 0 & \text{si} \quad \dfrac{|It(x,y) - \mu(x,y)|}{\sigma(x,y)} < Th \\ 1 & \text{si} \quad \dfrac{|It(x,y) - \mu(x,y)|}{\sigma(x,y)} \geq Th \end{cases} \tag{2.18}$$

Extensión a color

A continuación se trabaja con las imágenes en color. Se realiza una transformación del espacio de color, pasando de RGB a HSV. El espacio de color HSV es utilizado frecuentemente en tareas de segmentación de objetos en imágenes ([31], [43] y [50]). Algunas ventajas de usar HSV frente a RGB descritas en [31], [60] son:

- El matiz es invariante a ciertos tipos de sombras y luces.
- Las sombras y luces suelen estar aisladas en el canal V (Valor)

Para modelar el fondo mediante datos de entrenamiento con imágenes en color, se asume en una primera aproximación que las tres coordenadas de color (HSV) son independientes entre sí, calculando para cada píxel la media $\{\mu_H(x,y), \mu_S(x,y), \mu_V(x,y)\}$ y la varianza $\{\sigma_H(x,y), \sigma_S(x,y), \sigma_V(x,y)\}$ de cada componente en el conjunto de entrenamiento.

Para estimar si un píxel $I_t^{HSV}(x,y)$ pertenece o no al fondo, se procede de forma similar a cuando se trabajaba en escala de grises:

$$BW_t(x,y) = \begin{cases} 0 & \text{si} \quad \dfrac{|I_t^H(x,y) - \mu_H(x,y)||I_t^S(x,y) - \mu_S(x,y)||I_t^V(x,y) - \mu_V(x,y)|}{\sigma_H^2(x,y)\sigma_S^2(x,y)\sigma_V^2(x,y)} < Th \\ 1 & \text{si} \quad \dfrac{|I_t^H(x,y) - \mu_H(x,y)||I_t^S(x,y) - \mu_S(x,y)||I_t^V(x,y) - \mu_V(x,y)|}{\sigma_H^2(x,y)\sigma_S^2(x,y)\sigma_V^2(x,y)} \geq Th \end{cases} \tag{2.19}$$

Si no se asume que las tres coordenadas de color sean independientes entre sí, se deberá calcular el vector media $\boldsymbol{\mu}(x,y)$ y la matriz de covarianza $\Sigma(x,y)$ de cada píxel en el conjunto de entrenamiento. La nueva formulación para generar la máscara quedaría:

$$BW_t(x,y) = \begin{cases} 0 & \text{si} \quad (\mathbf{I}_t^{HSV}(x,y) - \boldsymbol{\mu}(x,y))^T \Sigma^{-1}(x,y)(\mathbf{I}_t^{HSV}(x,y) - \boldsymbol{\mu}(x,y)) < Th \\ 1 & \text{si} \quad (\mathbf{I}_t^{HSV}(x,y) - \boldsymbol{\mu}(x,y))^T \Sigma^{-1}(x,y)(\mathbf{I}_t^{HSV}(x,y) - \boldsymbol{\mu}(x,y)) \geq Th \end{cases} \tag{2.20}$$

El coste computacional de esta alternativa es muy elevado y, sin embargo, no proporciona resultados notablemente superiores por que se consideraron las componentes de color independientes.

Modelo de mezcla de Gaussianas

Al presentar las imágenes del conjunto de entrenamiento una cierta variabilidad, se puede usar un modelo de Gaussianas o GMM (*Gaussian Mixture Model* [1]) para simularla. Como ejemplo, un píxel perteneciente a una hoja en movimiento sobre un fondo de cielo, toma en ocasiones un color verde y en otras azul. Este cambio de color se podría modelar, mediante un GMM de 2 Gaussianas. Una Gaussiana se ajustará para cuando el píxel sea azul y otra para cuando sea verde, adaptándose así al cambio de color y permitiendo que estos dos valores puedan ser considerados como fondo.

Una mezcla de Gaussianas es una suma ponderada de k Gaussianas, definida como:

$$p(I(x,y)|\theta) = \sum_{j=1}^{k} \omega_j(x,y)\phi(I(x,y)|\mu_j(x,y), \Sigma_j(x,y)) \tag{2.21}$$

donde ω son los pesos de cada una de las componentes de la mezcla, y cumplen que: $\sum_{j=1}^{k} \omega_j(x,y) = 1$ y $\omega_j(x,y) > 0$, y $\phi(I(x,y)|\mu_j(x,y), \Sigma_j(x,y))$ es la Gaussiana asociada a cada componente definida como:

$$\phi(I(x,y)|\mu_j(x,y), \Sigma_j(x,y)) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma(x,y)|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(I(x,y) - \mu(x,y)\right)\Sigma^{-1}(x,y)\left(I(x,y) - \mu(x,y)\right)^{\top}\right)$$

Los parámetros a estimar para cada píxel se definen como $\theta(x,y) = \{(\omega_j(x,y), \mu_j(x,y), \Sigma_j(x,y))\}_{j=1}^{k}$. Para obtener un valor óptimo de dichos parámetros, buscamos el θ que maximice la función de verosimilitud:

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \log P(I_t(x,y)|\theta(x,y)) \tag{2.22}$$

La función de verosimilitud logaritmica $L(\theta(x,y))$ puede definirse mediante n muestras independientes del píxel, tomadas del conjunto de entrenamiento $I_1(x,y), I_2(x,y), \cdots, I_n(x,y)$:

$$L(\theta(x,y)) = \log\left\{\prod_{i=1}^{n} p((I_i(x,y)|\theta(x,y))\right\} = \tag{2.23}$$

$$= \sum_{i=1}^{n} \log p(I_i(x,y)|\theta(x,y)) = \tag{2.24}$$

$$= \sum_{i=1}^{n} \log\left\{\sum_{j=1}^{k} \omega_j(x,y)\phi(I_i(x,y)|\mu_j(x,y),\Sigma_j(x,y))\right\} \tag{2.25}$$

Para maximizar esta función se utiliza el algoritmo *Expectation Maximization* (EM), en el que se plantea una suposición acerca de los datos completos, para encontrar el θ que maximiza el valor esperado de la log-verosimilitud de estos datos. Una vez que se tiene el nuevo θ, es posible realizar una mejor elección. Este proceso se itera hasta que se cumple un cierto criterio de parada. El proceso de ajuste de una mezcla de Gaussianas a unos datos de entrenamiento se describe con profundidad en el apéndice B.

Una vez realizado el ajuste del GMM, se obtienen, para cada píxel de la imagen, los parámetros que definen su modelo de fondo: $\theta(x,y) = \{(\omega_j(x,y), \mu_j(x,y), \Sigma_j(x,y))\}_{j=1}^{k}$.

Para estimar si un píxel $I_t(x,y)$ pertenece al fondo, será necesario umbralizar la verosimilitud de un dato $p\left(I_t(x,y)|\theta(x,y)\right)$ de éste en el GMM:

$$BW_n(x,y) = \begin{cases} 0 & \text{si} \quad p\left(I_t(x,y)|\theta(x,y)\right) > Th \\ 1 & \text{si} \quad p\left(I_t(x,y)|\theta(x,y)\right) \leq Th \end{cases} \tag{2.26}$$

donde Th se ajustó empíricamente para obtener los mejores resultados.

2.4. Evaluación

Para evaluar qué método proporciona mejores resultados se anotaron manualmente las máscaras de las personas en todos los vídeos, usando un submuestreo temporal que proporciona máscaras en uno de cada diez planos. Algunas de estas máscaras, denominadas *ground truth*, se pueden apreciar en la figura 2.3.

Para cada plano etiquetado se ha calculado una medida de calidad de la extracción, propuesta en [53], definida como:

$$r_t = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} BW_t^{\mathrm{GT}}(x,y) \cap BW_t(x,y)}{\sum_{x=1}^{M}\sum_{y=1}^{N} BW_t^{\mathrm{GT}}(x,y) \cup BW_t(x,y)} \tag{2.27}$$

Esta medida es 0 en los planos donde la extracción y la mascara anotada son completamente disjuntas, y 1 sólo cuando coinciden exactamente. Cuanto más cerca esté del valor 1, mejor será la extracción. Para cada alternativa a evaluar, se ha obtenido la media de esta medida en cada una de las secuencias y en el conjunto de las mismas. Estos resultados se muestran en la siguiente sección.

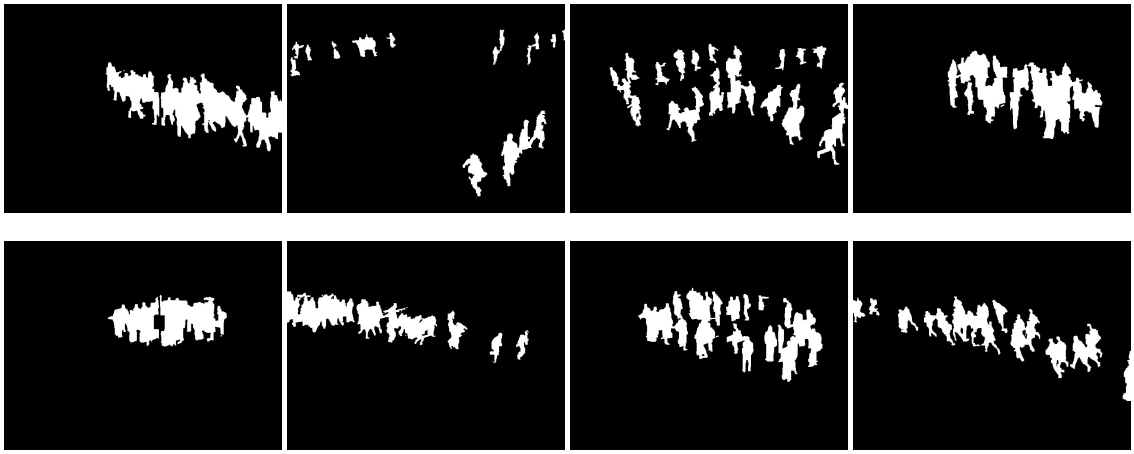


Fig. 2.3: Algunas máscaras anotadas manualmente.

2.5. Resultados

La evaluación de los métodos más sencillos se pueden encontrar en la tabla 2.1, donde se observan los resultados que obtienen las diferentes técnicas para las distintas secuencias denominadas: `14:16-A`, `14:16-B`, `14:27-A`, `14:27-B`, `14:31`, `14:33-A`, `14:33-B` (una descripción de éstas se puede encontrar en el apéndice A).

Técnica	14:16-A	14:16-B	14:27-A	14:27-B	14:31	14:33-A	14:33-B
Resta plano anterior	**0.4081**	**0.4421**	0.1258	0.1133	**0.4080**	0.2174	**0.2528**
Media móvil	0.2734	0.3341	**0.3084**	**0.3057**	0.3402	**0.3670**	0.1751
Modelo simple Gauss Gris	0.2173	0.0776	0.1583	0.1793	0.1320	0.1058	0.1271
Modelo simple Gauss HSV	0.0459	0.0414	0.0910	0.0953	0.0739	0.0579	0.0636

Tabla 2.1: Resultados obtenidos por los métodos más sencillos.

Como se puede observar en la tabla 2.1, la resta del plano anterior y la media móvil obtienen los mejores resultados. En lo que respecta a la media de todas las secuencias, se puede observar en la tabla 2.3, la vencedora es la media móvil. En las secuencias donde las personas apenas se mueven `14:27-A`, `14:27-B` y `14:33-A`, se obtienen los peores resultados, al estar basados estos métodos en el movimiento (en la resta del plano anterior se asume como fondo todo lo que no ha cambiado con respecto al plano anterior).

Para el modelo de mezcla de Gaussianas, como se aprecia en la tabla 2.2, se realizaron pruebas para distinto número de Gaussianas, tanto en escala de grises como en color.

Tipo	Componentes	14:16-A	14:16-B	14:27-A	14:27-B	14:31	14:33-A	14:33-B
Gris	2	0.2043	0.0952	0.2123	0.2273	0.1653	0.1432	0.1663
Gris	3	0.1928	0.0966	0.2079	0.2174	0.1649	0.1363	0.1566
Gris	4	0.1697	0.0919	0.2031	0.2111	0.1584	0.1318	0.1515
HSV	3	0.4587	0.1797	0.3099	0.3226	0.2861	0.2508	0.2866
HSV	4	0.4908	0.2018	0.3245	0.3435	**0.3009**	**0.2638**	**0.3005**
HSV	5	**0.4909**	**0.2050**	**0.3254**	**0.3439**	0.2984	0.2549	0.2949

Tabla 2.2: Resultados obtenidos por la mezcla de Gaussianas.

La mezcla de Gaussianas en color obtiene los mejores resultados. Respecto al número de componentes de ésta, depende de la secuencia, pero en media es ligeramente superior la mezcla de Gaussianas en color de 4 componentes como figura en la tabla 2.3.

Tipo	Media
Resta plano anterior	0.2811
Media móvil	0.3006
Gaussiana Gris	0.1425
Gaussiana HSV	0.0670
GMM 2 Comp Gris	0.1734
GMM 3 Comp Gris	0.1675
GMM 4 Comp Gris	0.1596
GMM 3 Comp HSV	0.2992
GMM 4 Comp HSV	**0.3180**
GMM 5 Comp HSV	0.3162

Tabla 2.3: Media en todas las secuencias que obtienen todos los métodos.

Como se puede observar, los modelos de aprendizaje sobre datos de entrenamiento funcionan más o menos bien en todas las secuencias. El método de mezcla de Gaussianas en color de cuatro componentes resulta el de mejor rendimiento, mientras que los modelos de actualización temporal funcionan muy bien en secuencias donde existe mucho movimiento, aunque de forma incorrecta cuando las personas están quietas.

2.6. Combinación de métodos

La mezcla de Gaussianas funciona bastante bien en todas las secuencias, independientemente del movimiento de éstas, pero surge el problema de que, al estar las secuencias de entrenamiento grabadas en una hora distinta a las de test, en la mascara aparecen zonas de sombras y luces, dado que éstas no estaban en los vídeos de entrenamiento.

Para solucionar este problema se han combinado la máscara de la mezcla de Gaussianas con la de la resta del plano anterior. De esta manera se logra una combinación de los modelos de actualización temporal y los modelos entrenados, que permitirá obtener mejores resultados en todas las situaciones.

Asumiendo que siempre va a existir un ligero movimiento, si dilatamos la máscara de la resta del plano anterior esta da una idea general de donde están las personas y, mediante la mezcla de Gaussianas, se refina la extracción.

Los pasos seguidos se pueden apreciar en la imagen 2.4, y se definen a continuación:

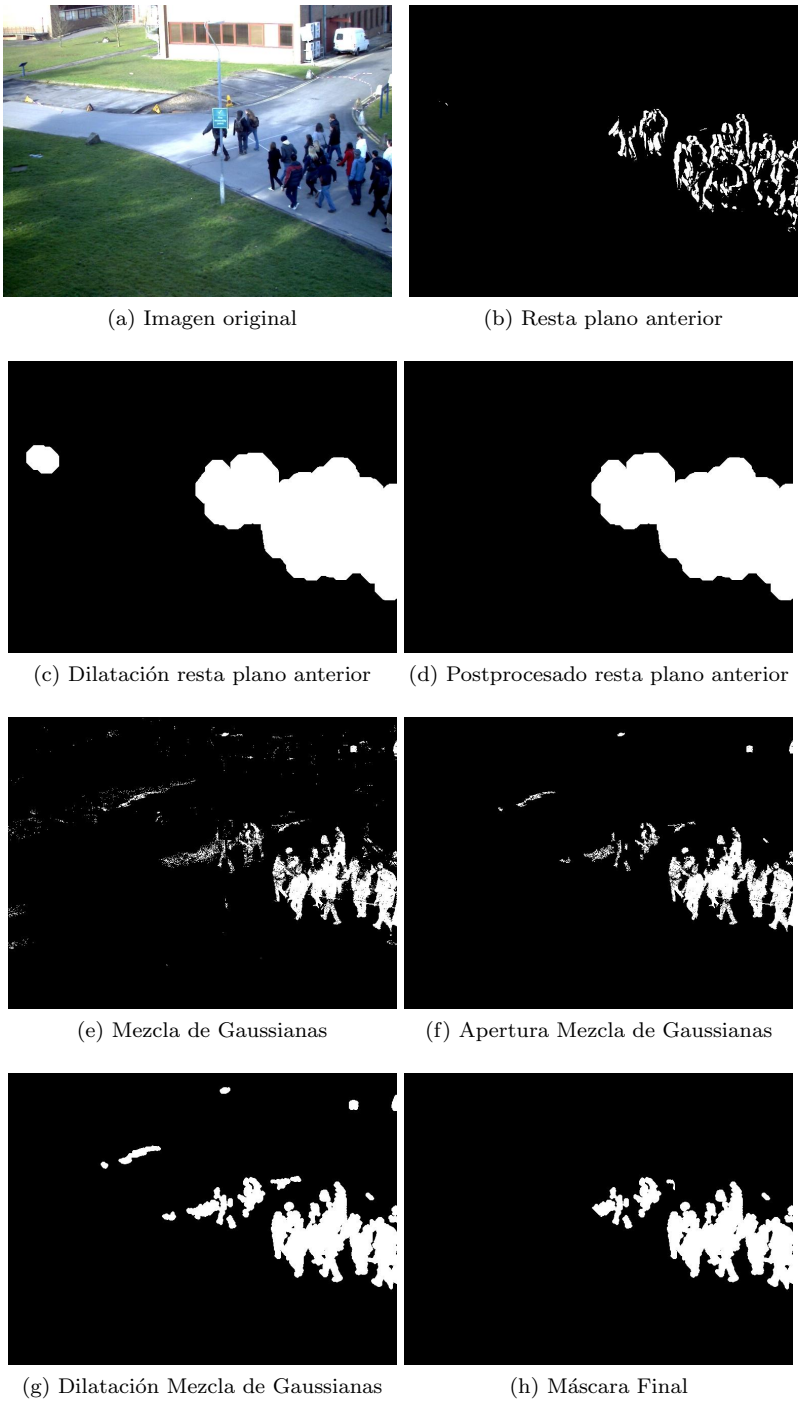
1. A partir de imagen original (figura 2.4a) se obtiene una primera máscara (figura 2.4b) mediante el método de la resta del plano anterior.

2. Se aplica a esta primera máscara, una gran dilatación (figura 2.4c) y se procesa eliminando los objetos menores que el 25 % del de mayor tamaño (figura 2.4d). Esta máscara proporciona una idea general de la posición de las personas.

3. Se obtiene la máscara mediante el método de mezcla de Gaussianas (figura 2.4e). Como se puede observar, en ella aparecen algunas regiones clasificadas como objetos de interés, que no son personas. Esto se debe a las luces y sobras presentes en la imagen.

4. Las máscara procedente del método de mezcla de Gaussianas se procesa morfológicamente realizando en primer lugar una apertura (figura 2.4f), para eliminar los objetos más pequeños y a continuación una dilatación (figura 2.4g).

