UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

# SOFT COMPUTING AND NON-PARAMETRIC TECHNIQUES FOR EFFECTIVE VIDEO SURVEILLANCE SYSTEMS

**Autor:**

ÓSCAR PÉREZ CONCHA

**Directores:**

JESÚS GARCÍA HERRERO
JOSÉ MANUEL MOLINA LÓPEZ

DEPARTAMENTO DE INFORMÁTICA

Colmenarejo, marzo 2008

TESIS DOCTORAL

# SOFT COMPUTING AND NON-PARAMETRIC TECHNIQUES FOR EFFECTIVE VIDEO SURVEILLANCE SYSTEMS

| | |
|---|---|
| Autor: | ÓSCAR PÉREZ CONCHA |
| Directores: | JESÚS GARCÍA HERRERO |
| | JOSÉ MANUEL MOLINA LÓPEZ |

# Agradecimientos - Acknowledgements

Cuatro años han pasado desde que llegué a esta universidad y empecé esta tesis. En este tiempo, he tenido la suerte de estar siempre rodeado de personas que me han enseñado y me han hecho crecer, tanto en el campo profesional, como en el personal. Esas personas han aportado un gran significado a este tiempo y son a todas y a cada una de ellas a quien les dedico esta tesis.

En especial, a mis padres y a mi hermano, por su amor, por estar siempre tan cerquita de mí y apoyarme incondicionalmente.

A José Manuel y Jesús, por brindarme la oportunidad de trabajar en este campo, tener la confianza en mí, enseñarme día a día y haber hecho posible este proyecto.

A todos mis compañeros del GIAA, por su inestimable ayuda y por todos los buenísimos e inolvidables ratos que hemos compartido.

A todos mis compañeros de la Universidad Carlos III, porque creo que hemos formado una gran familia.

Y también a todos mis amigos, por su gran apoyo en los buenos y malos momentos.

I would like to thank Massimo, who taught me so many things and gave me so much energy.

To James and Puneet, for such a wonderful time and the opportunity to work and learn so much in Buffalo.

And last but not least, I would like to thank my friends in Sydney and Buffalo, who made me feel so welcome.

# Abstract

This thesis proposes several interconnected objectives for the design of a video-monitoring system whose operation is thought for a wide rank of conditions.

Firstly an evaluation technique of the detector and tracking system is proposed and it is based on a minimum reference or ground-truth. This technique is an answer to the demand of fast and easy adjustment of the system adapting itself to different contexts.

Also, this thesis proposes a technique of optimization based on Evolutionary Strategies and the combination of fitness functions. The objective is to obtain the parameters of adjustment of the detector and tracking system for the best operation in an ample range of possible situations.

Finally, it is proposed the generation of a classifier in which a non-parametric statistic technique models the distribution of data regardless the source generation of such data. Short term detectable activities are chosen that follow a time pattern that can easily be modeled by Hidden Markov Models (HMMs). The proposal consists in a modification of the Baum-Welch algorithm with the purpose of modeling the emission probabilities of the HMM by means of a nonparametric technique based on the density estimation with kernels (KDE).

# Sumario

Esta tesis propone varios objetivos interconectados para el diseño de un sistema de vídeo-vigilancia cuyo funcionamiento es pensado para un amplio rango de condiciones.

Primeramente se propone una métrica de evaluación del detector y sistema de seguimiento basada en una mínima referencia. Dicha técnica es una respuesta a la demanda de ajuste de forma rápida y fácil del sistema adecuándose a distintos entornos.

También se propone una técnica de optimización basada en Estrategias Evolutivas y la combinación de funciones de idoneidad en varios pasos. El objetivo es obtener los parámetros de ajuste del detector y el sistema de seguimiento adecuados para el mejor funcionamiento en una amplia gama de situaciones posibles

Finalmente, se propone la construcción de un clasificador basado en técnicas no paramétricas que pudieran modelar la distribución de datos de entrada independientemente de la fuente de generación de dichos datos. Se escogen actividades detectables a corto plazo que siguen un patrón de tiempo que puede ser fácilmente modelado mediante HMMs. La propuesta consiste en una modificación del algoritmo de Baum-Welch con el fin de modelar las probabilidades de emisión del HMM mediante una técnica no paramétrica basada en estimación de densidad con kernels (KDE).

# Contents

Chapter 1

# Resumen en Español

Los sistemas de vídeo con fines de vigilancia están adquiriendo cada vez más importancia en todo el mundo debido principalmente a su clara utilidad y amplio abanico de aplicaciones tales como realizar tareas de seguridad en lugares públicos como estaciones o aeropuertos, velar por el cumplimiento de las normas de tráfico en carretera, tareas de monitorización en edificios y aparcamientos, etc. Además, el gran esfuerzo realizado por la comunidad científica en los últimos años se ha traducido en un gran desarrollo de la tecnología que permite considerar estos sistemas de vídeo como una opción atractiva y competitiva para las labores de vigilancia.

Tradicionalmente, los sistemas de vídeo-vigilancia actúan como sistemas pasivos en los que son operadores humanos los que deben revisar las secuencias de vídeo y decidir cuándo se produce un comportimiento anómalo que requiere o no su intervención. Así pues, es una necesidad crucial desarrollar sistemas de seguimiento que de forma automática y robusta se encarguen de mandar alertas sobre comportamientos anormales para que las personas adecuadas tomen las acciones correspondientes. Esta automatización de los sistemas de vídeo-vigilancia supone un gran reto de diseño y análisis que conlleva numerosos procesos que son dependientes en gran parte de la aplicación final del sistema.

Dichas aplicaciones son muy variadas, y así, encontramos ejemplos de sistemas de vídeo-vigilancia que se centran en el abandono de objetos o equipajes en lugares como centros comerciales, estaciones de tren , autobuses o aeropuertos [1], [2], la medición del flujo de personas en estaciones [3], la detección de actos vandálicos [4] o el análisis de las actividades llevadas a cabo en la pista de estacionamiento de un aeropuerto, [5] ...

Así pues, los sitemas de vídeo-vigilancia deben ser capaces de funcionar en muy distintos entornos con diferentes blancos (generalmente, objetos o personas en movimiento) y aplicaciones. Esta tesis se centra en las aplicaciones con fines de seguridad en un lugares públicos.

En este capítulo, en primer lugar, veremos una breve introducción a la arquitectura general de los sistemas de vídeo-vigilancia. A continuación, se expondrá una breve explicación de tres de los principales retos de diseño en estos sistemas: Evaluación, ajuste de parámetros y construcción de un classificador efectivo de actividades cuya técnica de construcción sea adaptable a la aplicación en cuestión.

## 1.1 Arquitectura de los Sistemas de Vídeo-Vigilancia

Los sistemas de vídeo-vigilancia en general constan de una o varias cámaras que transmiten su señal a un limitado número de monitores. Los llamados sistemas automáticos o inteligentes deben extraer la información últil capaturada por dichas cámaras e interpretar y entender el comportamiento de los blancos. Este análisis de comprotamiento se realiza de acuerdo a la aplicación final, de tal modo que las alertas y avisos sean generadas de forma rápida.

Para llevar a cabo todo este análisis, la señal de vídeo proporcionada por las cámaras se digitaliza y procesa en varios pasos. Así, si se asume que el sistema está formado por varias cámaras, se distinguen tres niveles lógicos:

- El primero consiste en el análisis e interpretación de una escena por una determinada cámara.
- A continuación, la información analizada y procesada por cada cámara es compartida y fusionada para poder así sacar la máxima información disponible.
- Finalmente, el sistema automático de vídeo-vigilancia distribuye tareas sobre cada cámara de acuerdo a la disponibilidad y habilidad de cada una de éstas para desarrollar una determinada actividad.