5. El resultado final 2.4h se obtiene realizando una operación AND entre estas dos mascaras.



(a) Imagen original (b) Resta plano anterior

(c) Dilatación resta plano anterior (d) Postprocesado resta plano anterior

(e) Mezcla de Gaussianas (f) Apertura Mezcla de Gaussianas

(g) Dilatación Mezcla de Gaussianas (h) Máscara Final

Fig. 2.4: Proceso que combina la mezcla de Gaussianas con la resta del plano anterior.

Este proceso asume que siempre va a existir un pequeño movimiento que genera una máscara que al dilatarse proporcionará la región aproximada en la que se encuentran las personas. Esta región se concreta mediante la máscara obtenida por el modelo del mezcla de Gaussianas.

Los resultados obtenidos para este método son excelentes. Como se aprecia en las tablas 2.4 y 2.5 se obtienen mejores resultados para todas las secuencias utilizando la combinación de la

resta del plano anterior con la mezcla de Gaussianas en color de 5 componentes.

Además, este método proporciona mejores resultados que todos los anteriores, como queda claro a la vista de las tablas 2.3 y 2.5

Componentes	14:16-A	14:16-B	14:27-A	14:27-B	14:31	14:33-A	14:33-B
4	0.5321	0.4011	0.4733	0.5361	0.4851	0.4476	0.4511
5	**0.5477**	**0.4348**	**0.4669**	**0.5284**	**0.5024**	**0.4659**	**0.4765**

Tabla 2.4: Resultados de la combinación de mezcla de Gaussianas y resta del plano anterior.

Componentes	Media
4	0.4752
5	**0.4889**

Tabla 2.5: Media en todas las secuencias.

2.7. Conclusiones

A la vista de los resultados expuestos en este capítulo, queda claro que el mejor algoritmo para la detección de individuos, surge de la combinación de la mezcla de Gaussianas con el método de la resta del plano anterior. En las figuras 2.5 y 2.6 se muestran una serie de máscaras, a modo de ejemplo, en las que se observa la eficacia de este método.

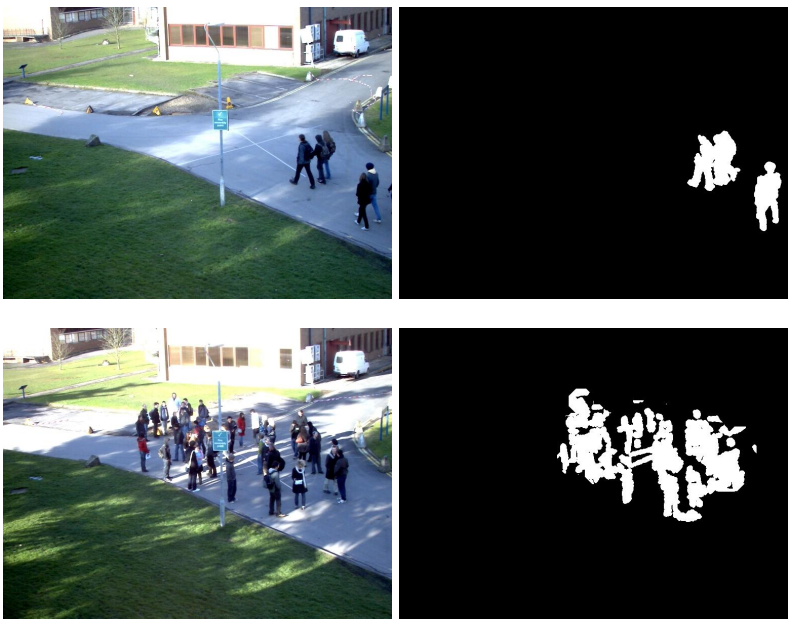


Fig. 2.5: Algunos ejemplos de los objetos detectados por el algoritmo propuesto.

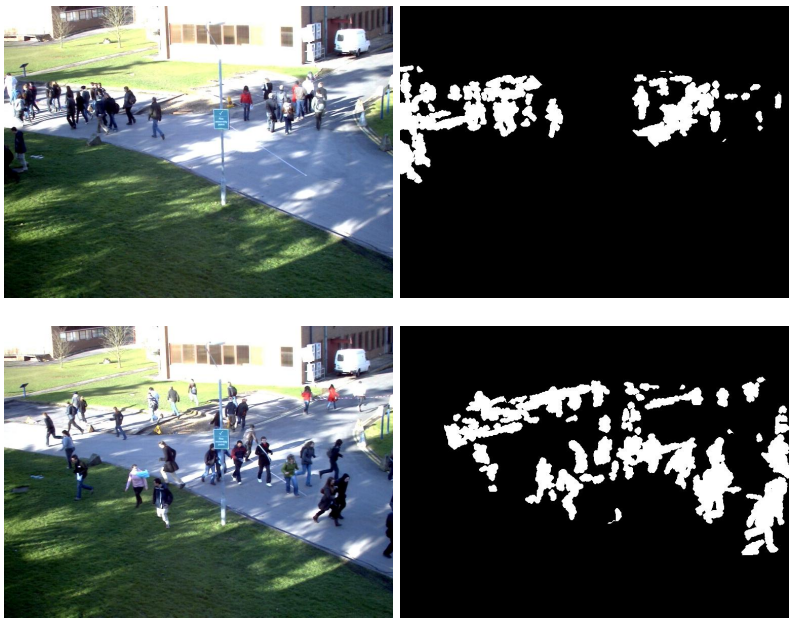


Fig. 2.6: Algunos ejemplos de los objetos detectados por el algoritmo propuesto.

3

Tracking

3.1. Introducción y objetivos

En un sistema como el que se propone, resulta necesario obtener el movimiento de los objetos detectados en la etapa anterior para, posteriormente, poder reconocer los diferentes eventos. Para ello, se ha optado por implementar una solución propuesta en el artículo [25], basada en la detección y seguimiento de puntos de interés a lo largo de las secuencias de vídeo.

Esta solución presenta varias ventajas como son el proporcionar buenos resultados cuando existe poco movimiento, y poder ser aplicada únicamente sobre los objetos detectados. Estas ventajas son la consecuencia de que haya sido elegida frente a otras soluciones, como pueden ser la estimación de movimiento tradicional o el cálculo de un mapa de vectores con técnicas de flujo óptico diferencial.

Esta solución se ha implementado sin haber realizado pruebas alternativas, dado que su rendimiento resultó excelente desde el principio, dedicando, por tanto, más tiempo a la etapa de reconocimiento de eventos. El sistema implementado se puede observar en el diagrama de bloques de la figura 3.1.

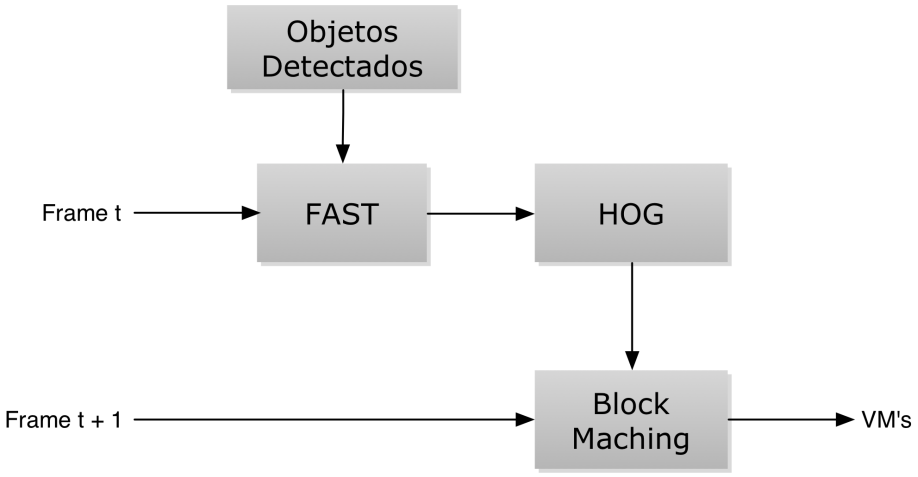


Fig. 3.1: Diagrama de bloques del sistema de *tracking*.

Los pasos en los que se divide el sistema son los siguientes:

1. Para cada plano se extraen unos puntos característicos utilizando el algoritmo FAST (*Features from Accelerated Segment Test* [57] [58] y [59]) .

2. Se eligen únicamente los puntos que pertenecen a los objetos detectados en la etapa anterior.

3. Cada uno de estos puntos se describe usando el algoritmo HOG (*Histograms of Oriented Gradients* [17] y [16]).

4. Se buscan los puntos en el plano siguiente, utilizando la técnica de *Block Matching*, dando lugar un vector de movimiento para cada punto.

En las siguientes secciones se describe detalladamente cada uno de estos pasos.

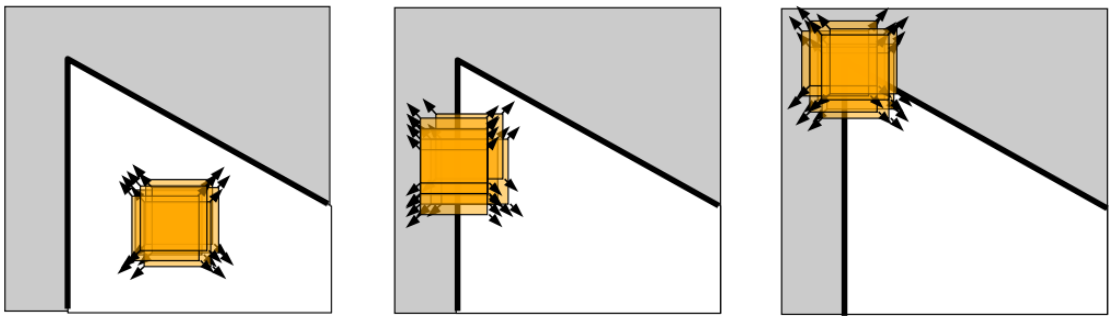
3.2. Extracción de puntos caracteristicos

Para poder obtener los vectores de movimiento es necesario, en primer lugar, extraer puntos que sean lo suficientemente característicos para ser buscados en los planos consecutivos. Estos puntos característicos (*Salients points, Interest points*), deben cumplir lo siguiente:

- Distintivos en la imagen.
- Invariantes frente a los cambios de vista, condiciones de iluminación, deformaciones y oclusiones parciales de los objetos.
- Estables de una imagen a otra en la secuencia.
- En todas las direcciones del entorno del punto deben existir grandes variaciones de la intensidad.

En general, se buscan variaciones grandes de la señal en dos dimensiones, como pueden ser esquinas, curvaturas, destellos, etc.

Una primera aproximación puede consistir en buscar los puntos donde convergen los bordes de la imagen (esquinas). En estos puntos, al desplazar una ventana en cualquier dirección, se origina un gran cambio en la intensidad como se puede observar en la imagen 3.2.



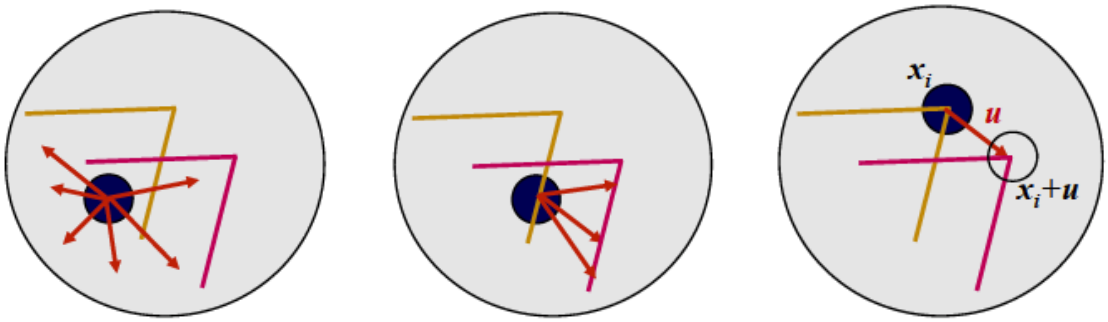
(a) Plana: ningún cambio en todas las direcciones

(b) Borde: ningún cambio a lo largo de la dirección del borde

(c) Esquina: cambio significativo en todas las direcciones

Fig. 3.2: Desplazamiento de una ventana sobre diferentes regiones

La clara conclusión es que, al extraer puntos de este tipo en cada plano e intentar buscarlos en la imagen siguiente, se puede lograr una buena correspondencia, obteniendo el vector de movimiento correcto (ver figura 3.3). Es deseable que cada punto extraído, al ser buscado en la siguiente imagen en una región alrededor del mismo, sea lo suficientemente característico para no confundirlo con otro de similares características.



(a) Region Plana: múltiples vectores posibles en todas direcciones.

(b) Borde: múltiples vectores posibles a lo largo del borde.

(c) Esquina: sólo un vector posible.

Fig. 3.3: Posibles vectores de movimiento del punto entre el plano donde es extraído (azul) y el siguiente (rojo). Imagen tomada de [2].

Utilizando la máscara obtenida en la etapa anterior se pueden detectar solamente los puntos que pertenecen a regiones que no son parte del fondo. Estos puntos se buscan en planos sucesivos, proporcionando información sobre el movimiento de las personas.

Existen muchos detectores en la literatura basados en localizar este tipo de puntos característicos en la imagen, como pueden ser: HARRIS [28], SIFT [45],SUSAN [61] ,etc.

En este proyecto se eligió el detector FAST *Features from Accelerated Segment Test* ([57] [58] [59]). FAST es un detector que se caracteriza por buscar las esquinas existentes en la imagen, siendo el algoritmo de detección más rápido en la literatura. Dado que se trata de una aplicación del ámbito de la vídeo vigilancia, donde es necesario que el sistema opere en tiempo real, es el que mejor se adapta a nuestras necesidades.

3.2.1. FAST

El detector FAST *Features from Accelerated Segment Test* ([57] [58] y [59]) se basa en el test del segmento. Considerando un círculo de dieciséis píxeles alrededor del píxel p candidato a considerarse como *corner* (como se observa en la figura 3.4), el detector FAST original [57] clasifica p como *corner* si existe un conjunto de n píxeles contiguos en el círculo, cuya intensidad sea para todos ellos mayor que la intensidad del píxel candidato I_p más un umbral t, o si todos ellos tienen una intensidad menor a $I_p - t$. Fijando $n = 12$, permite realizar una clasificación muy rápida de los píxeles que no son *corners*. Comprobando únicamente los píxeles 1, 5, 9 y 13 (las cuatro direcciones cardinales) podemos decidir si no es *corner*. Si p es *corner* al menos la intensidad de tres de estos píxeles deberá ser mayor que $I_p + t$ o menor que $I_p - p$, si esto no se cumple, p no puede ser un *corner*. Este test es lo que los autores denominan *Accelerated Segment Test*.

Con este detector se obtiene un alto rendimiento aunque presenta una serie de inconvenientes:

1. El test de alta velocidad no generaliza bien para $n < 12$.

2. La elección del orden en que se comprueban los píxeles afecta a la eficiencia del detector, debido a que esta no es óptima.

3. Muchos casos son detectados unos cerca de otros

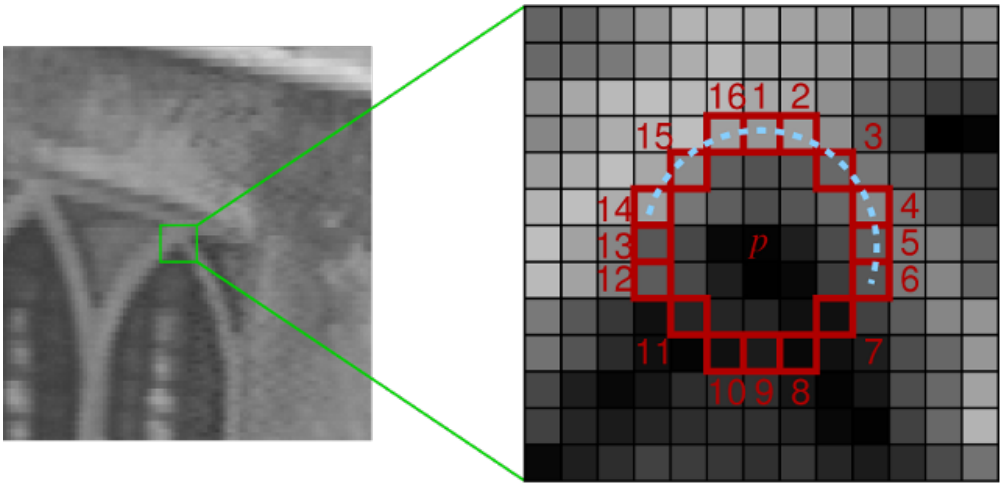


Fig. 3.4: Test del segmento para $n = 9$. Los píxeles en rojo son usados para la detección. El píxel p es el candidato, el arco marcado en azul indica que 9 píxeles contiguos tienen mayor intensidad que p, más un umbral. Foto tomada de [57].

Utilizando aprendizaje máquina en FAST

Los autores proponen basarse en aprendizaje máquina en [58] y [59], para solucionar los dos primeros problemas. El proceso funciona en dos pasos.

En primer lugar, de un conjunto de imágenes del ámbito de la aplicación en la que se será usado el detector final, se extraerán todos los posibles círculos de 16 píxeles. Estos son etiquetados como *corners* utilizando la implementación anteriormente descrita para el n dado y un umbral t correctamente elegido.

Para cada localización en el círculo $x \in \{1, ..., 16\}$, el píxel en esta localización relativa a p, denotado como $p \to x$, puede presentar uno de los siguiente estados:

$$S_{p\to x} = \begin{cases} d, & \text{si} \quad I_{p\to x} \leq I_p - t & \text{(darker)} \\ s, & \text{si} \quad I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & \text{si} \quad I_p + t \leq I_{p\to x} & \text{(brighter)} \end{cases} \tag{3.1}$$

Siendo P el conjunto de todos los píxeles en el conjunto de imágenes de entrenamiento, si se elige un x se puede dividir P en tres subconjuntos, P_d, P_s y P_b, donde:

$$P_b = \{p \in P : S_{p\to x} = b\} \tag{3.2}$$
$$P_d = \{p \in P : S_{p\to x} = d\} \tag{3.3}$$
$$P_s = \{p \in P : S_{p\to x} = s\} \tag{3.4}$$

Una elección de x divide el conjunto P en tres. El conjunto P_d contiene todos los puntos donde x es más oscuro que el píxel central p, más un umbral t. P_b contiene los puntos donde x tiene una intensidad mayor p más un umbral t y P_s contiene el resto de puntos donde x tiene una intensidad similar.