Esta tesis se concentra en el primer nivel lógico. Para ello, los procesos incluídos en este nivel suelen ser identificados como una secuencia, en la que la entrada de uno está formada por la salida del anterior y la clasificación normalmente usada es (ver figura 1.1): detección de blancos, clasificación de blancos, seguimiento y reconocimiento de actividades.

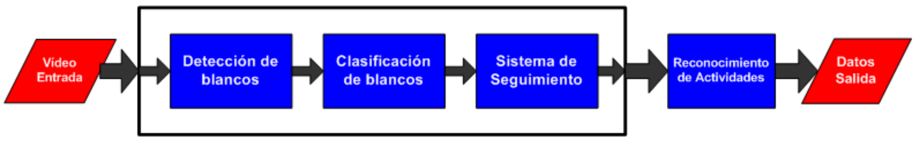


Figura 1.1: Procesos realizados por un cámara con fines de vídeo-vigilancia

Así, brevemente, cada uno de los procesos realiza las siguientes funciones:

- Detección de blancos: Es el proceso encargado de detectar píxeles en movimiento. Los píxeles conectados son normalmente agrupados en areas llamadas "blobs". Existen dos técnicas principales para detectar píxeles en movimiento: ”diferencia temporal” [6] y resta de fondo” [7]. El primero es un método que consiste en la resta de cada fotograma o *frame* con el anterior seguida de una umbralización. La segunda se basa en la construcción de un fondo o modelo de referencia que será restado con el fotograma actual y posteriormente umbralizado.

- Clasificación de blancos: Este módulo es el encargado de reconocer los objetivos de interés a vigilar. Así, puede estar constituido por una serie de máscaras que nos indiquen que solo debemos seguir a determinados objetos en un área o más aún, si la aplicación trata de vigilar personas, cómo reconocer a éstas y obviar el resto de objetos en movimiento.
- Sistema de seguimiento: Este módulo trata dos problemas fundamentales:
  - El problema del movimiento, es decir, predice la posición del objetivo en movimiento o alguna parte de éste para el próximo fotograma
  - El problema de asociación, es decir asociar regiones en movimiento con etiquetas o identificadores para cada fotograma
- Reconocimiento de actividades: Esta última parte se encarga del reconocimiento de actividadess. Estas actividades se definen en función de la aplicación del sistema. Así pues, de acuerdo con el conjunto de actividades, unas u otras medidas son escogidas y posteriormente un tipo u otro de classificador es seleccionado.

El comportamiento final de los procesos de detección y clasificación de blancos y el sistema de seguimiento dependen de unos parámetros de ajuste que harán que el sistema global tenga un funcionamiento con mayor o menor grado de adecuación a la aplicación final del sistema. Estos parámetros son, entre otros, el umbral de detección que discrimina entre píxeles en movimiento o no, un área mínima de blancos a seguir, parámetros de ajuste del sistema de seguimiento, etc.

Además, para ser capaces de medir de forma objetiva y cuantificable el funcionamiento de un sistema de vídeo-vigilancia, es necesario disponer de métricas de evaluación. La siguiente sección nos describe las técnicas de evaluación y la solución propuesta por nuestro sistema.

## 1.2 Evaluación de Sistema de Vídeo-Vigilancia

Para el buen diseño y ajuste de los procesos o módulos del sistema de vídeo-vigilancia (fundamentalmente detector y sistema de seguimiento) es importante contar con una función objetiva de evaluación. Existen multitud de trabajos al respecto, tanto técnicas de evalución a nivel detección (también llamada segmentación) [8], a nivel de segumiento [9] o a ambos niveles [10].

Algunas técnicas sugieren la comparación de resultados directamente con una referencia (o *ground-truth*) que habrá sido extraída por un operador humano que haya marcado las coordenadas donde se encuentran los blancos en cada momento. Los resultados de este tipo de técnicas suelen ser satisfactorios, pero presentan el gran incoveniente de obtener la referencia de antemano, lo cuál suele ser una tarea ardua y dura.

En contraposición, otros trabajos propopen técnicas de evaluación sin referencia. La principal ventaja de estas técnicas radica en que se evita así calcular dicha referencia, usando como tal otros criterios como la similud de la forma del blanco a lo largo del tiempo, la uniformidad en el movimiento o estabilidad en el histograma de color a lo largo de los distintos fotogramas. Pero estas técnicas presentan también grandes incovenientes, entre ellos, su dependencia en la estabilidad del histograma de color, el cuál se pueden ver fácilmente afectado por un cambio de posición del blanco frente a la cámara; o la estabilidad en la forma del blanco, que en el caso de una persona realizando varios tipos de ejercicios puede llevar al sistema a obtener un valor de evaluación erróneo.

Además, existen otras limitaciones importantes a las que los actuales sistemas de evaluación no han respondido:

1. Evaluación de un gran número de vídeos de forma que el resultado de la evaluación refleje estadísticamente el comportamiento del sistema de vídeo-vigilancia.
2. Evaluación en distintos escenarios obteniendo una referencia o *ground-truth* de forma rápida. Esto permitiría versatilidad para evaluar en diferentes situaciones.
3. Evaluación de vídeos independientemente de la tecnología utilizada o el número de blancos a seguir o las actividades que se estén realizando.

Como resultado de estas necesidades y limitaciones, esta tesis propone una nueva técnica para solventar dichos problemas. Esta técnica está basada en una mínima referencia o *ground-truth* $f(x)$ que se define como la aproximación a-priori de la trayectoria seguida por un objeto mediante segmentos de líneas rectas.

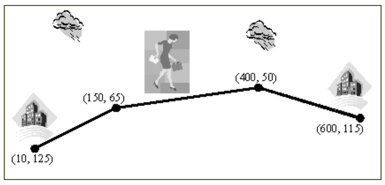


Figura 1.2: Aproximación de una trayectoria mediante segmentos

Así pues, la base de esta evaluación radica en que en múltiples ocasiones, los objetos en movimiento a seguir no se mueven de forma aleatoria y siguen un patrón regular. Por ejemplo, el caso de personas andando por aceras o aviones en una pista de aterrizaje.

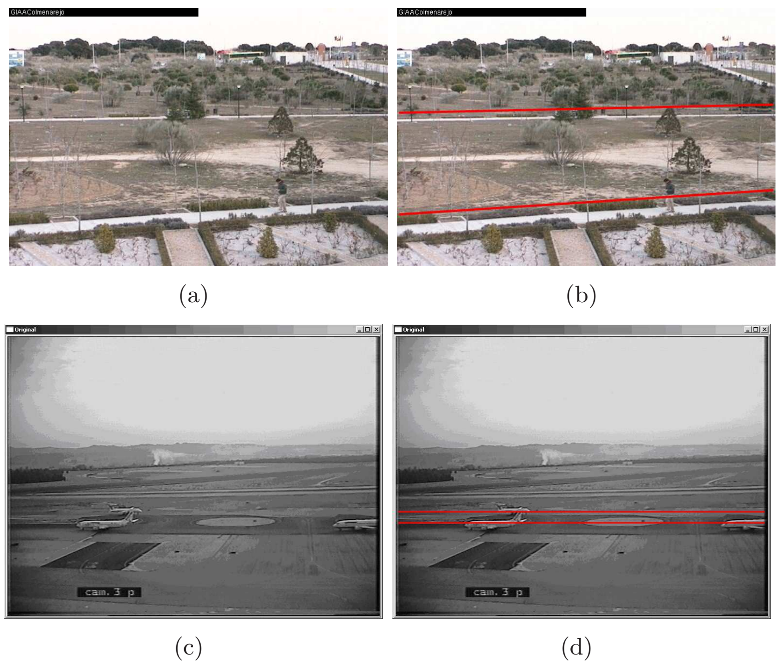


Figura 1.3: Ejemplos de objetos que siguen caminos regulares que pueden ser definidos por segmentos

Cada vez que una pista es inicializada o actualizada, el sistema considera como referencia el segmento con menor distancia a dicha pista. A partir de entonces, las métricas de evaluación son calculadas para cada pista; es decir, esta función de evaluación $e_{ij}$ se define por cada pista $i$ para un determinado escenario $j$. Dichas métricas se dividen en:

- Métricas de precisión: calculan estimaciones y comparativas entre las pistas actuales y las existentes en el fotograma anterior, además de calcular desviaciones respecto al comportamiento ideal (medido con referencia al *ground truth*).
- Métricas de robustez: estiman los fallos de continuidad del sistema de detección y seguimiento.

Estas métricas se diseñan partiendo de una técnica de evaluación con *ground-truth* o referencia propuesta también en esta tesis (sección 4.2). Una vez comprobada la validez de esta técnica con referencia, las métricas se extrapolan y adaptan para al caso que nos ocupa de mínima referencia. Para más información ver secciones 4.2 y 4.3.

Con el fin de comprobar la efectividad de esta nueva técnica de mínima referencia se mide la sensibilidad de las métricas frente a un empeoramiento en la detección y el seguimiento para unos determinados ejemplos de personas andando por aceras al aire libre (ver figuras 1.3(a) and 1.3(b)). En concreto, se evalúan 100 vídeos divididos en dos tandas de 50 vídeos cada una. Las métricas muestran el cambio y son capaces pues de reflejar y cuantificar el peor funcionamineto del sistema.

## 1.3 Optimización

Una vez que ya conocemos de forma objetiva y cuantitativa la evaluación $e_{ij}$ para cada pista $i$ en un escenario $j$, el siguiente paso es elegir los párametros de ajuste y encontrar los valores óptimos para un funcionamineto óptimo del sistema.

Las técnicas de *Soft Computing* resultan atractivas para este problema, ya que éstas se aprovechan de la tolerancia a la imprecisión y la incertidumbre con el fin de obtener una solución robusta [11]. Algunos ejemplos son la forma en que los humanos son capaces de entender el lenguaje hablado incluso si éste no es claro o leer cuando la escritura es confusa. Por consiguiente, estas técnicas usan la mente humana o la naturaleza en sí como modelo y tratan de formalizar matemáticamente el razonamiento o proceso seguido por estos dos ejemplos. En nuestra caso, una forma de interpretar esta tolerancia a la incertidumbre se puede traducir en que distintos ajustes de los parámetros pueden llevar a soluciones correctas. Las técnicas de *Soft Computing* están formadas por un grupo no homogéneo en el que se incluyen: lógica borrosa, neuro-computación, computación evolutiva y computación probalística [12]. Estas técnicas son complementarias más que competitivas entre sí.

En este problema, estamos interesados en una técnica de optimización para el ajuste de unos parámetros, que en el caso de las técnicas de *Soft Computing* viene dado por la computación evolutiva. La computación evolutiva resulta una alternativa apropiada para este problema ya que el conjunto de variables a optimizar combina variables continuas y discontinuas, es un problema con ruido y además admite varias soluciones. En concreto en este trabajo se utilizan las Estrategias Evolutivas [13] ya que están diseñadas para trabajar con números reales y pasos adaptables a medida que el algoritmo converge.

El objetivo de la optimización es encontrar el juego de valores de parámetros que haga que el funcionamiento del sistema sea óptimo. En este trabajo se pretende ir más allá de la optimización de un vídeo concreto, y se propone una técnica de diseño que ajuste dichos parámetros para distintas condiciones y escenarios de trabajo, siempre dentro de unos límites y un entorno específico. Por condiciones distintas se entiende, por ejemplo, cambios de luz o de condiciones atmosféricas, mientras que por distintos escenarios entendemos variabilidad en campo de visión de la cámara dentro de un mismo entorno (una estación de trenes, vistas de pistas en un aeropuerto, etc.).

La técnica de generalización consiste en construir una función de idoneidad o *fitness* para las Estrategias Evolutivas que está formada por la agregación de resultados individuales de evaluación por blanco y en cómo escoger una serie de escenarios de entrenamiento que definan lo mejor posible las condiciones de funcionamiento en el entorno de trabajo. Dichas agregaciones serán la suma (o el máximo) de funciones de evaluación por blanco o pista a lo larga de todos los objetos considerados en un escenario.

Para probar la efectividad de dicha técnica de generalización se propone usar un sistema de seguimiento flexible y sencillo basado en reglas y comprobar si los parámetros ajustados con la técnica de optimización propuesta y el sistema de seguimiento basado en reglas dan un funcionamiento mejor que si otros sistemas de seguimiento más específicos y matemáticamente más complejos hubieran sido utilizados.

Los experimentos llevados a cabo usando esta técnica de optimización en un entorno aeroportuario donde los objetos de movimiento son aviones y vehículos de mantenimiento en pista muestran cómo la técnica de optimización-generalización es capaz de encontrar juegos de parámetros que hacen que el funcionamiento del sistema de seguimiento de reglas sea superior a la mostrada por otros cinco sistemas de seguimiento conocidos y algunos, computacionalmente más complejos que el basado en reglas: CC (Connected Component Tracking [14]), MS (Mean Shift Tracking [15]), MSPF (Particle Filter based on MS weight [16]), CCMSPF (Connected Component tracking and MSPF resolver for collision [14]) y CGA (Association by Canonical Genetic Algorithm [17]).

Los experimentos demuestran en la sección 5.3 que los valores de los parámetros de la técnica de optimización dan un funcionamiento estable tanto para casos sencillos (como es el vídeo 2 dónde solo un par de aeronaves aparecen en escena), como para casos complejos (vídeo 1 y 3, donde vehículos de mantenimiento pequeños y autobuses aparecen con aeronaves). Por contra, los sistemas de seguimiento más complejos muestras un buen comportamiento ante casos sencillos y más deficiente en los complejos.

## 1.4 Reconocimiento de Actividades

Finalmente, una vez que la detección y el seguimiento han sido ajustados de forma robusta, el problema del reconocimiento de actividades o interpretación de la escena surge de manera natural.

En general, existen tres grandes puntos a tener en cuenta para el reconocimiento de actividades:

- Identificación del tipo de actividades a clasificar: actividades detectables a corto plazo (por ejemplo andar o correr), a largo plazo (mirar escaparates), con un determinado patrón de tiempo (sentarse o levantarse), etc. A raíz del tipo de actividades se elige el tipo de medidas más acorde para la clasificación.

- Selección de medidas: Es un paso esencial en el que las medidas más caracterísicas son extraídas de las secuencias de vídeo. Por ejemplo, si una de las tareas es distinguir entre saludar con la mano en alto y hacer salto con palmada sería importante medir la posición de las manos en cada instante.