En segundo lugar para recorrer el árbol, se emplea el algoritmo ID3 propuesto en [56], que comienza eligiendo el x que maximiza la información sobre si el píxel candidato es un *corner* o no. Definiendo K_p como una variable booleana que es verdadera si p es *corner* y falsa en caso contrario. Se puede medir la entropía de K_p en un conjunto arbitrario de *corners* Q definida como:

$$H(P) = (c + \bar{c}) \log_2(c + \bar{c}) - c \log_2 c - \bar{c} \log_2 \bar{c} \tag{3.5}$$

donde $c = |\{p|K_p \text{ is true}\}|$ (número de esquinas) y $\bar{c} = |\{p|K_p \text{ is false}\}|$. Se ha de elegir el x que maximice la ganancia de información:

$$H_g = H(P) - H(P_d) - H(P_s) - H(P_b) \tag{3.6}$$

Una vez se ha elegido el x para el que se obtiene la mayor información, se aplica el mismo proceso recursivamente, en los tres conjuntos que define el x elegido, por ejemplo en x_b se selecciona para partir P_b en $P_{b,d}$, $P_{b,s}$ y $P_{b,b}$, x_s para partir P_s en $P_{s,d}$, $P_{s,s}$ y $P_{s,b}$, etc. El proceso termina cuando la entropía de un subconjunto es cero.

Esto produce un árbol de decisión capaz de clasificar correctamente todos los *corners* del conjunto de entrenamiento siguiendo las reglas del algoritmo FAST. Este árbol de decisión se convierte a código C, creando una serie de bucles *else-if* que es usado como detector. Este proceso hace que el algoritmo funcione muy rápido y pueda utilizarse en tiempo real.

Supresión de puntos

Para solucionar el problema de detectar muchos puntos cercanos entre sí, se calcula una función de calidad V para cada punto. Los puntos cercanos a otro con mayor V serán eliminados. Existen muchas maneras de definir esta función como:

1. El máximo valor del número de puntos diferentes n para el cual p sigue siendo detectado.
2. El máximo valor del umbral t para el cual p sigue siendo detectado.
3. La suma de diferencias absolutas de la intensidad de los píxeles de la region circular donde se hace el test del segmento y el píxel p.

Las definiciones 1 y 2 pueden ocasionar problemas dado que muchos puntos pueden presentar valores similares. Los autores eligieron una versión modificada de 3, computacionalmente más eficiente, definida como:

$$V = \text{máx}\left(\sum_{x\in S_{bright}} |I_{p\to x} - I_p| - t, \sum_{x\in S_{dark}} |I_p - I_{p\to x}| - t\right) \tag{3.7}$$

donde: (3.8)

$$S_{bright} = \{x | I_{p\to x} \geq I_p + t\} \tag{3.9}$$

$$S_{dark} = \{x | I_{p\to x} \leq I_p + t\} \tag{3.10}$$

A pesar de esto, el algoritmo detecta demasiados puntos. Para que el *tracking* sea computacionalmente eficiente y pueda ejecutarse en tiempo real, en esta aplicación se usan como máximo 80 puntos.

Algunos resultados

En la figura 3.5 se observan los puntos detectados en la imagen y algunos parches de 16×16 píxeles alrededor de estos puntos.

(a) Puntos extraidos por FAST (b) Algunos parches

Fig. 3.5: Puntos extraídos por el algoritmo FAST y algunas regiones alrededor de estos.

3.3. Descriptor de puntos: HOG

Se necesita un descriptor que defina cada punto para, posteriormente, poder compararlo con otros en el siguiente plano y así poder elegir el más parecido. Para esto se ha usado una versión simplificada del algoritmo HOG (*Histograms of Oriented Gradients* [17] [16]), en el que se define un parche o región alrededor del punto detectado (se hicieron pruebas y se optó por un tamaño de 16×16 píxeles), y se describe basándose en su textura. El algoritmo se resume en los siguientes pasos:

Algorithm 1 Resumen del algoritmo HOG.

Require: Parche del que se quiere extraer el descriptor.
Ensure: Descriptor
Cálculo de gradientes y orientaciones de los bordes.
1: Realizar la convolución del parche con las máscaras $[-1, 0, 1]$ y $[-1, 0, 1]^T$ para obtener dx y dy.
2: Calcular el gradiente como $r = dx^2 + dy^2$.
3: Calcular la orientación como $\theta = \text{arc tg}\,(\frac{dy}{dx})$
Cálculo del histograma de orientaciones.
4: Dividir el parche en celdas. Se dividió en 3×3 celdas, esta division se aprecia en la figura 3.6.
5: Calcular en cada celda el histograma de orientaciones de 9 bins $b = 9$, sin tener en cuenta el signo de las orientaciones, es decir, dividiendo el intervalo de 0 a 180 grados en 9 bins.
6: La contribución de cada píxel al histograma es la magnitud del gradiente r.
Normalización y cálculo del descriptor final.
7: Normalizar el histograma de cada celda, dividiendo este entre el histograma del parche completo.
8: Recolectar los histogramas de todas las celdas, obteniendo un vector de tamaño $3 \times 3 \times 9$.

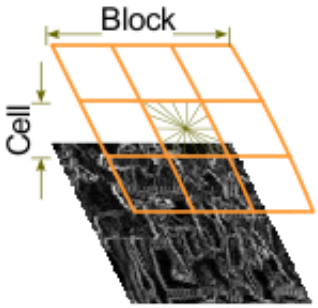


Fig. 3.6: División de un parche en 3×3 celdas. Imagen tomada de [17].

3.4. Block Matching

A cada punto detectado en la posición (i, j) se le asigna una región alrededor (16×16 píxeles) denotada $R(i, j)$, de ésta se calcula el descriptor HOG, tal y como se describió en la sección anterior. En el siguiente plano se define un área de búsqueda SR (*Search Range*) en el que se aplica la técnica de *Block Matching*, que consiste en localizar el punto del área de búsqueda que proporcione una región lo más parecida posible a la original. Dicho bloque proporciona el vector de movimiento del punto. Este procedimiento se puede observar en la figura 3.7.

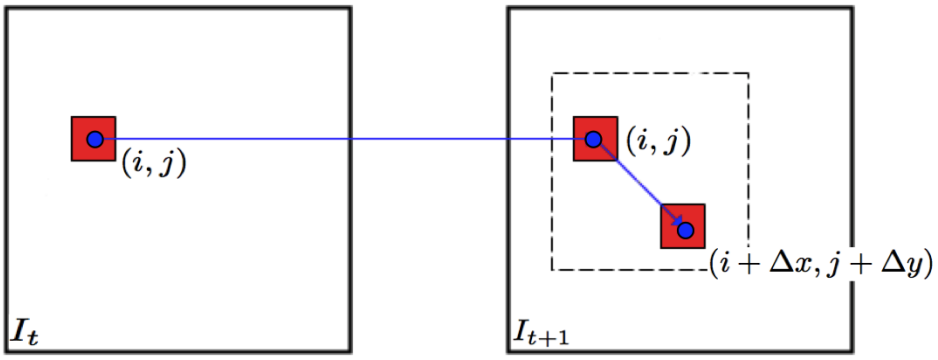


Fig. 3.7: Block Matching. Se muestra un plano y el siguiente, en el que se realiza la búsqueda del bloque de la posición (i, j) obteniendo el vector de movimiento $(\Delta x, \Delta y)$.

Por tanto, se realiza la búsqueda del bloque que dentro del área de búsqueda minimice la distancia con el bloque original:

$$\vec{v} = (v_x, v_y) = (\Delta x, \Delta y) = \arg \min_{(\Delta x, \Delta y)} \{\text{dist}(\Delta x, \Delta y)\} \tag{3.11}$$

Donde la distancia, para el caso de descriptores HOG, se puede calcular a partir de la intersección entre el histograma del bloque original $h_t(i, j)$ y el del bloque tomado de la región de búsqueda $h_{t+1}(i + \Delta x, j + \Delta y)$, definida como:

$$\text{dist}(\Delta x, \Delta y) = 1 - IH(h_t(i, j), h_{t+1}(i + \Delta x, j + \Delta y)) \tag{3.12}$$

La intersección de histogramas $IH(h_1, h_2)$ se define como:

$$IH(h_1, h_2) = \sum_{i=1}^{N} \text{mín}\{h_1(i), h_2(i)\} \tag{3.13}$$

donde N es la longitud del descriptor.

$\Delta x, \Delta y$ es la variación de la posición del bloque tomado del área de búsqueda con respecto al bloque original, que cumple: $0 \leq \Delta x, \Delta y \leq SR$ y proporciona el vector de movimiento buscado.

3.5. Resultados

Evaluar de forma cuantitativa este apartado resulta difícil, dado que realizar una anotación del movimiento de los puntos detectados resulta una tarea inabordable. Por ello, en este punto se optó por realizar una evaluación cualitativa realizando unos vídeos en los que se observa un plano y el siguiente, con los puntos detectados y los puntos resultantes del *tracking*, unidos por líneas. Una captura de estos vídeos puede verse en las figuras 3.9 y 3.8.

Al moverse las personas aproximadamente en la misma dirección las líneas deben ser lo más paralelas posibles, lo que indica que se ha obtenido un buen resultado. Observando detenidamente las figuras 3.9 y 3.8 se aprecia como la mayoría de los puntos han sido encontrados correctamente.

Estos vídeos se pueden encontrar en `http://www.tsc.uc3m.es/~fsilos/tfg/tfg.html`.

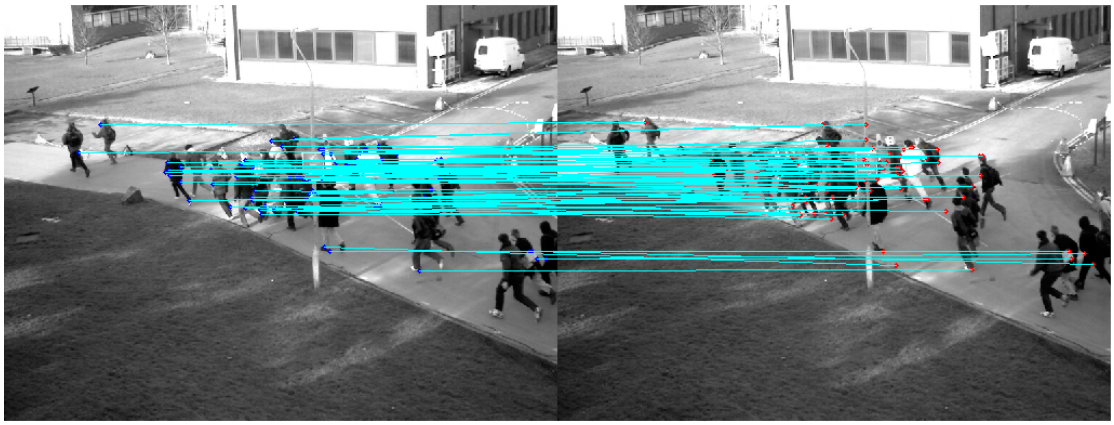


Fig. 3.8: Resultados tracking. Vídeo donde se observa un plano y el siguiente, con los puntos detectados y los puntos resultantes del *tracking*, unidos por líneas horizontales.

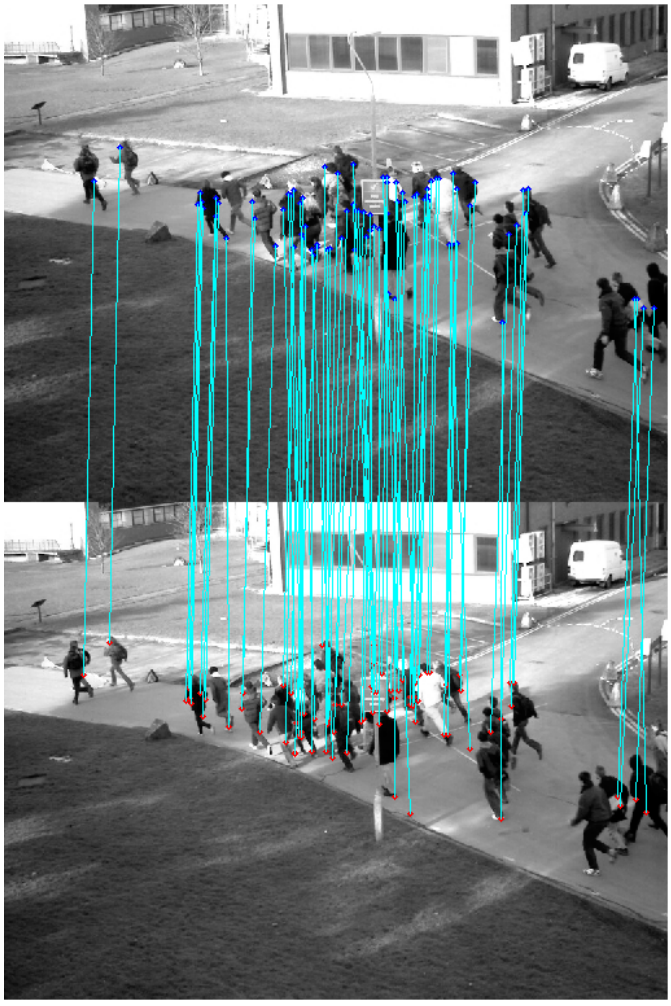


Fig. 3.9: Resultados tracking. Vídeo donde se observa un plano y el siguiente, con los puntos detectados y los puntos resultantes del *tracking*, unidos por líneas verticales.

4

Reconocimiento de Eventos

4.1. Introducción y objetivos

En este capítulo se describe cómo se reconocen los diferentes eventos que se dan en la base de datos PETS 2010. Estos eventos son:

- *Walking:* Representa a un número significativo de personas desplazándose lentamente.
- *Running:* Representa a un número significativo de personas desplazándose rápidamente.
- *Evacuation:* Representa la dispersión rápida en diferentes direcciones de una multitud.
- *Crowd Formation:* Representa la unión en un grupo de un gran número de individuos provenientes de diferentes direcciones.
- *Crowd Splitting:* Representa la división de un grupo de individuos, en dos o más grupos que toman diferentes direcciones.
- *Local Dispersion:* Representa la dispersión de un pequeño grupo de individuos de una multitud.

En primer lugar, se describirá el estado del arte de los métodos de reconocimiento de eventos en los que se ven involucradas un gran número de personas (*Crowd event recognition*), para pasar a presentar la solución adoptada para reconocer los distintos eventos a partir de los datos proporcionados por las dos etapas anteriores.

Se ha dedicado un especial interés a la parametrización de los distintos eventos con características innovadoras que no están en la literatura, así como al estudio sistemático de esta parametrización mediante criterios de información mutua. Otra contribución importante ha sido la incorporación de máquinas de vectores soporte para las tareas de clasificación.

4.2. Estado del arte

El reconocimiento de eventos inusuales en vídeo vigilancia ha sido tratado desde dos enfoques:

- Detección de eventos explícita (*Explicit event detection*): El problema se modela como una serie predefinida de eventos de interés que serán detectados en la operación normal del sistema.
- Métodos de desviación (*Deviation methods*): Considera eventos inusuales a aquellos que difieren considerablemente de la actividad considerada normal.

La división anterior no es debida únicamente a la tecnología, sino también a la naturaleza de los contenidos audiovisuales para cada aplicación.

4.2.1. Explicit Event Detection

La detección de eventos explícita (*explicit event detection*) asume que en la escena no ocurren demasiadas oclusiones, con el fin de detectar casos de seres humanos [17] [68] y modelar su comportamiento. Este enfoque puede encontrarse en trabajos recientes y sistemas presentados a competiciones internacionales como TRECVID [49], para la detección de eventos que únicamente involucran a una sóla persona (por ejemplo: *running, pointing, leaving/abandoning a baggage, using a cellphone*), pero también eventos que involucren a varias personas o grupos (*embracing, meeting, splitting from a group*).

En la detección de eventos donde se ven involucradas multitudes se sustituye el concepto de persona por el de objeto, el cual puede comprender a una o varias personas que formen un grupo.

Una vez que los seres humanos u objetos han sido detectados, estos pueden ser descritos con características robustas, como pueden ser HOG [17] o histogramas de color HSV [66], y además pueden ser seguidos (*tracked*) por métodos conocidos como *mean-shift* [13] , filtros de partículas [3] [47] o realizando un seguimiento basado en puntos de interés [65]. Para el caso particular del análisis de vídeo, las características locales espacio-temporales como MoSIFT [11] son de gran interés, ya que proporcionan representaciones compactas pero descriptivas para el análisis del movimiento.

La trayectoria de cada objeto en la escena se codifica como un vector de longitud fija, para poder utilizar técnicas de aprendizaje máquina (SVMs en [41] [36]); o como una secuencia de datos que se introducen en modelos probabilísticos estructurales como son HMMs [35] o SVM-HMM [26] [67] . Debido a la alta dimensionalidad de los datos resultantes, modelos como *Bag-of-features* son de gran ayuda, de manera que cada descriptor está asociado al índice de una palabra en un vocabulario espacio-temporal para, posteriormente, calcular histogramas de ocurrencias de palabras y clasificar el evento en varias regiones espaciales y temporales [40].

Debido a las particulares características de cada evento, las soluciones discriminativas tienden a ser el paradigma más típico en este campo. En muchos casos soluciones *ad-hoc* son aplicadas para mejorar los resultados, como en [27], donde los autores hacen uso de *Motion Edge Histogram Images* para detectar personas abrazadas o juntas, y en [24], donde los autores realizan una doble estimación del fondo (rápida y lenta) para detectar equipaje abandonado.

Por otra parte, eventos específicos prioritarios pueden apuntar a favor de la parte de la escena donde es más probable que el evento ocurra, como en [4], donde se calculan mapas de calor para cada evento y luego se consideran características independientes a las procedentes de las regiones calientes y frías. Esta técnica permite discriminar información proveniente de las áreas de interés y del fondo.

4.2.2. Deviation Methods

Los métodos de desviación, por el contrario, asumen que la detección y seguimiento de casos individuales no son factibles dada la alta densidad de personas en la escena, que dan lugar a espacios desordenados y con oclusiones. Además, esta situación impide la estimación de las trayectorias de los objetos, por lo que para este caso son necesarios enfoques más generales. Sin embargo, dado que muchos acontecimientos tienden a ocurrir en lugares determinados de la escena, un análisis global de la secuencia puede no ser adecuado. Es por ello que la literatura propone la detección espacio-temporal de las regiones de interés en la secuencia. Estas regiones de interés son localizadas alrededor de puntos de interés (*salients points*) [19] [42] o construidos mediante un mallado espacial a lo largo de los segmentos de la secuencia [22] [32]. El movimiento asociado a cada región se codifica usando métodos estadísticos tales como distribuciones Gausianas [22] [39], que son capaces de modelar la variabilidad del movimiento dentro de una región al no estar asociadas a objetos particulares.

La complejidad computacional de estos métodos se suele reducir mediante técnicas de sustracción del fondo. Esto solamente se puede aplicar en el caso de que la cámara este lo suficientemente estática para ser capaz de seleccionar las áreas de interés. Estas técnicas se basan en modelos probabilísticos como mezclas de gausianas (GMM) [63].

Algunos sistemas aplican el problema de detección de situaciones anómalas como en [46].

Una vez que las regiones espacio-temporales han sido descritas, se suelen utilizar métodos no supervisados de *clustering* para generar modelos de las situaciones "usuales y normales" en el vídeo. Para esto, los algoritmos hacen uso de la similitud entre medidas, para comparar los distintos casos y decidir cuándo ocurre una situación anómala. Ejemplos de estas medidas son *Symmetric KL Divergence* [69] (en caso de que la distribuciones de probabilidad se estén usando para modelar el movimiento) y varias medidas *ad-hoc* incluyendo estadísticas de movimiento [32] [25].