- Construcción de un clasificador: Este paso está íntimamente relacionado con los dos anteriores. El construir un clasificador puede ser visto como aprender una asignación entre un conjunto de datos $X$ (siendo $X$ un conjunto de $n$ puntos, donde $x_i \in X$ para todo $i \in [n] := 1, .., n$) y un conjunto discreto de etiquetas o clases $Y$. Matthias Seeger dice en [18] que *el aprendizaje de un conjuno de datos se puede ver como un intento de comprimir drásticamente datos sin perder información inherente*. Estas técnicas de aprendizaje pueden ser clasificadas en tres categorías dependiendo del número de datos etiquetados:

  – Aprendizaje supervisado: las actividades han sido previamente definidas y la clasificación debe encajar con estas especificaciones. El objetivo es aprender una asignación de $X$ a $Y$ dado un conjunto de entrenamiento de pares $(x_i, y_i)$. Aquí, las $y_i$ son llamadas las etiquetas del conjunto de muestras $x_i$. Ejemplos de estos clasificadores son las redes de neuronas, los árboles de decisión, lás máquinas de soporte de vectores (SVMs), etc. Estas técnicas son útiles cuando las clases son definidas previamente y tenemos un nutrido grupo de datos etiquetados.

  – Aprendizaje no-supervisado: Las actividades son inferidas al agrupar los datos $X$ que en este caso están sin etiquetar. En el aprendizaje no supervisado se asume que las observaciones son causadas por variables latentes y los clasificadores de aprendizaje no supervisado deben aprender a representar los datos de entrada de tal modo que relejen la estructura estadística de producción de dichos datos. Ejemplos de classificadores no supervisados son los métodos de agrupamiento o *clustering*, como por ejemplo el de la maximización de la esperanza (EM o *Expectation Maximization*). Estas técnicas son útiles cuando queremos inferir clases a partir de unos datos o cuando no disponemos de datos etiquetados.

  – Aprendizaje semi-supervisado: el clasificador se genera con datos etiquetados y no etiquetados. Normalmente, se tiende a tener un reducido número de datos etiquetados frente a un alto número de no etiquetados. Lo más común es realizar una primera etapa de aprendizaje no supervisado con los datos no etiquetados. Posteriormente se etiquetan las clases con ayuda de los datos etiquetados y algún criterio matemático o heurístico (como por ejemplo, etiquetar la clase dependiendo del mayor número de muestras de una etiqueta que haya en un clúster) y se usan dichos datos etiquetados para obtener una medida de error. En particular, este trabajo analiza los datos de una base de datos pública llamada CAVIAR [19] en la que los datos etiquetados representan una minoría frente a los no etiquetados, de ahí que esta tesis use aprendizaje semi-supervisado para construir un clasificador.

Además, una importante distinción entre todos estos métodos radica en el uso o no de aprendizaje estadístico. La principal ventaja de utilizar conceptos probabilísticos es que las probabilidades de error o acierto pueden caracterizarse como resultado del modelo.

De nuevo, y siguiendo el argumento de generalización y robustez de este trabajo, esta tesis propone la construcción de un clasificador capaz de modelar y classificar actividades cubriendo el mayor número de situaciones; esto es, el clasificador debe modelar las medidas seleccionadas por medio de un modelo probabilístico independientemente de la forma de la función de densidad de probabilidad que dichos datos definan. Estas medidas y sus correspondientes computaciones normalmente están generadas por funciones difícil de aproximar por densidades de estimación comunes como las Gaussianas.

De esta manera, lo primera acción será elegir un tipo de clasificador y más tarde decidir cómo modelar los datos probabilísticamente.

En este respecto, muchos trabajos han sido desarrollados en los últimos años [20]. Particularmente relevantes son los clasificadores que toman en cuenta las secuencia temporal de las actividades. Por ejemplo, si una persona está inactiva en un fotograma, es muy poco probable que en el siguiente esté corriendo. Pasará por una serie de fotogramas en que la persona se empieza a mover y habrá un transitorio entre inactivo a moviéndose y posteriormente a corriendo.

Especialmente destacados son los trabajos de Modelos de Markov Ocultos (*Hidden Markov Models (HMMs)* [21] que han probado ser eficientes para este modelaje temporal [22]. El uso de los HMMs se divide en dos pasos: entrenamiento y validación.

Es muy común modelar estadistícamente la probabilidades de emisión de los HMMs mediante técnicas paramétricas, especialmente con mezcla de funciones y en particular con mezcla de Gaussianas. La estimación de los parámetros en este caso se lleva a cabo mediante un algoritmo de maximización de la esperanza (EM) llamado Baum-Welch [23].

Sin embargo, la mezcla de Gaussianas presenta dos limitaciones importantes:

- los datos observados son difíciles de modelar con estas funciones de densidad de probabilidad, bien porque los datos no han sido generados por funciones Gaussians, bien porque el número de modos escogido no se ajusta a la realidad.
- la variabilidad de los resultados dependiente del conjunto de inicial de parámetros.

Por consiguiente, el autor propone modelar las probabilidades de emisión de los HMMs por medio de un modelo de estimación de densidad con kernels (KDE o *Kernel Density Estimation*) [24], una técnica no paramétrica que será acoplada al algoritmo de Baum-Welch (para lo cuál tendremos que modificar dicho algoritmo) y modelará eficientemente las medidas de manera que se superen las limitaciones de la tradicional mezcla de Gaussianas o mezcla de funciones en general.

La estimación de densidad con kernel (KDE) construye una estimación de la subyacente función de densidad de probabilidad no observable sin asumir conocimiento previo de la distribución de los datos. La función de densidad oculta se construye a partir de un gran número de datos centrando una función de densidad de probabilidad llamada kernel con ancho de banda $h$ en cada uno de estos datos. Así, el kernel es definido como una función de peso $K()$ que puede ser cualquier función de probabilidad: uniforme, triangular, Gaussiana, cosenoidal, etc.

Lo que hace a KDE un modelo adecuado y adaptable para la construcción de clasificadores es que es una técnica que proporciona una representación precisa de los datos independientemente de la forma que éstos tengan. Por la misma razón, KDE no es tan sensible a la inicialización.

Los experimentos realizados y que se muestran en la sección 6.4 usan los vídeos de la base de datos pública de CAVIAR con el fin de clasificar actividades a corto plazo: inactivo (IN), andando (WK) y corriendo (R), tal y como se hace en dicha base de datos.

En una primer paso se escogen las medidas que van a ser utilizadas como entrada en los clasificadores. Tras la aplicación de una serie de filtrados (ver sección 6.4.1) se deciden escoger dos tipos de velocidades calculadas a partir de la resta entre la posición del centro de masas de las personas en 5 y 25 fotogramas tal y como muestra a continuación:

$$speed_5 = \frac{1}{5}\sqrt{(x_i - x_{i-5})^2 + (y_i - y_{i-5})^2} \tag{1.1}$$

$$speed_{25} = \frac{1}{25}\sqrt{(x_i - x_{i-25})^2 + (y_i - y_{i-25})^2} \tag{1.2}$$

Posteriormente, el aprendizaje de los clasificadores se lleva a cabo en dos fases:

- En una primera tanda se realizan los experimentos con clasificadores conocidos: un árbol de decisión (J.48), dos algortimos Bayesianos (Bayes Net y Naive Bayes), un algoritmo basado en un sistema neuronal y borroso (Neuro-fuzzy) y un algoritmo basado en reglas (PART).
- A continuación, se llevan a cabo los experimentos para los HMMs modelados con mezcla de Gaussianas y con nuestra aproximación de KDE para varios valores de inicialización.

Los resultados mostrados en la sección 6.4.3 reflejan un funcionamiento superior para los clasificadores basados en HMMs. Así, los clasificadores basados en HMMs tienen un error de clasificación (en la fase de validación) comprendido entre un 15 y un 17.4 % aproximadamente. Por el contrario, el resto de los clasificadores muestra un sobreajuste, al tener un error bajo para la etapa de entrenamiento y muy alto para la de validación (un minímo de 22 % de error de clasificación y un máximo de un 60 % para el clasificador neuro-fuzzy).