4.3. Solución propuesta

Dado que nuestro sistema va a modelar una serie predefinida de eventos de interés que serán detectados en la operación normal del sistema, éste se puede encuadrar en la literatura del grupo de los sistemas de tipo *explicit event detection* aunque, como se vera posteriormente, también se ha implementado un sencillo sistema de reconocimiento de eventos anómalos. En nuestro caso, al tratarse de vídeos con multitudes, en ningún caso se desea la detección individual de personas sino de objetos que pueden englobar uno o más individuos.

El sistema tiene que reconocer en cada plano de vídeo la ocurrencia de cada uno de los eventos. Para ello se dispone de la siguiente información:

- Máscaras que proporciona el algoritmo de extracción de fondo, donde están presentes todos los objetos detectados.
- Puntos característicos extraídos por FAST.
- Vectores de movimento asociados a cada punto.

Con esta información se pueden extraer una serie de características que permiten parametrizar cada evento a detectar.

El diagrama de la solución implementada se puede observar en la figura 4.1.

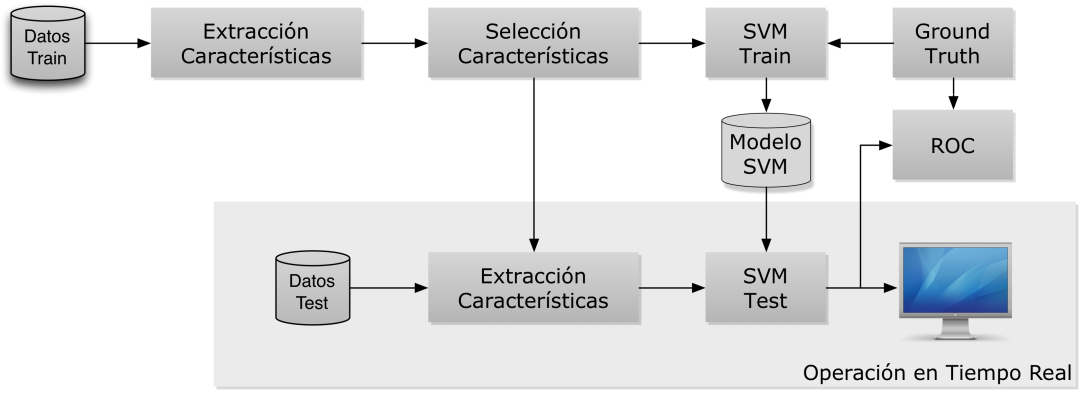


Fig. 4.1: Diagrama del sistema de reconocimiento de eventos.

El modelo propuesto consta de un detector binario para cada evento. Cada uno de estos detectores tiene características diferentes adaptadas al evento que se quiere detectar. La metodología para el desarrollo del sistema consta de los siguientes pasos:

1. División de la base de datos en dos conjuntos: entrenamiento y test. El 40 % de los planos estará disponible para entrenamiento y el resto para test.
2. Extracción de las diferentes características en cada plano.

3. Selección mediante criterios de información mutua de las características que definan mejor a cada evento.

4. Con las características procedentes de los planos de entrenamiento, se entrenará un clasificador binario para cada evento.

5. Clasificación de los planos de test para posteriormente calcular el rendimiento del sistema, calculando la curva ROC y otras medidas de interés.

Cada uno de los pasos se explica detalladamente en las siguientes secciones.

4.4. Extracción de características

4.4.1. Granularidad

A partir de la información disponible han sido extraídas las características que se detallarán a continuación. Estas características pueden haber sido calculadas de dos maneras:

1. Sobre el plano completo, existiendo una característica por plano y no teniendo en cuenta los distintos objetos detectados.

2. Sobre los diferentes objetos detectados, de forma que se obtenga una característica por cada objeto.

En función del evento que se quiera modelar se seleccionan, como se vera más adelante, unas o otras de estas características. Las características únicamente se asocian con las zonas que no pertenecen al fondo de la imagen.

4.4.2. Velocidad Media

Una buena característica para los eventos que involucran un alto movimiento en las escenas (*running, evacuation*) puede ser el desplazamiento medio de todos los vectores de cada plano. Esta característica se calcula por tanto a nivel de plano.

La velocidad media para el plano n se define como la media de todos los vectores disponibles para este:

$$\bar{v}_n = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{4.1}$$

donde N es el número de vectores.

Con el fin de extraer la variabilidad temporal de la velocidad media, la característica final ha sido definida como la evolución temporal de la velocidad media en una ventana de tiempo de los k últimos planos:

$$\bar{v} = \begin{bmatrix} \bar{v}_n \\ \bar{v}_{n-1} \\ \bar{v}_{n-2} \\ \vdots \\ \bar{v}_{n-k} \end{bmatrix} \tag{4.2}$$

Se realizaron pruebas preliminares donde se obtuvo el mejor rendimiento para una ventana temporal de longitud $k = 5$.

4.4.3. Histograma de orientaciones del movimiento

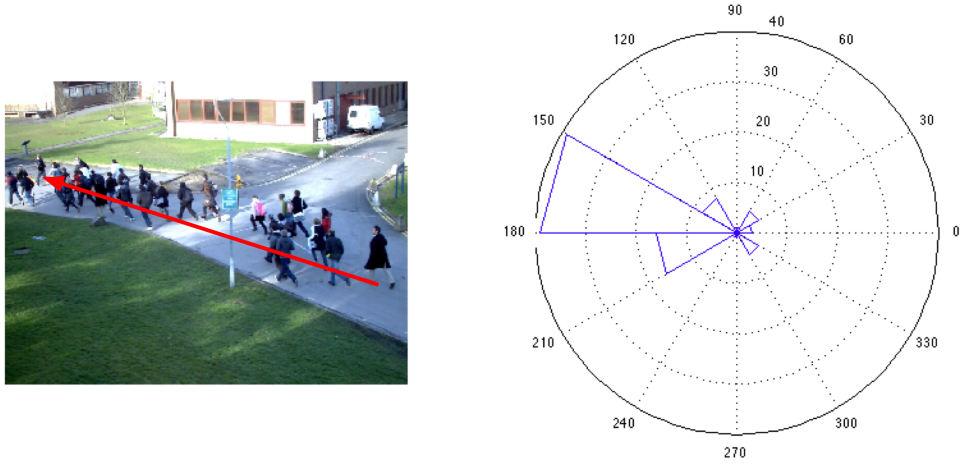
Para definir algunos eventos es importante obtener las direcciones dominantes del movimiento que está ocurriendo en cada plano. Para ello se dispone de los vectores de movimiento que han sido calculados en la etapa de *tracking*. Estos vectores están definidos por dos puntos, el punto

de origen p y el punto final p' encontrado mediante el *tracking* en el plano siguiente. Por tanto, se puede calcular el ángulo que forma este vector con respecto al eje de abscisas como:

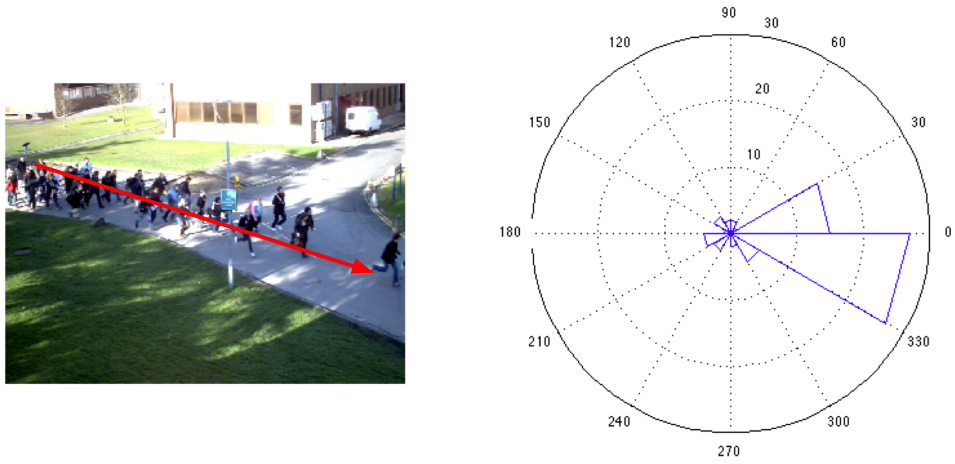
$$\theta = \text{arc tg}\left(\frac{p'_y - p_y}{p'_x - p_x}\right) \tag{4.3}$$

Este cálculo se realiza para todos los vectores de cada plano.

Posteriormente, se calcula el histograma dividiendo el plano cartesiano en b regiones y se cuenta la frecuencia de la orientación de los vectores en cada una de estas regiones. A partir de pruebas preliminares se eligió $b = 12$ como número adecuado de orientaciones para el histograma. En la figura 4.2 se observa como esta característica capta las direcciones principales en las que ocurre el movimiento.



(a)



(b)

Fig. 4.2: Ejemplo de histograma de orientaciones de movimiento en dos planos con diferente movimiento.

Es de esperar que en eventos de tipo formación, evacuación y dispersión local, esta característica sea de gran ayuda dado que estos eventos están altamente relacionados con la dirección del movimiento.

Este histograma se calcula a nivel de objeto, obteniendo uno por objeto detectado. Posteriormente, para la clasificación en la SVM se realiza la media de todos los histogramas, dado que

el número de objetos detectados es variable.

4.4.4. Parámetro de divergencia

Se puede interpretar que, mediante los vectores de movimiento, unos puntos en un plano se transforman en otros en el plano siguiente. Por tanto, el punto de origen p se transforma en p' según el vector de movimiento siendo $p' = p + v$. En el conjunto de puntos de cada plano se puede parametrizar como:

$$P' = \mathbf{H}P \tag{4.4}$$

donde $\mathbf{H}$ es una matriz 3×3 de transformación afin que relaciona las localizaciones de los puntos, P' y P son las matrices que contienen los puntos asociados a cada plano. Se puede expresar por tanto de la siguiente manera:

$$\begin{bmatrix} p'_{x1} & p'_{x2} & \dots & p'_{xn} \\ p'_{y1} & p'_{y2} & \dots & p'_{yn} \\ 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\epsilon\,\mathrm{sen}\,\theta & t_x \\ \epsilon\,\mathrm{sen}\,\theta & \epsilon\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x1} & p_{x2} & \dots & p_{xn} \\ p_{y1} & p_{y2} & \dots & p_{yn} \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

donde n es el número de puntos disponibles para el plano.

A partir de la definición de transformación afín, se puede obtener un parámetro de divergencia que indica cuando los vectores están convergiendo o divergiendo, definido como:

$$div = 2\epsilon\cos\theta \tag{4.5}$$

Este mismo procedimiento se emplea en algunos algoritmos ([33]) que intentan detectar movimientos de cámara, más concretamente cuando ocurren *zooms*.

Este parámetro puede ayudar a definir los eventos de evacuación y formación, dado que como se observa en la figura 4.3, en el evento de evacuación los puntos divergen y en formación tienden a converger en un punto.

(a) Evacuación (b) Formación

Fig. 4.3: El parámetro divergencia modela los eventos de evacuación y formación aprovechandose de la transformación que estos eventos realizan en los puntos de un plano a otro.

Tenemos un sistema sobredeterminado (más ecuaciones que incógnitas) con lo que la matriz P no es invertible. Para calcular H se tienen dos opciones:

1. Se puede buscar la solución H de mínimos cuadrados utilizando la matriz pseudoinversa, realizando el siguiente despeje en la ecuación:

$$P' = \mathbf{H}P \tag{4.6}$$
$$P'P^T = \mathbf{H}PP^T \tag{4.7}$$
$$\mathbf{H} = P'P^T(PP^T)^{-1} = P' \cdot \mathrm{pmv}(P) \tag{4.8}$$

Donde $P^T(PP^T)^{-1}$ es la matriz pseudoinversa de P denotada pmv(P). El inconveniente es que esta solución es poco robusta frente *outliers* (puntos que se mueven de forma diferente al objeto, a causa de una estimación incorrecta, aparición de brillos u otros artefactos).

2. Se puede usar el algoritmo RANSAC (*RANdom SAmple Consensus*) que estima la matriz H, sin tener en cuenta los puntos que son *outliers*. Se puede encontrar información detallada sobre este algoritmo en el apéndice C.

Se probaron ambas soluciones obteniendo mejores resultados usando el algoritmo RANSAC, debido a que es más robusto frente a *outliers*.

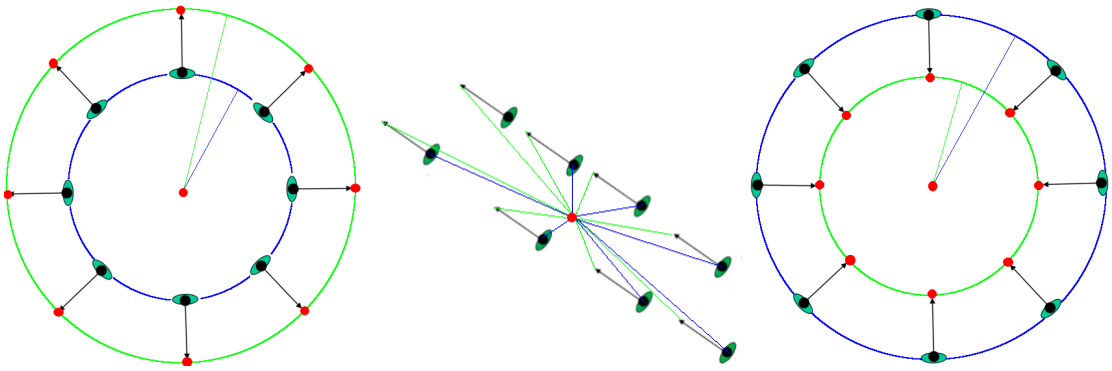
4.4.5. Ratio de distancias

Otra característica que puede definir algunos eventos es el ratio entre la distancia media de los puntos al punto medio en el plano $t-1$ y la distancia media de los puntos al punto medio en el plano t. Este ratio se define como:

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n} d(p_i, \bar{p})}{\frac{1}{n}\sum_{i=1}^{n} d(p'_i, \bar{p})}$$

donde p_i es cada punto original, $p'_i = p_i + v_i$ siendo v_i el vector de movimento asociado a cada punto; y $d(p_i, \bar{p}) = ||p_i - \bar{p}|| = \sqrt{(p_{x_i} - \bar{p}_x)^2 + (p_{y_i} - \bar{p}_y)^2}$

Es de esperar que cuando ocurre un evento de evacuación (figura 4.4 a) este ratio sea menor que uno, al estar alejandose las personas unas de otras. En un evento de formación sin embargo. ocurre lo contrario (figura 4.4 c) y cuando las personas se mueven en la misma dirección este ratio es aproximadamente igual a uno (figura 4.4 b).



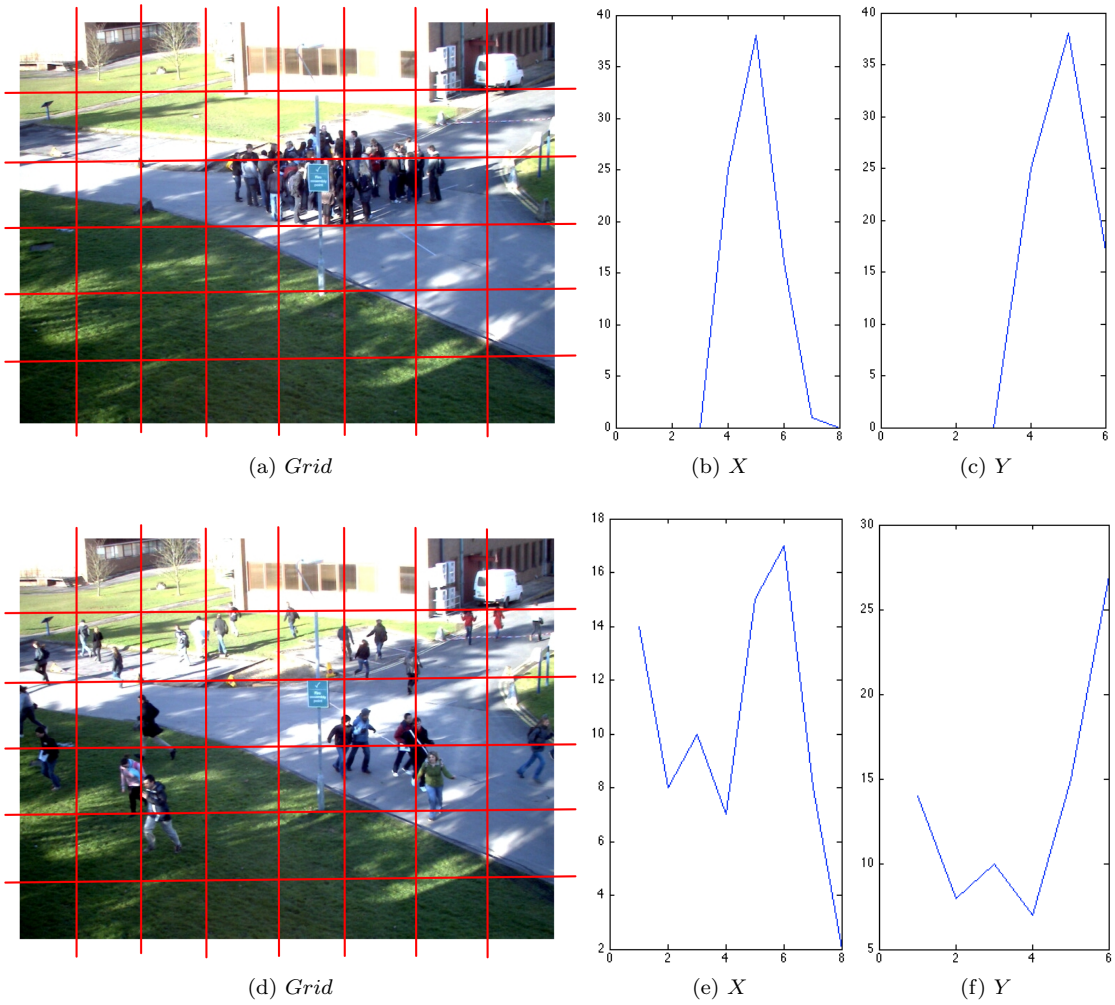
(a) $r < 1$ (b) $r = 1$ (c) $r > 1$

Fig. 4.4: Distintos valores que puede tomar el ratio de distancias. Las líneas azules representan el numerador y las verdes del denominador de la ecuación anterior. Cuando ocurre un evento de evacuación (figura a) este ratio es menor que uno; cuando las personas se mueven en la misma dirección el ratio es aproximadamente igual a uno (figura b) y cuando es un evento de formación el ratio es mañor que uno.

4.4.6. Densidad de puntos

Como última característica se ha definido un *grid* de 8×6 en la imagen (figura 4.5a) donde se cuentan los puntos en el eje X (figura 4.5b) e Y (figura 4.5c), permitiendo al clasificador aprender las zonas de ocurrencia y la distribución espacial de cada evento.

Esta característica es muy dependiente de la cámara dado que modela como las personas se sitúan en el escenario.



(a) *Grid* (b) X (c) Y

(d) *Grid* (e) X (f) Y

Fig. 4.5: Se puede observar dos ejemplos de situaciones diferentes del *grid* y de la variación de puntos en el eje X e Y.

4.5. Selección de las caracterísiticas mediante el estudio de información mutua

Para elegir, de entre todas las características, las que definirán a cada evento, se ha desarrollado un procedimiento basado en criterios de información mutua que mide la información mutua de las distintas características con el vector de etiquetas de cada evento (0 cuando no ocurre 1 en caso contrario).

La información mutua $MI(X;Y)$ entre dos variables aleatorias se define como una cantidad de medida de la dependencia mutua de las dos variables, es decir, la reducción de la entropía de una variable aleatoria X, dado el conocimiento del valor de otra variable aleatoria Y. Matemáticamente puede expresarse como:

$$MI(X;Y) = H(X) - H(X|Y) \tag{4.9}$$

donde H es la entropía.

Se puede observar gráficamente la definición de información mutua en la figura 4.6.

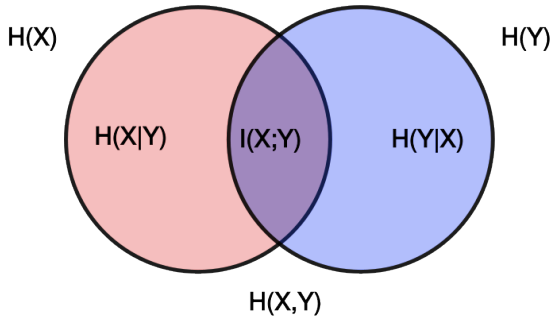


Fig. 4.6: Diagrama de Venn donde se puede observar la representación grafica de la información mutua $I(X;Y)$ entre dos variables aleatorias X e Y.

El procedimiento desarrollado se recoge en el algoritmo 2.

Algorithm 2 Selección de caracteristicas mediante información mutua.

1: Normalización de las características para que estén comprendidas en el intervalo: $[0, 1]$.
2: Calcular la información mutua $MI(X_i; Y)$ entre cada una de las características X_i y el vector de etiquetas del evento Y.
3: Comenzar con el conjunto vacío X^0.
4: **for** $t = 1 \rightarrow t =$ número de características **do**
5: La característica que obtenga la mayor información mutua X_i^* se añade a las restantes $X^t = [X^{t-1}, X_i^*]$.
6: Calcular la información mutua $MI(X^t, X_i; Y)$ entre cada una de las características restantes a las que se le ha añadido el mejor conjunto de las mismas, con el vector de etiquetas de ese evento.
7: **if** No existe un incremento de la informacion mutua. **then**
8: Se eligen como características X_t.
9: **end if**
10: **end for**

En la tabla 4.1 se muestran los resultados proporcionados por el método para cada uno de los eventos.

Este proceso proporciona una guía para la elección, aunque puede darse el caso de que, incluyendo alguna característica no considerada por el algoritmo, se obtenga un mejor resultado en el clasificador. Si existen dudas de este tipo se puede comprobar posteriormente.

Por tanto, las características que se eligen para cada evento son las siguientes:

- **Walking** Se eligen: velocidad media y densidad de puntos.
- **Running** Se eligen: Velocidad media, densidad de puntos y ratio de distancias.
- **Evacuation** En primer lugar se eligieron: velocidad media, densidad de puntos, ratio de distancias. Pero se obtuvieron mejores resultados de clasificación con el parámetro de divergencia, por lo que finalmente fue incluido.
- **Crowd Formation** En primer lugar se eligieron: velocidad media, densidad de puntos. Pero se obtuvieron mejores resultados de clasificación con el parámetro de divergencia, por lo que finalmente fue incluido.
- **Crowd Splitting** Se eligen: velocidad media, histograma orientaciones, densidad de puntos.

- **Local Dispersion** Se eligen: velocidad media, histograma orientaciones, densidad de puntos.

Evento	Característica	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
Walking	Velocidad media	**0.3761**	—	—	—	—
	Densidad de puntos	0.0386	**0.5016**	—	—	—
	Ratio de Distancias	0.0386	0.3667	0.4984	—	—
	Parámetro de divergencia	0.0929	0.3510	0.4966	0.4919	—
	Histograma orientaciones	0.2267	0.3250	0.3813	0.3675	0.3630
Running	Velocidad media	**0.3335**	—	—	—	—
	Densidad de puntos	0.0837	**0.3589**	—	—	—
	Ratio de distancias	0.0837	0.3362	**0.3631**	—	—
	Parámetro de divergencia	0.1522	0.3214	0.3520	0.3510	—
	Histograma orientaciones	0.2258	0.3140	0.3155	0.3073	0.2995
Evacuation	Velocidad media	**0.0925**	—	—	—	—
	Densidad de puntos	0.0525	**0.1338**	—	—	—
	Ratio de distancias	0.0525	0.0813	**0.1398**	—	—
	Parámetro de divergencia	0.0456	0.0821	0.1322	0.1329	—
	Histograma orientaciones	0.0867	0.1006	0.1051	0.0980	0.1013
Crowd Formation	Velocidad media	**0.2860**	—	—	—	—
	Densidad de puntos	0.0903	**0.3911**	—	—	—
	Parámetro de divergencia	0.1497	0.3000	0.3906	—	—
	Ratio de distancias	0.0903	0.2618	0.3903	0.3865	—
	Histograma orientaciones	0.2175	0.2556	0.3308	0.3128	0.3180
Crowd Splitting	Histograma orientaciones	**0.1612**	—	—	—	—
	Densidad de puntos	0.0621	**0.1883**	—	—	—
	Velocidad media	0.1045	0.1849	**0.2329**	—	—
	Parámetro de divergencia	0.0797	0.1642	0.1987	0.2237	—
	Ratio de distancias	0.0621	0.1656	0.1920	0.2189	0.2225
Local Dispersion	Histograma orientaciones	**0.1107**	—	—	—	—
	Densidad de puntos	0.0559	**0.1853**	—	—	—
	Velocidad media	0.1024	0.1186	**0.2037**	—	—
	Parámetro de divergencia	0.0606	0.1159	0.1974	0.1968	—
	Ratio de distancias	0.0559	0.1188	0.1935	0.1967	0.1963

Tabla 4.1: Resultado del proceso de seleción de características. En rojo se marcan los máximos y las características que en sus filas tengan algún paso en rojo son eleguidas para definir el evento.

Para estimar la información mutua se ha utilizado el método descrito en [38] y [64]; el software utilizado se puede encontrar en: `http://www.klab.caltech.edu/~kraskov/MILCA/`. Para este proyecto se usa la función `MIxnyn` que calcula la información entre dos canales de entrada de dimensiones arbitrarias.

4.6. Clasificador

Una vez encontradas las características que definen cada evento se necesita un clasificador que permita predecir cuándo ocurre. Para este proyecto se uso como clasificador las máquinas de vectores soporte, SVM (*Support Vector Machines*), por la simplicidad a la hora de utilizarlas y por su capacidad de generalización con pocas muestras de entrenamiento, como ocurre en este escenario.

El objetivo de las máquinas de vectores soporte es obtener un modelo (basado en los datos de entrenamiento) que pueda predecir la clase de los datos de test. Se etiquetan todos los planos en los que ocurre cada evento y se usan como datos de entrenamiento para cada evento, las características seleccionadas mediante criterios de información mutua. No se usan SVMs multiclase dado que a cada evento lo pueden definir distintas características, necesitando entrenar una SVM binaria diferente para cada evento que se quiera detectar.

Por tanto, se construye un conjunto de entrenamiento de pares de características-etiquetas (x_i, y_i), con $i = 1, ..., l$, donde $x_i \in R^n$ son las características que definen a cada evento, e $y \in \{1, -1\}^l$ es la etiqueta que toma el valor $y_i = 1$ cuando ocurre el evento y $y_i = -1$ en caso contrario.

Las máquinas de vectores soporte ([5], [14]), para encontrar el hiperplano $\mathbf{w}^T\phi(x) + b = 0$ que separa las dos clases, requieren solucionar el siguiente problema de optimización:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \tag{4.10}$$

$$\text{sujeto a:} \quad y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \tag{4.11}$$

Los vectores de entrenamiento x_i son transformados a un espacio de dimensión mayor por la función ϕ. Por defecto, las SVM son clasificadores lineales que maximizan el margen entre la frontera de decisión y las muestras más cercanas pero, eligiendo correctamente una transformación ϕ, se pueden hacer no lineales, permitiendo que las máquinas de vectores soportes encuentren un hiperplano separador entre las clases en este espacio de mayor dimensión (observar figura 4.7).

C es el parámetro de penalización del término de error, cuando este aumenta (ver figura 4.9) se permite un menor error, adaptando el clasificador en mayor medida a los datos de entrenamiento. Este parámetro se elige empíricamente, buscando un compromiso entre la adaptación a los datos de entrenamiento y la capacidad de generalización del decisor.

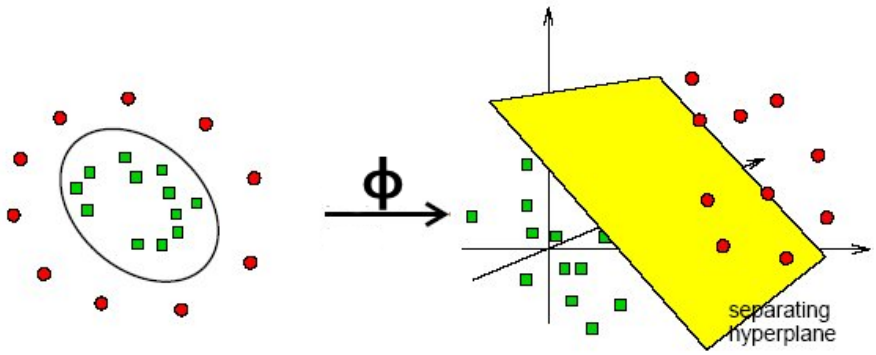


Fig. 4.7: A través de la función ϕ se logra la tranformación de los datos a un espacio de mayor dimensión, que permite encontrar el plano separador de las clases de forma más sencilla.

La función de *kernel* encargada de trasformar los datos a un espacio de mayor dimensión queda definida como: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Las más comunes son las siguientes:

1. Lineal: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$
2. Polinómica: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma\mathbf{x}_i^T\mathbf{x}_j + r)^d$, $\gamma > 0$
3. Radial basis function(RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \mid\mid \mathbf{x}_i - \mathbf{x}_j \mid\mid^2)$, $\gamma > 0$
4. Sigmoide: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i^T\mathbf{x}_j + r)$

Donde γ,r yd son parámetros del *kernel*

Para obtener mejores resultados se ha seguido el procedimiento propuesto en [30], que usa la función de *kernel* RBF y realiza una validación cruzada de los parámetros C y γ. Consta de los siguientes pasos:

Algorithm 3 Proceso para obtener los mejores parámetros en una SVM.

1: Normalización los datos linealmente, para que cada característica este comprendida en el intervalo $[-1, 1]$.
2: Considerar la función de kernel RBF.
3: Realizar una validación cruzada para encontrar los parámetros C y γ.
4: Utilizar los mejores C y γ para entrenar en el conjunto de entrenamiento completo.
5: Test.

En la figura 4.8 se observa cómo la correcta elección de estos parámetos, logra aumentar notablemente el rendimiento del clasificador.

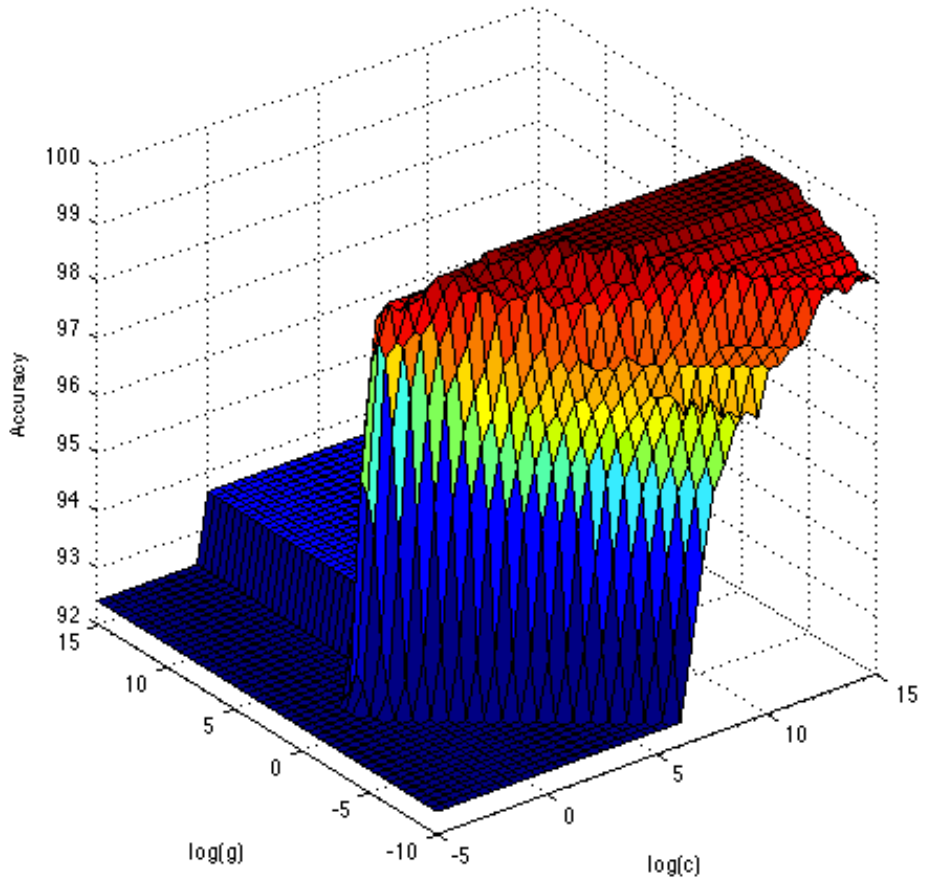
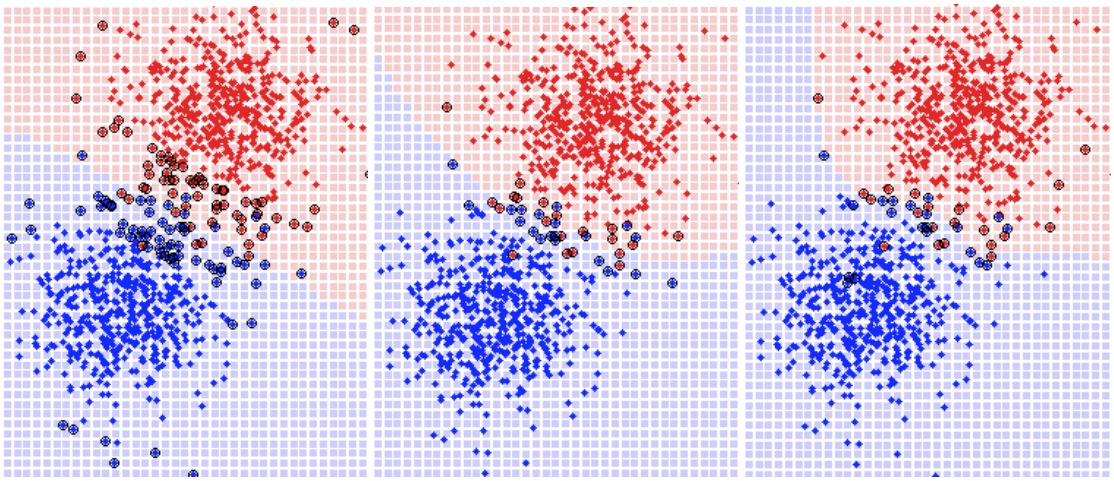


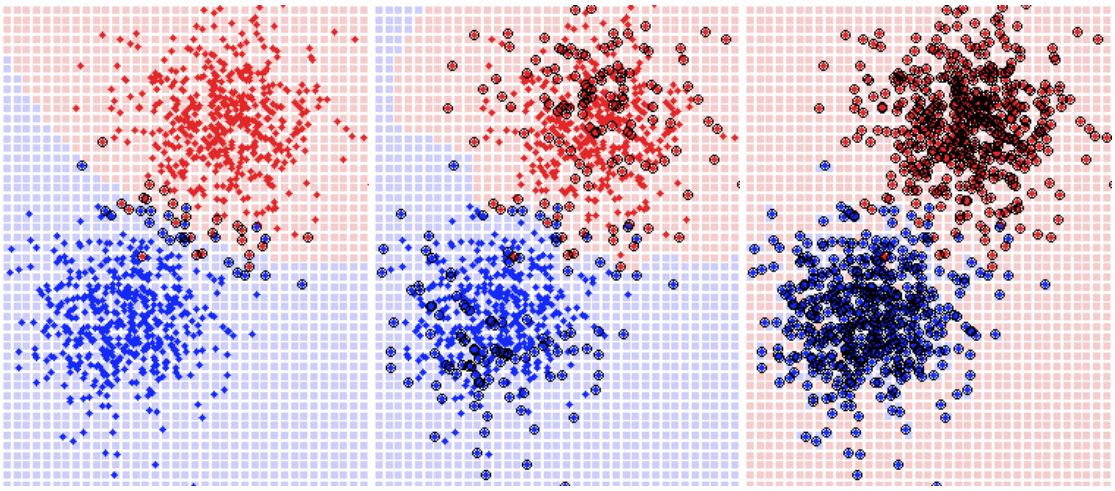
Fig. 4.8: Variación *Accuracy* en función de C y γ, se observa como una mala elección de estos, da lugar a una bajada del rendimiento del clasificador.

La importancia de elegir correctamente C y γ se observa en las figuras 4.9 y 4.10. C modela el margen de error permitido al colocar el plano separador en el conjunto de entrenamiento. Una mala elección de C puede provocar que el clasificador se sobreajuste a los datos de entrenamiento. El parámetro γ modela el tamaño de las Gaussianas de la función de *kernel* RBF.



(a) $C = 0{,}1$ (b) $C = 10$ (c) $C = 1000$

Fig. 4.9: Influencia de C dejando γ fijo en su valor óptimo. Los puntos son las muestras de entrenamiento de cada clase, estando en rojo o azul marcada la región que el clasificador asigna a cada clase. Cuanto mayor es C se permite un menor error al colocar el plano separador, aumentando la complejidad de la frontera, adaptándose cada vez más a los datos de entrenamiento.



(a) $\gamma = 0{,}1$ (b) $\gamma = 10$ (c) $\gamma = 1000$

Fig. 4.10: Influencia γ dejando C fijo en su valor óptimo. Los puntos son las muestras de entrenamiento de cada clase, estando en rojo o azul marcada la región que el clasificador asigna a cada clase. Cuanto mayor es γ, la varianza de las Gaussianas es menor.