Si comparamos entre HMMs, se comprueba que el KDE-HMM proporciona una salida más estable independientemente de la inicialización de los parámetros (entre el 16 y el 16.45 %) frente al HMM-GM modelando con mezcla de Guassianas (entre el 15.07 y el 17.32 %). Sin embargo, cabría hacer un estudio futuro más en profundidad para comprobar si esta diferencia de resultados se debe al mejor funcionamiento del clasificador propuesto o a la variabilidad de los datos debido al ruido inherente que éstos siempre accarean.

## 1.5 Conclusiones y Futuro Trabajo

Esta tesis propone varios objetivos interconectados para el diseño de un sistema de vídeo-vigilancia cuyo funcionamiento es concebido para un amplio rango de condiciones en un determinado entorno.

En primer lugar, se proponen dos técnicas de evaluación para el detector y el sistema de seguimiento de tal modo que se consiga una evaluación objetiva de dichos módulos. La primera técnica es una técnica que usa *ground-truth* o referencia y se usa como base para extender sus métricas al caso de mínima referencia. La métrica de mínima referencia se presenta como respuesta a la demanda de ajustar el sistema de seguimiento de forma rápida y fácil adecuándose a distintas localizaciones de dicho sistema. Este ajuste necesita de vídeos de los que extraer la mínima referencia, la cúal no es más que una trayectoria que siguen vehículos y personas en multitud de ejemplos (vehículos en una carretera, personas en una acera, etc.). Esta trayectoria o *ground-truth* se aproxima por una unión de segmentos a partir de los cuáles se definen las métricas. Los experimentos llevados a cabo en la sección 4.4 demuestran que las métricas de evaluación con mínima referencia reflejan objetivamente el comportamiento del detector y el sistema de seguimiento con la ventaja de simplificar enormemente la obtención de las referencias.

Las principales limitaciones a esta técnica de mínima referencia son la restricción de definir la referencia como el camino regular que siguen los objetos en movimiento. Así pues, esta técnica solo puede aplicarse en escenarios donde los objetos sigan este tipo de trayectorias.

En segundo lugar, esta tesis propone una técnica de optimización basada en Estrategias Evolutivas, la selección de un grupo de vídeos de entrenamiento que representen el entorno en el cual el sistema va a realizar sus funciones y la combinación de funciones de idoneidad o *fitness* en varios pasos. El objetivo de esta optimización es obtener los valores de los parámetros de ajuste del detector y el sistema de seguimiento adecuados para el mejor funcionamiento en una amplia gama de situaciones posibles. La estrategia para demostrar la validez de esta técnica se basa en la optimización de un detector flexible basado en reglas frente a unos sistemas de seguimiento conocidos y matemáticamente más complejos.

Los experimentos llevados a cabo en la sección 5.3 muestran que la técnica de optimización es capaz de ajustar los valores de los parámetros para múltiples escenarios dentro del mismo contexto y objener un funcionamiento más estable que el mostrado por otros sistemas más complejos.

Finalmente, este trabajo se marcó como objetivo la construcción de un clasificador basado en técnicas no paramétricas que pudieran modelar la distribución de datos de entrada independientemente de la fuente de generación de dichos datos. Además, esta tesis se centra en las actividades de corto plazo que siguen un patrón de tiempo que puede ser fácilmente modelado mediante HMMs. La propuesta de esta tesis consiste en una modificación del algoritmo de Baum-Welch con el fin de modelar las probabilidades de emisión del HMM mediante la técnica no parámetrica basada en estimación de densidad con kernels (KDE).

El resultado de los experimentos (sección 6.4) para una serie de vídeos incluídos en la base de datos de CAVIAR [19] muestra resultados estables independientemente de la inicialización del modelo para el modelo HMM-KDE. Si se compara con el HMM-GM, los resultados no son espectacularmente mejores, pero como dijimos anteriormente sí estables. La comparación con otros clasificadores clásicos (J48, Bayes Net, Naive Bayes, PART y Neuro-fuzzy) muestra que el modelo propuesto supera a todos ellos en clasificación.

El mayor incoveniente de este clasificador es su pobre escalabilidad ya que a más variedad de medidas, mayor dimensionalidad, lo que conlleva una generación del clasificador lenta y tediosa.

El objetivo de obtener un sistema de vídeo-vigilancia robusto fue conseguido gracias a la técnica de optimización y a la generación de un clasificador capaz de modelar distribuciones difíciles de datos. Además, el sistema fue entrenado y validado bajo una amplia gama de escenarios, mostrando un buen comportamiento en todos los casos.

Finalmente, la validación de cada uno de las propuestas de esta tesis se ha llevado a cabo mediante experimentos particulares para cada uno de los objetivos. Por consiguiente, el trabajo futuro más inmediato consistirá en la elaboración de un experimento en el que todas las técnicas sean probadas de forma conjunta.

# Chapter 2

# Introduction

Surveillance systems are becoming more important every day all over the world since they deal with security applications designed to avoid catastrophes and terrorist attacks or perform the daily safety duties in office buildings, stations, roads or streets.

Moreover, the deep study in image processing and the rapid development of visual technology have resulted in visual surveillance systems being considered as the best option for surveillance.

In the past, video surveillance systems relied on human operators who might not detect all the events since they had to control a big amount of cameras and might get tired or distracted after some time.

Thus, it is crucial to develop robust and reliable automatic visual surveillance systems that warn of suspicious behaviors in order to take the corresponding actions. These systems allow us to have plenty of information that must be studied, processed, computed and filtered so that the output of the system is based on the specific warnings and alarms that we want to obtain depending on the application.

There are many applications where video surveillance systems are required. This section intends to give a brief overview of some of them in order to offer a clear idea of the multiple applications where surveillance systems can be used.

For example, some works focus on detecting abandoned objects or luggage in places like shopping malls, railway and bus stations, airports or universities. In [1], a monochromatic camera is recording in the waiting room of unattended railway stations. Constantly, when an abandoned object is detected, an alarm is sent via code-division multiple-access (DS/CDMA) to a control center placed a few miles away from the unattended stations. This center centralizes and displays all the alarms generated in the unattended railway stations. Then, a human operator decides if the signal is a false or real alarm. In addition, in [2], the authors present a network of visual sensor to monitor sensitive areas also in the area of public transportation premises. The system detects and reports suspicious activities like theft, left bags, loitering, etc.

A good example of several applications of surveillance systems in transportation areas is the book "Advanced Video-Based Surveillance Systems " published in 1999 [4]. This book has some works applied to security in ports, highway traffic monitoring, prevention of vandalism in metro stations, etc. In particular, the European CROMATICA (Crowd Management with Telematic Imaging and Communication Assistance) project [3] addresses to measure the flow of people in metro stations in order to detect abnormal behaviors such as overcrowding, vandalism, people that might fall on the tracks, etc. In this line, another European project, AVS–PV (Advanced Video Surveillance–Prevention), detects possible vandalism in metro stations [25] measuring the time that a person remains on the platform or in the same place without taking a train.

Another remarkable application is the one presented by [26] in which the system is capable of detecting objects and subsequently analyzing if these objects are being stolen. First of all, a novel kernel-based tracking system is proposed for tracking and obtaining the trajectory of each object. Then, the system includes a module that checks if the condition of "a person $A$ approaching a person $B$, transferring an object between them and then a person $A$ leaving" is accomplished. The authors use mixture of Gaussians to model the suspicious subjects' visual properties and to clearly segment the transfered object.

Other applications analyze human behavior and classify it among several activities. For example, in [27] the authors distinguish among punching, hand-shaking, pushing, or hugging. The system proposed analyzed the image on three levels: pixel level, blob level and semantic level. In addition, there have been many applications in the area of sports. Thus, we can find some applications to track soccer players in order to obtain useful information that can improve the performance of the player by using better planning [28].

But not all the applications have to do with people and the interpretation of their activities as targets . For example, the work in [29] presents a multi-camera system for tracking moving vehicles and people, and for carrying out scene interpretation in airport aprons (a person or a vehicle has appeared in the area of interest, a person is close to a vehicle, a vehicle is remaining for a long period of time, two vehicles are really close, etc.). Other applications built systems in charge of tracking and classifying the traffic on a road [5], the surface traffic on the taxiways of an airport [30], etc.

Therefore, video surveillance systems must be able to work in very different environments, with different types of targets and applications. The main interests of the author are applications for security in public places that incorporate some logic for the human behavior understanding.

## 2.1 Video Surveillance Architecture

Video surveillance systems consist of one or several cameras that transmit their signal to a limited number of monitors. The intelligent or automatic video surveillance systems must extract the useful information captured from the cameras and interpret the behavior of the objects according to the final application in order to rapidly generate alerts and warnings upon detecting suspicious or abnormal behaviors. To do that, the video streaming provided by the cameras is digitalized and subsequently processed to obtain the data in which the application is interested. Usually, these processes may include several stages: environment modeling, moving objects detection, moving objects classification, tracking, action recognition and, finally, multiple-camera information fusion.

Therefore, if it is assumed that surveillance systems are formed by a group of cameras, then, three different operational levels of logic can be distinguished:

- First, the scene analysis and interpretation from each sensor is enacted by a single camera.
- Second, the information analyzed and processed by each single local camera is fused in order to obtain the maximum information available.
- Finally, the surveillance process is distributed over several cameras, according to their individual ability to contribute their local information to a global solution.

For example, let's assume that the application of several cameras is simply to detect individuals in a room in which there are multiple obstacles (columns, furniture, etcetera). In the first stage, each camera will individually process the information on the people in the room depending on each camera's field of view. The second stage will fuse the information of all the cameras, matching the different views provided by each camera and obtaining a final set of tracks or individuals moving in the room. In this stage, a key problem is the non-duplication of targets due to the different view provided by different cameras. Thus, for example, if a camera is tracking a person that in the next seconds is going to be hidden by a column, another camera in the room will detect and identify that person as the one detected before. This distribution of tasks constitutes the third logical level.

This thesis focuses on the first level of logic; that is, in all the processes that a single camera of the surveillance system has to carry out in order to extract the maximum information derived from the video. These processes can be identified as a sequence, where the input of each process is formed by the output of the previous one. The set of processes of a single camera problem can be depicted as the following sequence: object detection (or low-level vision), object classification, tracking (or intermediate-level vision) and activity recognition (or high-level vision).

Thus, the functionality of each process can be briefly explained as follows:

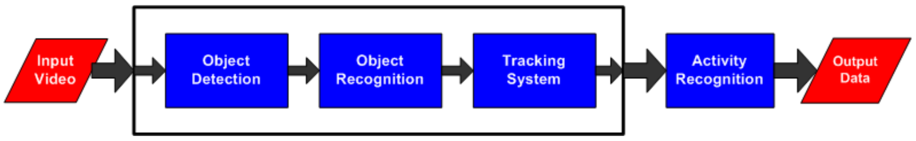


Figure 2.1: Processes for a single surveillance camera

1. Object detection or Segmentation: The aim of this process is the detection of moving pixels. The connected pixels are commonly grouped into compact areas called ”blobs”. There are two main conventional approaches to moving pixel detection: ”temporal difference” [6] and ”background subtraction” [7]. The first approach consists of the subtraction of two consecutive frames followed by thresholding. The second technique is based on the subtraction of a background or reference model and the current image followed by a thresholding and labeling process. After using either of these approaches, morphological operations are typically applied to reduce the noise of the image difference.

2. Object recognition module: The main task is the recognition of objects of interest [31], that is, different moving targets must be considered in a different way in order to further track and analyze their behaviors in correctly. For example, if the application is interested in people, the system must discard waving trees, vehicles or other kinds of moving objects. Therefore, this module uses model-based approaches to create constraints in the object appearance model, e.g. the constraint that people appear upright and in contact with the ground. The object recognition task then becomes a process of using model-based techniques in an attempt to exploit this knowledge.

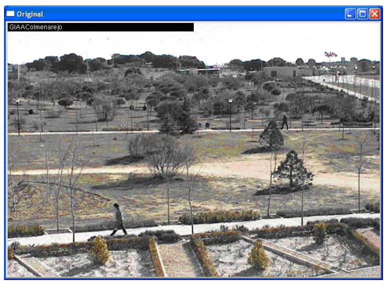
3. Tracking module: This module generally addresses two problems [32]:

   - The ”motion problem” calculates a prediction of the position of the moving object or part of it in the next frame.
   - The ”association or matching problem” associates moving regions with labels or identifiers for each frame by taking into account the appearances and existing tracks in the field of view (FOV).

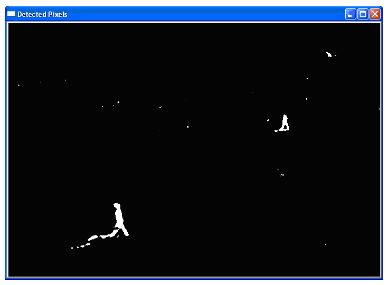
   One of the most common solutions for the motion problem is the Kalman filter [33], [34], although other methods have been proposed, i.e. HMMs (hidden Markov models) [35], Particle Filters, etc.

4. Activity recognition process: the final module is in charge of recognizing activities and behaviors. These activities will be defined depending on the application of the system (i.e. luggage abandoned in a station, people fighting in a public building, etc). Therefore, according to the set of activities, one or other measures are extracted from the previous modules to be subsequently transformed and classified. There are several approaches to this classification: Dynamic time warping (DTW) [36], [37], HMM (hidden Markov models), Bayesian networks [38], [39] or declarative models [40] among others.

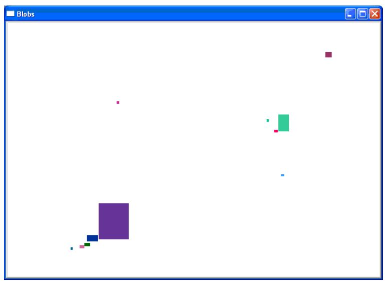
The object detection, object recognition and tracking system processes (low and intermediate vision, see figure 2.2) have some parameters to adjust in order to obtain the best performance of the system. For example, this set of parameters can be formed by a threshold that decides which pixels are moving or stationary, a minimum value for the area of the blob (group of connected moving pixels) so that noise can be removed and not treated as a moving object, a minimum density value to group the blobs and matching them with a track, a minimum value for the area of a target (track), etc.



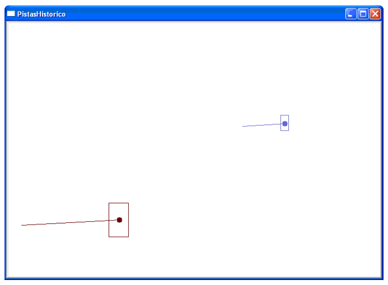
(a)



(b)



(c)



(d)

Figure 2.2: Information levels in the processing chain: a) Raw Image, b) Detected Pixels, c) Extracted blobs and d) Estimated Tracks

These parameters and their values will depend on the application and the environment where the cameras are set. For example, the minimum area or shape restrictions to consider a moving object as a track are not the same if the final purpose of the system is to track aircraft or people, or if the camera is set at a long distance from the targets or in dark places, etc. Moreover, the design of the system must take into account that the system must work under changing conditions: weather, lights, number of targets, wind, etc.