Para este proyecto se ha utilizado la librería `LibSVM` [10] como máquina de vectores soporte. Se puede encontrar para su descarga en: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

5

Resultados

5.1. Introducción

En este capítulo se presentan los resultados obtenidos por el sistema propuesto para la detección de eventos en la base de datos PETS 2010 (para más información consultar el apéndice A).

En la sección 5.2 se describen los resultados en base a la matriz de confusión, en 5.3 se pueden observar las curvas ROC que se obtienen para los diferentes eventos, en 5.4 se comparan los resultados obtenidos con otros artículos del estado del arte que utilizan la misma base de datos, y en 5.5 se describen cualitativamente los resultados sobre los vídeos de la base de datos empleada.

5.2. Matriz de confusión

Para cuantificar el rendimiento del sistema propuesto se pueden contabilizar los planos donde éste acierta o falla, utilizando las siguientes definiciones ([21]):

- **TP:** *True Positives*. Eventos correctamente detectados.
- **FN:** *False Negatives*. Eventos no detectados.
- **FP:** *False Positives*. Eventos que el sistema detecta pero que no son correctos.
- **TN:** *True Negatives*. No eventos que son correctamente rechazados.

La figura 5.1 muestra la matriz de confusión que contiene estos números, donde la suma de la diagonal principal representa las decisiones correctas y la suma de la otra diagonal representa los errores (la confusión) entre ambas clases.

Estos números se pueden convertir en ratios ([21]), definidos como:

- **TPR:** *True Positive Rate*, La probabilidad de que el sistema identifique un evento cuando está ocurriendo. También se denomina $P_{\text{Detección}}$ o *recall*.

$$TPR = \frac{tp}{tp + fn} \tag{5.1}$$

- **FPR:** *False Positive Rate*, La probabilidad de que cuando un evento no está ocurriendo el sistema lo identifique como positivo. También se denomina $P_{\text{Falsa Alarma}}$.

$$FPR = \frac{fp}{tn + fp} \tag{5.2}$$

- **ACC:** *Accuracy*, Es una medida que indica como de bien funciona el sistema, es decir la proporción de eventos correctamente detectados (TP y TN).

$$ACC = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.3}$$

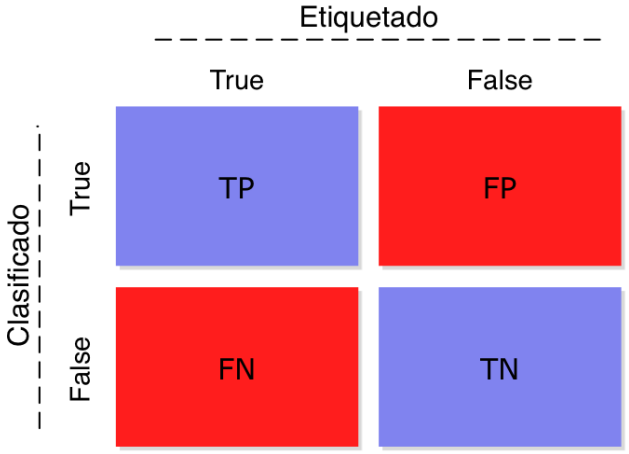


Fig. 5.1: Representación grafica de la matriz de confusión.

Los resultados obtenidos para cada evento se detallan en la tabla 5.1.

Evento	FPR	TPR	ACC
Walking	0.027	0.941	0.957
Running	0.034	0.975	0.967
Evacuation	0.009	0.977	0.991
Crowd Formation	0.056	0.939	0.943
Crowd Splitting	0.005	1	0.995
Local Dispersion	0.035	0.990	0.967

Tabla 5.1: Resultados obtenidos para cada evento.

Como se puede apreciar, se obtienen buenos resultados para todos los eventos, *Accuracy* y *True Positive Rate* cercanos a uno y *False Positive Rate* cercanas a cero.

5.3. Curvas ROC

La curva ROC es una gráfica bidimensional donde el eje Y representa el TPR, y el eje X el FPR. Un clasificador duro (su salida es sólo una etiqueta) produce un único par (FPR,TPR), que se corresponde con un punto en el espacio ROC. Sin embargo, en un clasificador blando (su salida es una probabilidad), al ir variando el umbral que determina la salida dura (θ en la figura 5.2), se obtienen diferentes pares (FPR,TPR) que permiten dibujar la curva ROC completa. Un punto del espacio ROC importante es el (0, 1), que representa la clasificación perfecta. En este punto se detectan correctamente todos los eventos y no se produce ninguna falsa alarma.

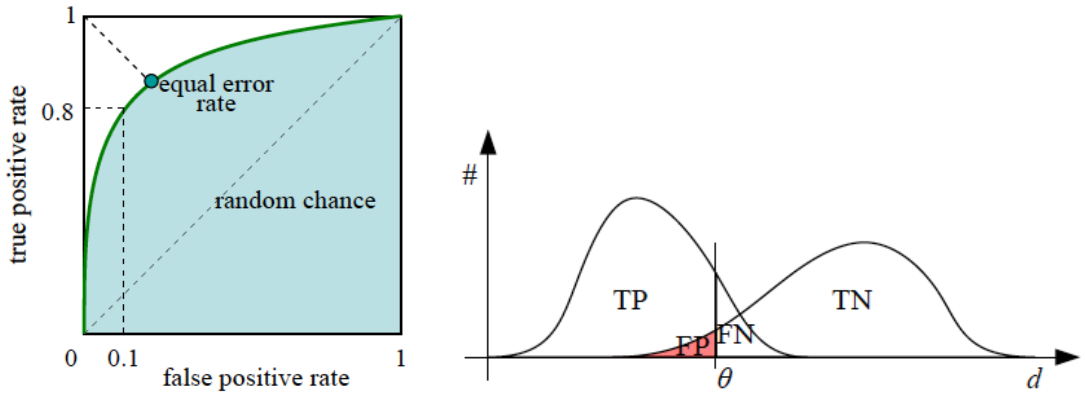


Fig. 5.2: La curva ROC enfrenta TPR y FPR (figura (a)). Idealmente el TPR deberá ser cercano a uno, mientras que el FPR estará cercano a cero. El área bajo la curva ROC es utilizada frecuentemente como medida del rendimiento del clasificador. En la figura (b) se observa como el variar el umbral θ en la salida del clasificador blando, permite obtener varios puntos para dibujar la curva ROC. Imagen tomada de [2].

Las distintas curvas ROC para cada evento se pueden observar en la figura 5.3. Para todos los eventos se obtiene curvas ROC muy buenas incluso algunas de ellas están muy cerca de lograr el clasificador perfecto como es el caso de *Crowd Splitting* y *Evacuation*.

El área bajo la curva ROC se utiliza con bastante frecuencia para indicar el rendimiento del clasificador. El clasificador prefecto obtendría un AUC (*Area Under the Curve*) de uno. En la tabla 5.2 se puede observar el área que obtiene el sistema para cada evento.

Evento	AUC
Walking	0.98391
Running	0.99712
Evacuation	0.90116
Crowd Formation	0.96508
Crowd Splitting	0.99991
Local Dispersion	0.98641

Tabla 5.2: Área bajo la curva ROC para cada evento.

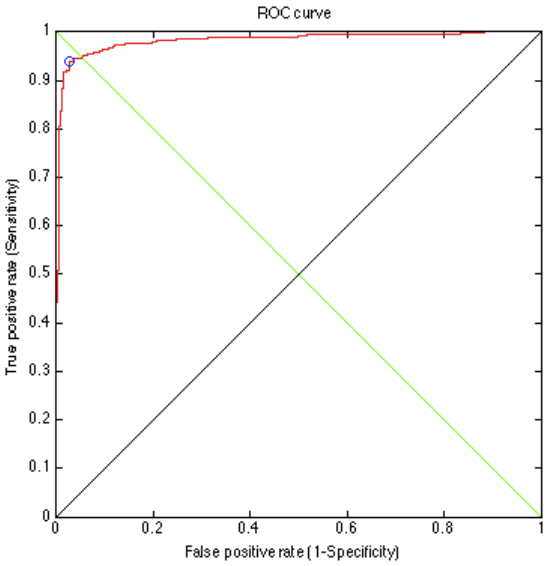
El área obtenida bajo la curva ROC es muy buena en los eventos: *Walking*, *RunningCrowd SplittingLocal* y *Dispersion*, al estar incluidos en el intervalo $[0.97, 1)$ que se considera como test excelente ([21]) y los eventos *Evacuation* y *Crowd Formation* en el intervalo $[0.9, 0.97)$, considerado como test muy bueno.

Los errores existentes en el sistema se producen principalmente por dos motivos:

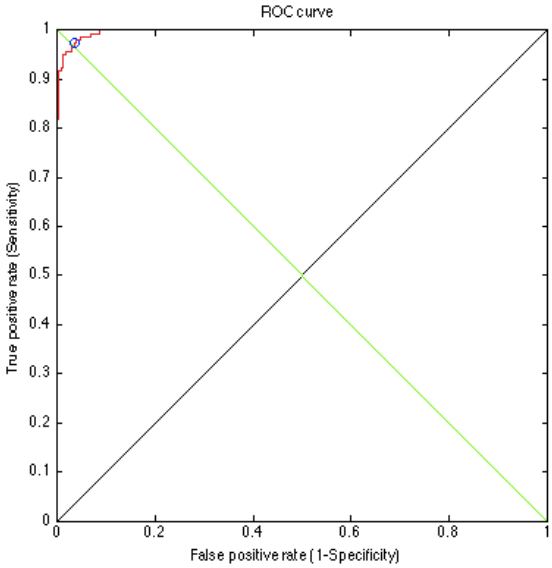
1. El sistema no es capaz de detectar en algunos casos los planos en donde esta empezando a ocurrir el evento, dado que en estos, el evento no está definido completamente y las características que se emplean aún no son lo suficientemente discriminatorias para detectarlo.

2. Al disponer de tan pocos datos de entrenamiento existen algunas situaciones, como por ejemplo el evento *evacuation*, que el sistema no logra modelar correctamente.

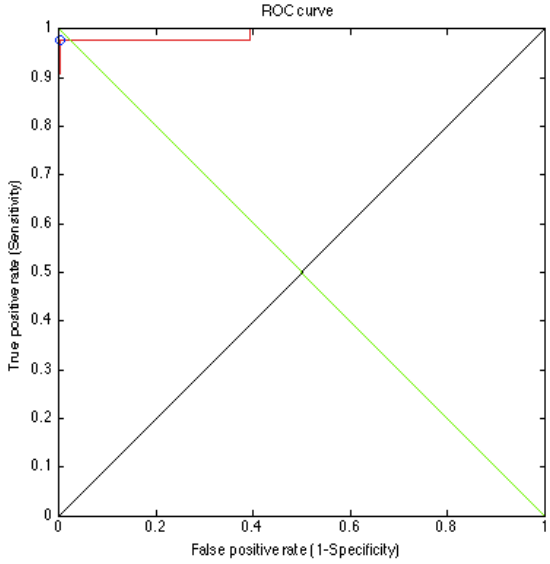
Estos errores se pueden solucionar aumentando el tamaño de la base de datos y incluyendo alguna característica más, que permita detectar los eventos en su comienzo.



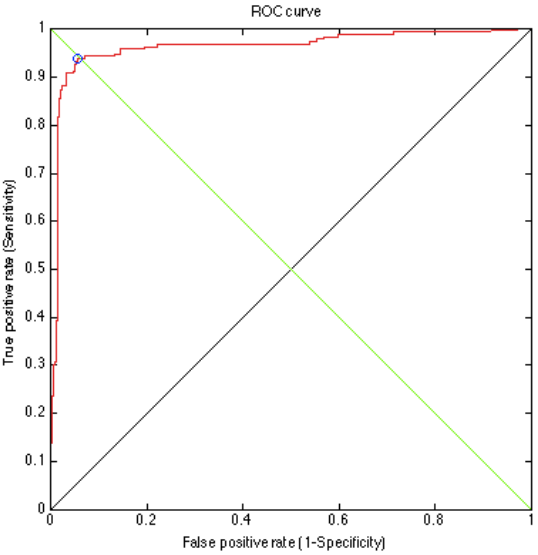
(a) *Walking*



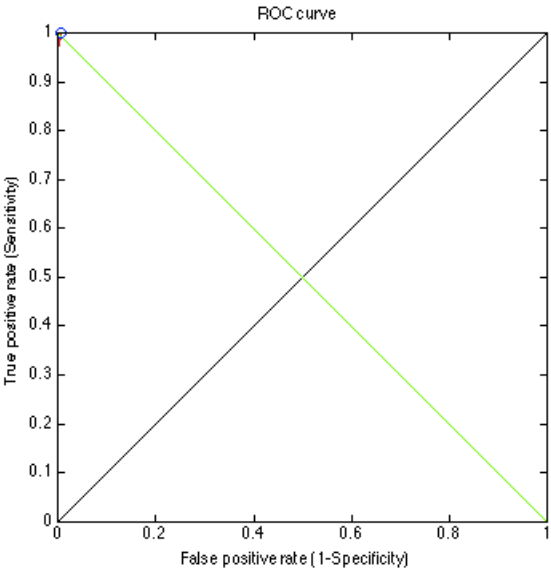
(b) *Running*



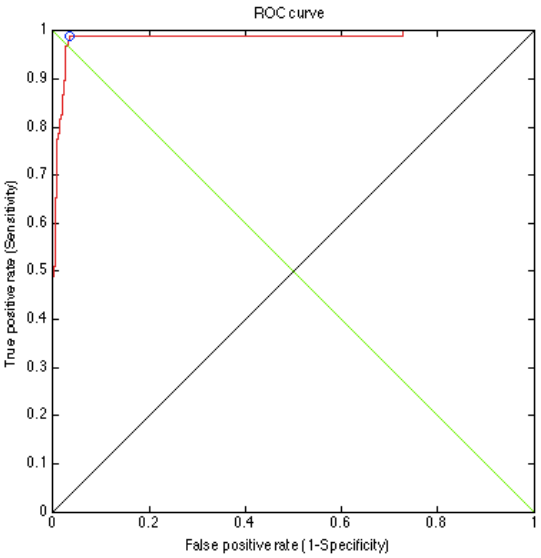
(c) *Evacuation*



(d) *Crowd Formation*



(e) *Crowd Splitting*



(f) *Local Dispersion*

Fig. 5.3: Curvas ROC para los distintos eventos.

5.4. Comparación con otros artículos

Al pertenecer la base de datos utilizada a un concurso (PETS 2009 y PETS2010), se puede comparar el sistema con otros descritos en diferentes artículos. Algunos de estos, detectan explícitamente cada uno de los eventos, mientras que otros abordan el problema desde el punto de vista de la detección de eventos anómalos, donde únicamente indican cuándo ocurre algo que no se considera normal (ocurre cualquier evento diferente a *Walking*).

En la sección 5.4.1 se compara con los artículos que presentan un sistema que detecta todos los eventos y en 5.4.2 con los que solamente detectan eventos anómalos.

5.4.1. Detección de eventos

El artículo [25] presenta un sistema parecido al propuesto, pero no utiliza ningún método de aprendizaje máquina para la clasificación de los eventos, colocando unos umbrales a las diferentes características para lograr un buen rendimiento en esta base de datos en particular. El sistema propuesto en [18] utiliza exclusivamente histogramas de orientaciones de movimiento, para la detección de los distintos eventos. En [9] se detectan grupos de personas presentes en la escena, se divide la imagen en regiones y se cuentan las personas en cada una éstas, posteriormente se usa una SVM para clasificar los diferentes eventos de forma similar a la empleada en este proyecto.

Los artículos [25] y [18] proporcionan la TPR que han obtenido para cada evento. En la tabla 5.3 se observan los resultados obtenidos contra los presentados en estos artículos.

Evento	MP	[25]	[18]
Walking	0.94	**0.92**	—
Running	0.98	**0.92**	**0.80**
Evacuation	0.98	**0.97**	**0.89**
Crowd Formation	0.94	**0.60**	1.00
Crowd Splitting	1	**0.79**	**0.75**
Local Dispersion	0.99	**0.77**	**0.88**

Tabla 5.3: Comparación de los resultados obtenidos (MP) con respecto a los artículos [25] y [18].

El sistema propuesto es muy superior en algunos eventos como: *Crowd Splitting* y *Local Dispersion*; en el resto son aproximadamente iguales (algo superiores) excepto contra el artículo [18] en el evento *Crowd Formation*, donde es superado.

Para comparar con [9] se tiene que emplear la medida ACC (*Accuracy*). Los resultados obtenidos se encuentran en la tabla 5.4.

Evento	MP	[9]
Walking	0.96	**0.87**
Running	0.97	**0.88**
Evacuation	0.99	**0.94**
Crowd Formation	0.94	**0.68**
Crowd Splitting	0.99	**0.77**
Local Dispersion	0.97	**0.80**

Tabla 5.4: Comparación de los resultados obtenidos (MP) con respecto al artículo [9].

El sistema obtiene un mejor rendimiento en todos los eventos comparado con el presentado en [9].

Como se puede observar, salvo en un evento contra el artículo [18], el método propuesto es superior a los demás. Este mayor rendimiento puede deberse a la presencia de una máquina de vectores soporte y a una correcta elección de las características que modelan cada evento.

En particular, la características: histograma de orientaciones de movimiento, parámetro de divergencia y ratio de distancias diseñadas específicamente para este proyecto, dan lugar a muy buenos resultados. También el proceso de selección de características mediante información mutua que obtiene conjuntos óptimos logra aumentar el rendimiento.

5.4.2. Detección de eventos anómalos

Para poder comparar con estos artículos se utilizó la salida de la detección del los eventos específicos, con el fin de poder indicar cuando estaba ocurriendo una anomalía. La probabilidad que el sistema devuelve es la del evento que está ocurriendo o la mayor de ellas, en caso de ocurrir varios a la vez, por lo que se puede calcular la curva ROC. Se tienen en cuenta todos los eventos menos *walking*, dado que las personas caminando no se consideran una anomalía.

Los resultados obtenidos se muestran en la tabla 5.5 y la curva ROC en la figura 5.4.

Evento	AUC	FPR	TPR	ACC
Anomalías	0.97015	0,076	0.904	0.914

Tabla 5.5: Resultados obtendios en la deteccíon de anomalías

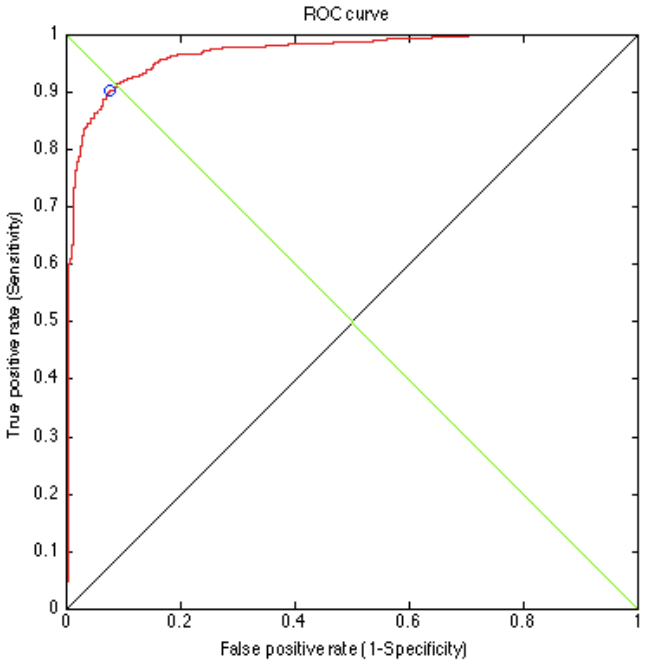


Fig. 5.4: Curva ROC para el detector de eventos anomalos.