If these processes are adjusted correctly, they will provide precise and reliable data to the high level vision stage (activity recognition), which will produce description of actions in an automatic way. This is of vital importance, as the final goal of all surveillance systems to provide robust information in an automatic way requires a high level of accuracy [41]. Moreover, due to the sequential nature of the processes, it is crucial to solve the failures as soon as they appear. For example, if a person is not detected in the first stage, it is impossible to track and analyze his/her behavior in subsequent steps.

Thus, the next section will explain the evaluation process for tracking systems in order to know objectively what good or bad performance is.

## 2.2 Evaluation

For the good design of these modules (object detector and tracking system), it is important to have an objective function that measures the performance of the system. There are numerous papers on performance evaluation functions for object detection or segmentation [8], tracking [9] or both [10].

Some of the work carried out in this field suggests performance metrics based on the direct comparison of the results of the tracking systems against an ideal ground-truth obtained in an off line process. These techniques give good results in general, but the main drawback lies in the difficulty and tedious work of extracting the exact position of each moving target in every frame.

Other work proposes an evaluation function without ground truth at all. The main advantage of these techniques is that they avoid calculating the ground-truth extraction by using other evaluation criteria such as track shape similarity, track temporal stability, motion uniformity, inter-frame color histogram differencing, etc. But these techniques also present several limitations. For example, some are strongly based on color histograms and in many occasions, the video sequences are recorded in black and white. Or, more importantly, the target can change position with respect to the camera; thus, the colors of the target captured for evaluation and comparison might be very different than the ones stored from previous frames. Another limitation is due to the metrics based on the target shape similarity between frames. The target can be performing different exercises or movements, which can be interpreted by the evaluation system as low performance of the detector and tracker.

Moreover, the author found several limitations that the current evaluation techniques could not solve:

1. Evaluating a high number of video sequences or long sequences recorded during long periods. That is, the author is interested in an evaluation function capable of evaluating not only one video, but many, providing certain metrics that reflect the behavior of the total set of video sequences.

2. Evaluating very different scenarios (versatility) by rapidly obtaining a ground truth dataset.

3. Evaluating video sequences regardless of the color technology of the camera, type or number of targets, or activities they are performing.

As a result, this thesis proposes a new quantitative performance evaluation function based on minimum ground truth for the assessment of the performance of the surveillance system. The function is formed by several metrics that evaluate object detection and tracking after having matched the ground truth with the tracking objects in many video sequences. As a result of this, we can evaluate different scenarios in a fast and easy way.

We can then easily checked that certain changes in the set of parameters produce significant changes in the performance results. In order to do this, the author is interested in obtaining the best set of parameters for the proposed scenarios and problems. For example, if the minimum blob area value decreases, then small waving trees or noise spots are classified as potential blobs to associate to tracks. Or, if a margin distance is set to preclude the appearance of new tracks around an existing track and this margin distance is set to a very high value, the system will avoid the creation of new tracks in the ”margin” surroundings of an existing track.

It is important, then, to select the proper set of parameters depending on the scenario and application. This selection can be made by the trial-and-error method (trial-and-evaluation in this case), but it will be a tedious and almost impossible task. Optimization techniques can be used to solve the necessity of obtaining the optimal parameters for certain specifications. In this particular case, the specifications will be provided by the good performance of the system given by the evaluation metrics.

The next section introduces the optimization techniques used in this thesis.

## 2.3 Optimization

In order to select the proper value of the parameters that tune each of the tracking processes, the optimization theory offers a wide variety of methods: statistical optimization, gradient based methods, evolutionary algorithms, etc. In this particular problem, the high, non-linear dependency among the modules and the sensitivity of the parameters make the optimization of this problem a difficult task for the classical techniques in which computing time increases as a function of precision.

This work explores the soft-computing techniques [12] which are formed by a non-homogeneous group of techniques and concepts, not only optimization techniques.

Soft computing techniques have as a main aim *exploiting the tolerance of imprecision and uncertainty to achieve tractability, robustness, and low cost solutions* [11]. This is the kind of ability that humans use to understand human speech, read non clear handwriting, etc. Thus, soft computing techniques use the human mind as a model and try to mathematically formalize the reasoning processes that humans follow to take decisions on a daily basis.

A good example, which Lotfi A. Zadeh explains in his work [11] is the problem of parking a car. Almost everybody is capable of parking because the final position of the vehicle and its orientation are not specified precisely. If they were, the difficulty of parking would increase geometrically with the grade of precision. The important point in this example is that the problem is easy for humans to solve whereas it is very difficult to solve by classical techniques. This is the argument that the author applies to the current problem of optimizing the parameters of a surveillance system. That is, the result must be robust and reliable, but we do not have to impose precise specifications; we just have to to be able to detect and track people. Soft computing techniques include fuzzy logic, neuro-computing, evolutionary computing and probabilistic computing. A remarkable quality of soft computing is that different methods are complementary rather than competitive and exclusive [42]. Therefore, evolutionary computation is more appropriate for this problem of optimization, since it is a systematic random search to find an optimal solution to a problem. The author has chosen Evolutionary Strategies [13] for this problem as they are designed for working with real numbers and adaptable steps while the algorithm converges to a solution.

This thesis then aims to solve the problem of obtaining optimized parameters values for the good performance of all the parts of the interrelated blocks, not only in a particular case, but also for more general solutions. As a result, one of the main novelties of this thesis consists in addressing the optimization of the parameter values of the surveillance system in order to make it as versatile as possible. That is, this work proposes a tool or design technique for adjusting the values of the parameters that adjust the tracking process for certain variable conditions.

That means that the system must be able to work under different conditions and scenarios. In this context, *different conditions* means changing weather or lighting conditions, variability in the number of people or how they appear in the scene (in groups, running, holding objects, etc), unstable background, etc. On the other hand, *different scenarios* means different locations of the camera in the same type of environments: for example, tracking people in the premises of a building and moving the field of view of the camera or just changing the camera from one location to another.

The results will be useful when the system has a single camera and it is used in multiple locations (or fields of view) according to necessities, or when the system has several cameras that must be tuned in quickly by using only one process to tune the parameters, etc.

Thus, this work proposes a design technique for optimization and generalization that allows users to find the most suitable set of parameters (optimization) for the surveillance system in different scenarios (generalization), for the best performance of the system. Because of this, the training videos must be carefully selected so that there is enough difference among them to avoid over-fitting. At the same time, the training videos must share the same environment and have common classes of targets. The generalization methodology consists of minimizing several combinations of the evaluation function of each track by means of the Evolutionary Strategies in progressive steps in order to gradually build a more general fitness function and parameters.

The strategy that the author proposes to prove the validation of this technique is to optimize generalizing a simple tracker based on *Point Tracking* and certain rules for the association problem. As the author explained before, in this particular case, the "Rules" tracker is used in this work as a recursive Kalman filter [43] that updates the centroid position, rectangle bounds and velocity of each moving object. There are more sophisticated trackers, which could solve the challenging problems of changing conditions or people holding objects, etc. For example, there are trackers that use templates and density-based appearance models [44], multiview appearance models [45], silhouettes [46], etc. Nevertheless, the objective of this work is to find out if performance of the Rules tracker after having applied the proposed optimization/generalization methodology surpassed the performance obtained by more complex trackers.