Los sistemas con los que se va a realizar la comparación aparecen mencionados a continuación. El artículo [52] se basa en el modelado estadístico de patrones de movimiento, [7] utiliza redes espacio-temporales, [48] emplea filtros de partículas y flujo óptico diferencial, y [9] usa propiedades holísticas.

Los resultados obtenidos en esta comparativa se muestran en la tabla 5.6.

Métodos	Resultado (%)
Método propuesto	91.4
[52]	97.1
[7]	96.4
[48]	96
[9]	**81**

Tabla 5.6: Comparación del porcentaje de anomalías correctamente detectadas del método propuesto con respecto a los artículos [7], [52], [48] y [9].

Como se puede observar, el sistema supera a [9] y obtiene unos resultados cercanos a [52], [7] y [48]. Dado que el método propuesto está diseñado para detectar los eventos individualmente, mientras que estos sistemas han sido desarrollados para detectar solamente anomalías, el rendimiento del sistema puede considerarse aceptable, aún cuando no ha sido diseñado específicamente para esta tarea.

5.5. Vídeos

En la página web `http://www.tsc.uc3m.es/~fsilos/tfg/tfg.html` se pueden visualizar los vídeos del sistema funcionado, tanto en la detección de eventos individuales, como en la de anomalías. Una captura del sistema detectando los eventos individualmente se puede observar en la figura 5.5 y en 5.6 la captura del vídeo en el que el sistema detecta las anomalías.

Fig. 5.5: Captura de vídeo de resultados: Se puede apreciar la probabilidad de que ocurra cada evento en las diferentes barras. Cuando la letra de cada evento cambia al color rojo, el sistema está detectando el evento.

Fig. 5.6: Captura de vídeo de resultados anómalos: cuando la letra de la palabra *Anomaly* cambia al color rojo el sistema está detectando una anomalía.

6

Conclusiones y lineas futuras

6.1. Conclusiones

Este proyecto presenta una forma novedosa de reconocimiento de eventos en secuencias audiovisuales, en las que se ven involucradas un gran número de personas. En este trabajo se han realizado numerosas aportaciones entre las que destacan:

- Empleo del modelo de mezcla de Gaussianas para la sustracción del fondo. La utilización de este modelo ha supuesto una mejora sustancial de los resultados de detección debido a que es capaz de modelar el fondo a partir de datos en los que no están presentes los objetos de interés.

- Diseño de un mecanismo de combinación de modelos de actualización temporal y modelos entrenados para la sustracción del fondo. Al combinar el modelo de mezcla de Gaussianas con el modelo temporal se han resueltos los problemas presentes en el modelo de mezcla de Gaussianas, provocados por cambios de iluminación presentes en la base de datos de entrenamiento, lo que ha logrado mejorar el rendimiento notablemente.

- Inclusión de características innovadoras no presentes en la literatura. Ésta se ha convertido en la aportación más importante de todo el trabajo, ya que las características son las encargadas de modelar los distintos eventos. Entre las más innovadoras destacan el parámetro de divergencia y el histograma de orientaciones del movimiento, las cuales constituyen descriptores especialmente discriminativos para algunos eventos de interés.

- Estudio sistemático para la selección de las distintas características mediante información mutua. Aunque las características sean buenas, es necesario un proceso que seleccione de entre todas ellas, el subconjunto que mejor defina cada evento. Se ha podido lograr esta selección gracias al estudio de la información mutua.

- Empleo de máquinas de vectores soporte para las tareas de clasificación. Al ser una base de datos con pocas muestras de entrenamiento, las SVM, debido a su capacidad de generalización, logran aumentar de forma muy significativa el rendimiento del sistema.

Estas aportaciones han supuesto aumentar en gran medida el rendimiento del sistema, resultando éste muy eficaz y prometedor. Gracias a estas mejoras, el sistema se ha podido comparar positivamente con otros propuestos en la literatura, siendo su rendimiento superior en la mayoría de los casos.

En cuanto a la usabilidad del sistema en situaciones reales, puede afirmarse que aún es necesario seguir desarrollándolo, dado que la base de datos empleada es muy concreta y no se adapta al mundo real. Resultaría necesario, por tanto, entrenar y probar éste en situaciones más realistas. Además al estar implementado en Matlab, la ejecución en tiempo real del sistema

no resulta posible, siendo deseable implementarlo utilizando otras tecnologías que sí permitan cumplir con este requisito.

Debido al pequeño tamaño de la base de datos, existe una sobreadaptación de todos los sistemas, incluido el desarrollado en este trabajo, por lo que sería necesario aumentar ésta para poder realizar una comparación más veraz y realista.

A la vista de todo lo anterior, queda claro que el sistema desarrollado es bastante completo y competitivo. También es destacable la extensa revisión de la literatura concerniente al ámbito de la vídeo vigilancia, el tratamiento de imagen y el aprendizaje máquina que ha sido necesaria para poder realizar este trabajo.

6.2. Lineas futuras

Algunas líneas futuras para poder ampliar la investigación desarrollada en este proyecto ordenadas por necesidad, son:

1. Ampliación de la base de datos: Son necesarias más secuencias de los distintos eventos dado que la base de datos PETS 2010 no resulta del todo completa.

2. En este proyecto se detectan únicamente seis eventos, por lo que sería interesante incluir otros tipos de eventos. Para ésto es necesario aumentar la base de datos con vídeos donde ocurran estos nuevos eventos.

3. Resultaría interesante implementar el sistema sobre una cámara estática, para poder comprobar si el funcionamiento es correcto en una situación real. Como se puede observar, algunos de los vídeos de la base de datos resultan un poco irreales.

4. Utilizar otras tecnologías para implementar el algoritmo. Al estar implementado en MATLAB no resulta eficiente y, para aplicaciones en tiempo real, sería interesante programarlo usando otras tecnologías como por ejemplo OpenCV [6].

5. Inclusión de nuevas características que permitan mejorar los resultados

6. Implementación de un sistema de detección de anomalías desde cero, dado que el actual está basado en la detección de los eventos específicos.

7

Presupuesto

En este capítulo se presentan los cálculos asociados a la realización del proyecto. El presupuesto se desglosa en costes materiales (sección 7.1), costes personales (sección 7.2) y el coste total (sección 7.3).

7.1. Costes materiales

Los costes materiales asociados a este proyecto han sido los siguientes:

- **Espacio de trabajo:** Se incluyen los costes de electricidad, mantenimiento, mobiliario, etc. El alquiler del espacio de trabajo tiene un coste de 1000 €al mes, al tratarse de un espacio compartido entre 7 personas, tiene un coste asociado de 143 €al mes. La duración del proyecto ha sido aproximadamente de 6 meses por lo que el coste asociado al espacio de trabajo asciende a 853 €.

- **Ordenador personal:** Se emplea para la programación de los distintos algoritmos, y la escritura de la memoria del proyecto. El equipo empleado tiene un coste de 1100 €, como éste pude ser reutilizado después, su coste se puede amortizar en cuatro años, por lo que el coste asociado asciende a 138 €.

- **Ordenador pruebas:** Ha sido necesario utilizar un ordenador adicional para la realización de pruebas computacionalmente muy costosas. Al estar compartido con el departamento donde se ha realizado el proyecto su coste estimado es de 100 €.

- **Software** Se ha intentado que todos los programas empleados fueran de software libre. Solamente se han necesitado licencias de programas de pago para:

  - Matlab R2007b de MathWorks: Utilizado para la implementación del sistema, cuya licencia tiene un coste de 2000 €, dado que se puede amortizar en 4 años, supone la cantidad de 250 €.

- **Linea de datos ADSL:** La tarifa plana ADSL tiene un coste mensual de 60 €, compartido entre las 7 personas del laboratorio, durante 6 meses, tiene un coste asociado de 52 €.

- **Material de oficina:** Todo el gasto referente a material de oficina: papel, impresiones, bolígrafos, etc. Tiene un coste estimado de 30 €.

En la tabla 7.1 se resumen todos los costes materiales asociados a este proyecto.

Descripción	Coste imputable
Espacio de trabajo	143 €
Ordenador personal	138 €
Ordenador pruebas	100 €
Matlab R2007b de MathWorks	250 €
Linea de datos ADSL	52 €
Material de oficina	30 €
TOTAL	**712 €**

Tabla 7.1: Resumen de los costes materiales.

7.2. Costes personales

Para calcular los costes personales asociados a este proyecto es necesario tener en cuenta la duración y las horas de trabajo realizadas en el mismo. El tiempo empleado para la realización del proyecto ha sido de 6 meses, con una jornada de trabajo de 4 horas diarias, cinco días a la semana, por tanto el número total de horas ha sido de 480. Los honorarios del ingeniero realizador del proyecto son de 36 €/hora dando lugar a un gasto total de 17280 €.

En cuanto a la dirección del proyecto se han considerados unos costes asociados de 3600 €, por 50 horas trabajadas a 72 €/hora.

En la tabla 7.2 se resumen todos los costes personales asociados a este proyecto.

Persona	Honorario	Horas	Total
Fernando de la Calle Silos	36 €/hora	480	17280 €
Iván González Díaz	72 €/hora	50	3600 €
		TOTAL	**20880 €**

Tabla 7.2: Resumen de los costes personales.

7.3. Presupuesto total

Considerando los costes materiales y personales anteriormente detallados el coste total del proyecto se resume en la tabla 7.3.

Descripción	Coste imputable
Costes Materiales	712 €
Costes Personales	20880 €
Subtotal	21592 €
IVA (21 %)	4534,32 €
TOTAL	**26126,32 €**

Tabla 7.3: Resumen del presupuesto total.

El presupuesto total del proyecto asciende a VEINTISÉIS MIL CIENTO VEINTISÉIS EUROS

Fdo: Fernando de la Calle Silos.
Graduado en Ingeniería de Sistemas Audiovisuales.

Apéndices

A

Base de datos PETS 2010

La base de datos PETS 2010, que fue creada para un concurso que se realizó en el año 2009 y en 2010, ha sido la utilizada en este proyecto. Esta disponible para su descarga en: `http://www.cvg.rdg.ac.uk/PETS2010/`

Esta base de datos se divide en varios subconjuntos, como se puede observar en la figura A.1 y ,además, cada secuencia está grabada desde diversos puntos de vista con diferentes cámaras sincronizadas.

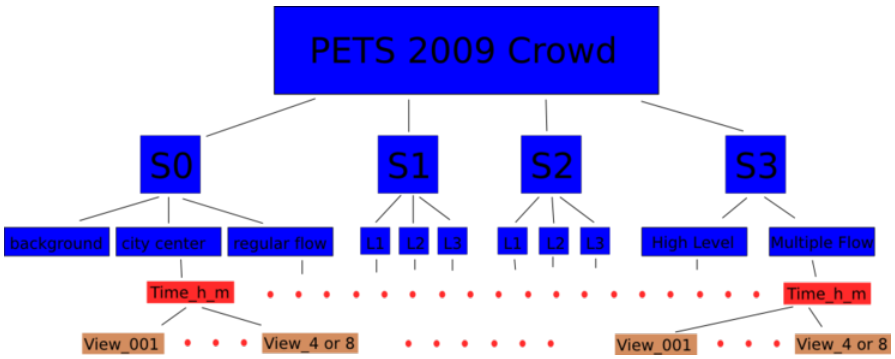


Fig. A.1: Representación grafica de la estructura de la base de datos.

A.1. Datasets

Los subconjuntos que se han utilizado para este proyecto se detallan a continuación.

A.1.1. Dataset S0: Training Data: Background

Este conjunto de imágenes consta de varias secuencias en las que no existe flujo de personas. Se ha utilizado por tanto, para desarrollar un modelo de fondo para todas las cámaras.

A.1.2. Dataset S3: Flow Analysis and Event Recognition: High Level

La principal característica de las escenas de este conjunto es que en ellas aparecen grandes aglomeraciones de personas. La tarea es detectar y identificar el principio y el final de cada uno de los siguientes eventos:

- *Walking:* Representa a un número significativo de personas desplazándose lentamente.

- *Running:* Representa a un número significativo de personas desplazándose rápidamente.
- *Evacuation:* Representa la dispersión rápida en diferentes direcciones de una multitud.
- *Crowd Formation:* Representa la unión en un grupo de un gran número de individuos provenientes de diferentes direcciones.
- *Crowd Splitting:* Representa la división de un grupo de individuos, en dos o más grupos que toman diferentes direcciones.
- *Local Dispersion:* Representa la dispersión de un pequeño grupo de individuos de una multitud.

Consta de cuatro secuencias pero se ha considerado como en [25] que tres de ellas (`14:16, 14:27 y 14:33`) se pueden dividir para realizar un mejor análisis, dado que los eventos que ocurren son diferentes. Los planos donde se realiza la división se aprecian en la tabla A.1.

Nombre	Plano Inicial	Plano Final
14:16-A	0	107
14:16-B	108	222
14:27-A	0	184
14:27-B	185	333
14:31	0	130
14:33-A	0	310
14:33-A	311	377

Tabla A.1: Planos en los que se divide cada secuencia.

Una descripción de las distintas secuencias se puede encontrar a continuación:

- `14:16-A:` Las personas entran caminando por la derecha y cuando llegan al centro de la imagen empiezan a correr. Ocurren los eventos: *walking* y *running*.
- `14:16-B:` Las personas entran caminando por la izquierda y cuando llegan al centro de la imagen empiezan a correr. Ocurren los eventos: *walking* y *running*.
- `14:27-A:` Hay un grupo de personas en el centro de la imagen, al final del vídeo éstas se dispersan levemente. Ocurren los eventos: *walking* y *local dispersion*.
- `14:27-B:` Hay un grupo de personas en el centro de la imagen, al final del vídeo estas se dispersan levemente. Ocurren los eventos: *walking* y *local dispersion*.
- `14:31:` Las personas entran caminando por la derecha y cuando llegan al centro de la imagen se dividen en tres grupos diferentes. Ocurren los eventos: *walking* y *crowd splitting*.
- `14:33-A:` Las personas entran caminado por todos los bordes de la imagen para terminar juntándose en el centro. Ocurren los eventos: *walking* y *crowd formation*.
- `14:33-B:` Aparece un grupo en el centro de la imagen, después todas las personas salen corriendo en direcciones diferentes. Ocurren los eventos: *walking*, *running* y *evacuation*.

A.2. Ubicación

Los vídeos han sido grabados en *University of Reading, UK* cubriendo un área aproximada de 100 × 30 metros. Las secuencias con las que se ha trabajado están grabadas desde cuatro cámaras diferentes (`view 001,view 002,view 003,view 004`). La situación de éstas, se puede observar en la imagen A.2. Un ejemplo de planos captados desde estas posiciones se muestra en A.3

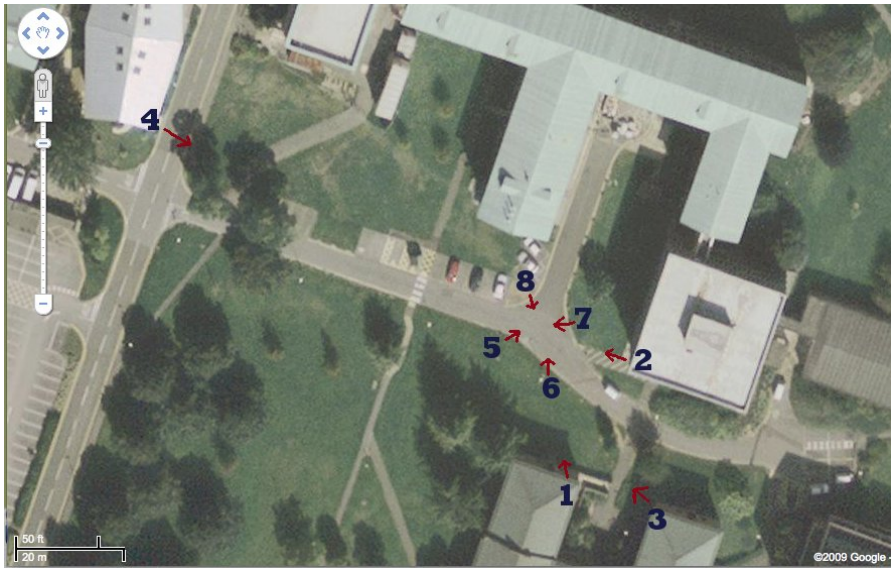


Fig. A.2: Plano de la situación de las cámaras.

(a) view 001

(b) view 002

(c) view 003

(d) view 004

Fig. A.3: Imágenes de ejemplo de las distintas cámaras.

A.3. Cámaras

En la tabla A.2 aparecen los modelos y principales características técnicas de las cámaras que se utilizaron para realizar esta base de datos.

Cámara	Modelo	Resolución	fps
01	Axis 223M	768 × 576	∼7
02	Axis 223M	768 × 576	∼7
03	PTZ Axis 233D	768 × 576	∼7
04	PTZ Axis 233D	768 × 576	∼7

Tabla A.2: Detalles de las cámaras.

Estas cámaras (`Axis 223M` y `PTZ Axis 233D`) como se aprecia en la figura A.4, se corresponden con los típicos modelos utilizados en vídeo vigilancia. Las características técnicas específicas aparecen en la página web del fabricante para la `Axis 223M` en:`http://www.axis.com/es/products/cam_223m/index.htm` y para la `PTZ Axis 233D` en: `http://www.axis.com/es/products/cam_233d/`.

(a) `Axis 223M` (b) `Axis 223M`

Fig. A.4: Cámaras empleadas en la grabación de la base de datos.

B

Formulación mezcla de Gaussianas

Este apéndice trata de forma general el ajuste de una mezcla de Gaussianas a unos datos. Parte del desarrollo matemático presentado a continuación se ha obtenido de [1] y el algoritmo EM empleado se puede encontrar en [20].

B.1. Mezcla de Gaussianas

Una mezcla de Gaussianas es una suma ponderada de k Gaussianas, definida como:

$$p(y|\theta) = \sum_{j=1}^{k} \omega_j \phi(y|\mu_j, \Sigma_j)$$

donde ω_j son los pesos de cada una de las componentes de la mezcla, y cumplen que: $\sum_{j=1}^{k} \omega_j = 1$ y $\omega_j > 0$, y $\phi(y_i|\mu_j, \Sigma_j)$ es la Gaussiana asociada a cada componente definida como:

$$\phi(y_i|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(y-\mu_j)\,\Sigma_j^{-1}\,(y-\mu_j)^{\top}\right)$$

Dadas n muestras independientes $y_1, y_2, y_3, \cdots, y_n \in R^d$ tomadas de una mezcla de Gaussianas con parámetros: $\theta = \{(\omega_j, \mu_j, \Sigma_j)\}_{j=1}^{k}$, se debe encontrar el θ que maximize la función de verosimilitud:

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \log P(y|\theta)$$

Mediante estas n muestras independientes la función de verosimilitud logarítmica puede definirse como:

$$\begin{aligned} L(\theta) &= \log\left\{\prod_{i=1}^{n} p(y_i|\theta)\right\} = \\ &= \sum_{i=1}^{n} \log p(y_i|\theta) = \\ &= \sum_{i=1}^{n} \log\left\{\sum_{j=1}^{k} \omega_j \phi(y_i|\mu_j, \Sigma_j)\right\} \geq \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij} \frac{\omega_j \phi(y_i|\mu_j, \Sigma_j)}{\gamma_{ij}} \end{aligned}$$

Para ello se utiliza el algoritmo *Expectation Maximization* (EM), que parte de una suposición acerca de los datos completos, para luego encontrar el θ que maximiza el valor esperado de la log-verosimilitud. Una vez que se dispone del nuevo θ, se puede realizar una mejor elección. El algoritmo EM consta de dos pasos: E (*Expectation*) y M (*Maximization*) que se repiten hasta que se cumple un cierto criterio de parada.

B.1.1. Paso E

Se define la probabilidad a posteriori de que la i-ésima muestra pertenezca a la j-ésima Gaussiana como:

$$\gamma_{ij}^{(m)} = \frac{\omega_j^{(m)} \phi(y_i|\mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \phi(y_i|\mu_l^{(m)}, \Sigma_l^{(m)})}$$

que cumple que $\sum_{j=1}^{k} \gamma_{ij}^{(m)} = 1$

Usando la desigualdad de Jensen [34], se define una cota inferior de la $\log L$ denotada $Q(\theta|\theta^{(m)})$, que es más facil de maximizar:

$$\log L \geq Q(\theta|\theta^{(m)}) - \gamma_{ij}^{(m)} \log \gamma_{ij}^{(m)} \propto Q(\theta|\theta^{(m)})$$

Por tanto:

$$\begin{aligned} Q(\theta|\theta^{(m)}) &= E_{Z_i|y_i,\theta^{(m)}}[\log p_X(y_i, Z_i|\theta)] = \\ &= \sum_{j=1}^{k} \gamma_{ij}^{(m)} \log p_X(y_i, j|\theta) \\ &= \sum_{j=1}^{k} \gamma_{ij}^{(m)} \log \omega_j \phi(y_i|\mu_j, \Sigma_j) = \\ &= \sum_{j=1}^{k} \gamma_{ij}^{(m)} \left(\log \omega_j - \frac{1}{2} \log |\Sigma_j| - \frac{1}{2} (y_i - \mu_j)^T \Sigma_j^{-1} (y_i - \mu_j) \right) + c \end{aligned}$$

La función $Q(\theta|\theta^{(m)})$ queda definida como:

$$Q(\theta|\theta^{(m)}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij}^{(m)} \left(\log \omega_j - \frac{1}{2} \log |\Sigma_j| - \frac{1}{2} (y_i - \mu_j)^T \Sigma_j^{-1} (y_i - \mu_j) \right)$$

B.1.2. Paso M

Definiendo:

$$n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)}$$

La función $Q(\theta|\theta^{(m)})$ se puede reescribir como:

$$Q(\theta|\theta^{(m)}) = \sum_{j=1}^{k} n_j^{(m)} \left(\log \omega_j - \frac{1}{2} \log |\Sigma_j| \right) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij}^{(m)} \log |\Sigma_j| - \frac{1}{2} (y_i - \mu_j)^T \Sigma_j^{-1} (y_i - \mu_j)$$

El paso M consiste en maximizar $Q(\theta|\theta^{(m)})$, bajo la restricción: $\sum_{j=1}^{k} \omega_j = 1, \quad \omega_j > 0, \quad j = 1, ..., k$.

Para maximizar una función no lineal $f(x_i, x_2, ..., x_n)$ sujeta a una serie de restricciones dadas por: $g_1(x_1, x_2, ..., x_n) = b_1 \quad g_2(x_1, x_2, ..., x_n) = b_2 \quad ... \quad g_m(x_1, x_2, ..., x_n) = b_m$, se debe determinar en primer lugar el Lagrangiano asociando un multiplicador de Lagrange λ por cada restricción:

$$J(x_1, x_2, ..., x_n, \lambda_1, \lambda_2, ..., \lambda_m) = f(x_i, x_2, ..., x_n) + \sum_{i=1}^{m} \lambda_i \left[b_i - g_i(x_1, x_2, ..., x_n) \right]$$

Para después resolver el sistema de n+m ecuaciones:

$$\begin{aligned} \frac{\partial J}{\partial x_j} &= 0 \quad \forall j = 1, 2, ..., n \\ \frac{\partial J}{\partial \lambda_j} &= 0 \quad \forall j = 1, 2, ..., m \end{aligned}$$

En este caso como $Q(\theta|\theta^{(m)})$ no es lineal se forma el Lagrangiano:

$$J(\omega, \lambda) = \sum_{j=1}^{k} n_j^{(m)} \log \omega_j + \lambda \left(\sum_{j=1}^{k} \omega_j - 1 \right)$$

Nótese que se eliminan los términos que no dependen de ω .

Para encontrar los pesos se deriva respecto a w_j:

$$\frac{\partial J}{\partial \omega_j} = \sum_{j=1}^{k} n_j^{(m)} + \lambda = 0 \quad j = 1, ..., k$$

Obteniendo k ecuaciones.

Derivando respecto a λ obtenemos otra ecuación:

$$\frac{\partial J}{\partial \lambda} = \sum_{j=1}^{k} \omega_j - 1 = 0 \quad \Longrightarrow \quad \sum_{j=1}^{k} \omega_j = 1$$

Finalmente se resuelve el sistema de $k+1$ ecuaciones:

$$\omega_j^{(m+1)} = \frac{n_j^{(m)}}{\sum_{j=1}^{k} n_j^{(m)}} = \frac{n_j^{(m)}}{n} \quad \text{para:} \quad j = 1, ..., k.$$

De esta forma se obtienen los pesos.

Realizando este mismo procedimiento se obtienen $\mu_j^{(m+1)}$ y $\Sigma_j^{(m+1)}$:

$$\begin{aligned} \mu_j^{(m+1)} &= \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i \quad \text{para:} \quad j = 1, ..., k. \\ \Sigma_j^{(m+1)} &= \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} \left(y_i - \mu_j^{(m+1)} \right) \left(y_i - \mu_j^{(m+1)} \right)^T \quad \text{para:} \quad j = 1, ..., k. \end{aligned}$$

B.1.3. Criterio de parada

Se calcula la nueva verosimilitud:

$$L^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \omega_j^{(m+1)} \phi(y_i|\mu_j^{(m+1)}, \Sigma_j^{(m+1)} \right)$$

Si $|L^{(m+1)} - L^(m)| > \delta$ donde δ es el umbral de parada se realiza una nueva iteración (Paso E - Paso M), en caso contrario el algoritmo finaliza.

B.2. Resumen

Una mezcla de Gaussianas es una suma ponderada de k Gaussianas, definida como:

$$p(y|\theta) = \sum_{j=1}^{k} \omega_j \phi(y|\mu_j, \Sigma_j)$$

Donde:

$$\phi(y|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_j|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (y - \mu_j) \Sigma_j^{-1} (y - \mu_j)^\top \right)$$

Dadas n observaciones independientes, se desea estimar los parámetros:

$$\theta = \{(\omega_i, \mu_i, \Sigma_i)\}_{i=1}^{k}$$

Para lograr estos parámetros, se utiliza el algoritmo EM, cuyos pasos quedan definidos a continuación: El algoritmo empleado sigue los siguientes pasos:

1. **Inicialización:** Elegir los valores iniciales para los parámetros que se desea estimar: $w_j^{(0)}, \mu_j^{(0)}, \Sigma_j^{(0)}$ $j = 1, ..., k$. y calcular la verosimilitud:

$$L(\theta)^{(0)} = \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \omega_j^{(0)} \phi(y_i | \mu_j^{(0)}, \Sigma_j^{(0)} \right)$$

2. **Paso E:** Calcular:

$$\gamma_{ij}^{(m)} = \frac{\omega_j^{(m)} \phi(y_i | \mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \phi(y_i | \mu_l^{(m)}, \Sigma_l^{(m)})} \quad \text{para:} \quad i = 1, ..., n \quad \text{y} \quad j = 1, ..., k$$

y:

$$n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)} \quad \text{para:} \quad j = 1, ..., k$$

3. **Paso M:** Calcular los nuevos valores que se quieren estimar:

$$\begin{aligned}
\omega_j^{(m+1)} &= \frac{n_j^{(m)}}{n} \quad \text{para:} \quad j = 1, ..., k. \\
\mu_j^{(m+1)} &= \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i \quad \text{para:} \quad j = 1, ..., k. \\
\Sigma_j^{(m+1)} &= \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} \left(y_i - \mu_j^{(m+1)} \right) \left(y_i - \mu_j^{(m+1)} \right)^T \quad \text{para:} \quad j = 1, ..., k.
\end{aligned}$$

4. **Test de convergencia:** Calcular la nueva verosimilitud:

$$L(\theta)^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \omega_j^{(m+1)} \phi(y_i | \mu_j^{(m+1)}, \Sigma_j^{(m+1)} \right)$$

Volver al paso 2 si $|L^{(m+1)} - L^(m)| > \delta$ donde δ es un umbral elegido anteriormente, en caso contrario el algoritmo finaliza.

C

Algoritmo RANSAC

El algoritmo *RANdom SAmple Consensus* (RANSAC) propuesto por Fischler y Bolles [23] se diseño para estimar parámetros en un conjunto de datos con un gran número de *outliers*.

RANSAC es una técnica de remuestreo que genera conjuntos de soluciones usando el mínimo número de observaciones necesarias para estimar el modelo. Al contrario que las técnicas convencionales en las cuales se usan todos los datos disponibles para obtener una solución inicial y a partir de esta intentar eliminar los *outliers*, RANSAC usa el conjunto más pequeño de datos posible para la solución inicial y aumenta este conjunto con datos consistentes.

Un resumen básico del algoritmo se puede observar a continuación:

Algorithm 4 RANSAC

1: Seleccionar aleatoriamente el número mínimo de puntos necesarios para determinar los parámetros del modelo.
2: Encontrar los parámetros del modelo.
3: Determinar cuántos puntos del conjunto completo se corresponden con el modelo con una tolerancia ϵ (número de *inliers*).
4: **if** El número de *inliers* sobre el número de puntos totales es mayor que un umbral τ **then**
5: Volver a estimar los parámetros del modelo usando todos los *inliers* del conjunto completo de puntos.
6: **else**
7: Repetir de 1-4 un máximo de N veces.
8: **end if**

El número de iteraciones N tiene que elegirse de forma que se asegure que la probabilidad p de que al menos uno de los subconjuntos elegidos aleatoriamente no incluya ningún *outlier*.

Siendo u la probabilidad de que cualquier punto sea un *inlier*, $v = 1 - u$ la probabilidad de que sea un *outlier* y m el número mínimo de puntos necesarios para estimar el modelo. Se define:

$$1 - p = (1 - u^m)^N$$

Despejando el número de iteraciones N:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)}$$

Para una formulación completa del algoritmo RANSAC consultar en [23] y [29].

Bibliografía

[1] *Finite Mixture Models*. Wiley, 2000.

[2] *Computer Vision: Algorithms and Applications*. Springer, 2010.

[3] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174 –188, feb 2002.

[4] R. Benmokhtar and I. Laptev. Inria-willow at trecvid 2010 : Surveillance event detection. In *TRECVID 2010 Workshop*.

[5] Isabelle M. GUYON BOSER, Bernhard E. and Vladimir N. VAPNIK. A training algorithm for optimal margin classifiers.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] A. Briassouli and I. Kompatsiaris. Spatiotemporally localized new event detection in crowds. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 928 –933, nov. 2011.

[8] Simone Calderara, Rudy Melli, Andrea Prati, and Rita Cucchiara. Reliable background suppression for complex scenes. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, VSSN '06, pages 211–214, New York, NY, USA, 2006. ACM.

[9] A. B. Chan, M. Morrow, and N. Vasconcelos. Analysis of crowded scenes using holistic properties. In *In IEEE Intl. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2009)*, june 2009.

[10] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] M.-Y. Chen and A. Hauptmann. Mosift : Recognizing human actions in surveillance videos. *Transform*, pages 1–16.

[12] Sen ching S. Cheung and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video.

[13] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603 –619, may 2002.

[14] Corinna; Cortes and Vladimir N. Vapnik. Support-vector networks.

[15] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1337 – 1342, oct. 2003.

[16] N. Dalal. *Finding People in Images and Videos*. PhD thesis, Institut National Polytechnique de Grenoble / INRIA Grenoble , 2006.

[17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, june 2005.

[18] H.M. Dee and A. Caplier. Crowd behaviour analysis using histograms of motion direction. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1545 –1548, sept. 2010.

[19] H.M. Dee and A. Caplier. Crowd behaviour analysis using histograms of motion direction. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1545 –1548, sept. 2010.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977.

[21] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[22] Jie Feng, Chao Zhang, and Pengwei Hao. Online learning with self-organizing maps for anomaly detection in crowd scenes. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3599 –3602, aug. 2010.

[23] Martin A. Fischler and Robert C. Bolles. Readings in computer vision: issues, problems, principles, and paradigms. chapter Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[24] S. Foucher and M. Lalonde andL. Gagnon. Crim's notebook paper - trecvid 2010 surveillance event detection. In *TRECVID 2010 Workshop*.

[25] C. Garate, P. Bilinsky, and F. Bremond. Crowd event recognition using hog tracker. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, pages 1 –6, dec. 2009.

[26] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB'99*.

[27] X. Guo, Y. Chen, W. Liu, Y. Mao, H. Zhang, K. Zhou, L. Wang, Y. Hua, Z. Zhao, Y. Zhao, and A. Cai. Bupt-mcprl at trecvid 2010. In *TRECVID 2010 Workshop*.

[28] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[29] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[30] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. 2010. Available at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[31] Zhi-Kai Huang and De-Hui Liu. Segmentation of color image using em algorithm in hsv color space. In *Information Acquisition, 2007. ICIA '07. International Conference on*, pages 316 –319, july 2007.

[32] N. Ihaddadene and C. Djeraba. Real-time crowd motion analysis. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.

[33] R.S. Jadon, Santanu Chaudhury, and K.K. Biswas. A fuzzy theoretic approach to camera motion detection. *Proceedings of 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2002.

[34] J. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906. 10.1007/BF02418571.

[35] Fan Jiang, Ying Wu, and A.K. Katsaggelos. Abnormal event detection from surveillance video by dynamic hierarchical clustering. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V –145 –V –148, 16 2007-oct. 19 2007.

[36] Y. Kawai, M. Takahashi, M. Fujii, and M. Naemura. Nhk strl at trecvid 2010: Semantic indexing and surveillance event detection. In *TRECVID 2010 Workshop*.

[37] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 4, pages 3776 –3781 vol.4, dec 1994.

[38] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[39] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1446 –1453, june 2009.

[40] I. Laptev, C. Schuldt B. Caputo, and T. Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. In *Computer Vision and Image Understanding*.

[41] H. Li, L. Bao, Z. Gao, A. Overwijk, W. Liu, LF. Zhang, S-I. Yu, MY. Chen, F. Metze, and A. Hauptmann. Informedia @ trecvid2010. In *TRECVID 2010 Workshop*.

[42] A.A. Liu and Z. Gao. Mmm-tju at trecvid 2010. In *TRECVID 2010 Workshop*.

[43] Hui Liu, Chenhui Yang, Xiao Shu, and Qicong Wang. A new method of shadow detection based on edge information and hsv color information. In *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on*, volume 1, pages 286 –289, dec. 2009.

[44] B.P.L. Lo and S.A. Velastin. Automatic congestion detection system for underground platforms. In *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, pages 158 –161, 2001.

[45] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[46] V. Mahadevan, Weixin Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1975 –1981, june 2010.

[47] H. Medeiros, J. Park, and A. Kak. A parallel color-based particle filter for object tracking. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –8, june 2008.

[48] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935 –942, june 2009.

[49] M. Michel, J. Fiscus, and P. Over. Trecvid 2010 video surveillance event detection task. In *TRECVID 2010 Workshop*.

[50] A.T. Nghiem, F. Bremond, and M. Thonnat. Shadow removal in indoor scenes. In *Advanced Video and Signal Based Surveillance, 2008. AVSS '08. IEEE Fifth International Conference on*, pages 291 –298, sept. 2008.

[51] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831 –843, aug 2000.

[52] S.S. Pathan, A. Al-Hamadi, and B. Michaelis. Crowd behavior detection by statistical modeling of motion patterns. In *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*, pages 81 –86, dec. 2010.

[53] I.T. Phillips and A.K. Chhabra. Empirical performance evaluation of graphics recognition systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):849 –870, sep 1999.

[54] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099 – 3104 vol.4, oct. 2004.

[55] P. Wayne Power and Johann A. Schoonees. Understanding background mixture models for foreground segmentation.

[56] J. R. Quinlan. Induction of decision trees. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[57] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[58] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[59] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[60] Li Shuhua and Guo Gaizhi. The application of improved hsv color space model in image processing. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 2, pages V2–10 –V2–13, may 2010.

[61] Stephen M. Smith and J. Michael Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[62] IEEE Computer Society, IEEE Signal Processing Society, and Boston University. Performance evaluation of tracking and surveillance 2010 database. `http://www.cvg.rdg.ac.uk/PETS2010/`, 2010.

[63] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 2 vol. (xxiii+637+663), 1999.

[64] Harald Stögbauer, Alexander Kraskov, Sergey A. Astakhov, and Peter Grassberger. Least-dependent-component analysis based on mutual information. *Phys. Rev. E*, 70:066123, Dec 2004.

[65] Duy-Nguyen Ta, Wei-Chao Chen, N. Gelfand, and K. Pulli. Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2937 –2944, june 2009.

[66] G. Tian, Z. Wang R. Hu, and Y. Fu. Improved object tracking algorithm based on new hsv color probability model. In *Proceedings of the 6th International Symposium on Neural Networks: Advances in Neural Networks - Part II*.

[67] Peking University-IDM. Pku@trecvid2010: Pair-wise event detection in surveillance video. In *TRECVID 2010 Workshop*.

[68] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 734 –741 vol.2, oct. 2003.

[69] Shu Wang and Zhenjiang Miao. Anomaly detection in crowd scene. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1220 –1223, oct. 2010.

[70] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 51 –56, oct 1996.

[71] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780 –785, jul 1997.

[72] Zoran Zivkovic and Ferdinand van der Heijden. Effficient adaptive density estimation per image pixel for the task of background subtraction. 2006.