## 2.4 Action Recognition: Learning Activities

Finally, once the object detection and tracking processes have been reliably adjusted, the problem of activity recognition or scene interpretation follows naturally. This work is going to concentrate on behavior understanding, concretely on distinguishing different activities carried out by pedestrians in video sequences.

In general, and in this thesis in particular, there are three key aspects to activity recognition:

- Identification of type of activities that must be classified: short term activities (i.e. walking, running, fighting), long term (i.e. browsing), activities with a clear time pattern (i.e. sitting down, standing up), etc. A study of what kind of features define each activity must be carried out in order to extract these features, as the next point explains.
- Proper feature selection depends on the type of activity to be recognized. This first step consists in the extraction of measurements from the previous stages and their subsequent computation in order to obtain relevant, remarkable and representative features to clearly distinguish the activities to be recognized. For example, if the interest is in distinguishing waving from doing jumping jacks, it would be interesting to track the position of the hands at each instant or the evolution of the silhouette. Supposing the activity consists in detecting a person that lies on the ground; it would be important to measure the time the person remains in the scene and some body features to match the detections with a human body.
- Classifier. This step is crucial and the construction of a classifier is closely related to the way the data are presented. Building a classifier can be seen as learning a mapping from a set of data $X$ (let $X$ be a set of $n$ points, where $x_i \in X$ for all $i \in [n] := 1, .., n$) to a discrete set of "labels" or classes $Y$. Matthias Seeger says in [18] that *learning from data can be seen as the most rigorous attempt to*

*"drastically compress" data without losing much of inherent information.* These learning techniques can be classified according to the availability of label data:

– Supervised learning: the activities have been previously predefined and the classification must match these specifications. Examples of these classifiers are artificial neural networks, decision trees, support vector machines, Bayesian statistics, etc. Thus, the goal is to learn a mapping from $X$ to $Y$, given a training set of pairs $(x_i, y_i)$. Here, the $y_i$ are called the labels of the samples $x_i$.
– Unsupervised learning: the activities are inferred from grouping by observing the data without using labels [47]. Thus, in unsupervised learning all the observations are assumed to be caused by latent variables and unsupervised classifiers study how systems can learn to represent particular input patterns in a way that reflects the statistical structure of this input patterns group. Examples of these classifiers are Radial basis function networks, all clustering methods (i.e. k-means) Expectation-Maximization, etc.
– Semi-supervised learning (SSL): the classifier is generated with both labeled and unlabeled data. It is very common to have a training set in which the unlabeled data are much more numerous than the labeled ones. Labeling video sequences is usually done by hand, which is tedious and difficult. These techniques, then, solve this problem. In particular, this work will analyze the data from a public database, CAVIAR [19], in which this situation is occurs. This is the main reason this dissertation will use a semi-supervised learning technique to generate an unsupervised classifier that will make use of labels for the validation stage [48].

One important distinction among all of these methods is the use or not of statistical learning; that is, do the learning techniques utilize concepts from probability theory or not? The main advantage of utilizing probabilistic concepts is that the classification error/success probability for new income samples is modeled by means of probability functions.

Again, and following the argument of this dissertation, this thesis aims to build a statistical classifier capable of modeling and sorting activities covering the widest number of situations; that is, the classifier must model the measurements or features selected in the previous stage by means of a probabilistic model. The measurements captured by the tracker and the corresponding features computed from these measurements usually have distributions that do not fit in common probabilistic models like Gaussians.

Therefore, the first step will be to choose a classifier and then decide how to build the probabilistic model from the features.

Many works have been developed in the field of classifying activities [20]. Nevertheless, the most relevant methods for classifying are those that take time sequences into account since human activities usually follow some kind of temporal pattern. That is, if a person is inactive in a specific scene, it is very unlikely that person would be running fast in the next frames. There would be some transition in which the person starts moving and then running.

Hidden Markov Models (HMMs) [21] have proved to be very effective for modeling these temporal transitions and the activity recognition task [22]: HMMs are tools that provide sophisticated analysis of data with spatio-temporal variability. The use of HMMs is divided into steps: training and classification. In the training stage, the number of states is set and the state transition and output probabilities are optimized in order to build the most suitable model that corresponds to the features of the movements or activities.

A very common method to model HMMs consists in using probabilistic parametric techniques, especially mixtures of functions, and in particular, mixtures of Gaussians. The estimation of the parameters in this case is carried out by means of an Expectation Maximization algorithm called **Baum-Welch algorithm** [23].

Nevertheless, the mixture of Gaussians presents two important limitations:

- the observed data are difficult to model by these probability density functions.
- the variability in the results depends on the initialization of parameters.

Therefore, the author proposes to model the emission probabilities of the HMM by means of kernel density estimation (KDE) [24], a non parametric technique that will operate simultaneously with that of all the other model parameters by an adapted Baum-Welch algorithm. This allows the retention of the maximum-likelihood estimation while overcoming the known limitations of mixture of Gaussians in modeling certain probability distributions. *Non-parametric* means that no assumptions are made about the form of the probability density functions (PDF's) from which the samples are drawn.

Thus, Kernel Density Estimation constructs an estimate of an unobservable underlying probability density function based on the observed data without assuming prior knowledge of the distribution of the data. The hidden density function is built from a large and distributed population of data; a probability function called kernel is set in each data point. In addition, the kernel is defined as a weighting function $K()$ (with bandwidth $h$), which can be any probability function depending on the type of data to be modeled: uniform, triangle, Gaussian, cosine, etc.

What makes KDE suitable and interesting for the generation of classifiers is that under realistically complex environments, the observed data and features for the purpose of activity recognition can be too complicated to assume a distribution of the data. This technique gives a very precise representation of the data regardless of the forms they have. For the same reason, the KDE is less sensitive to initialization, since the probability density function is estimated from the data. As a result of this, large errors derived from making incorrect assumptions about the distribution are avoided by using this non parametric technique.

It should be noted, however, that predictions out of the range of the observations are more difficult than in the parametric methods. Another drawback of the non-parametric techniques is that they usually require a great deal of training data to provide a good estimation.

## 2.5 Motivation and Thesis Objective

The goal of this thesis is to obtain a robust and reliable mono-camera video surveillance system capable of dealing with changing conditions and different locations and fields of view. In order to build this system, all the processes included in a surveillance system must be carefully designed and adjusted. In particular, this work focuses on tracking of humans and moving vehicles and human activity recognition tasks.

Many projects for tracking systems have been developed. Nevertheless, many of them fail when they are used under real conditions: occlusions, changes in lighting, abruptness in motion, etc. The contextual and prior information is not usually taken into account in the design of the tracking systems [49]. For example, in the context of this thesis, tracking of humans (vehicles), the design must take into account that they usually are on specific footpaths (roads) and are constrained to paths on the ground as apposed to making vertical movements. Some of the literature shows us an improvement in performance by using this contextual information [50]. In addition, all of the methods that constitute the trackers are controlled by parameters that condition the final performance. Very often, the adjustment of these parameters is carried out ad hoc for the specific application without considering the relocation (or movement of the focus) of the camera in the same application environment or the multiple types of changing conditions. Moreover, the classical optimization methods for the adjustment of parameters do not give good results since the complexity of this problem makes the computing time increase as a function of precision, and the generation of a classifier for the activity recognition by means of a kernel density estimation technique.

This thesis proposes an optimization in the whole tracking process and the generation of a classifier for activity recognition by using a non-parametric probability estimation model adaptable to difficult distribution of the measurements.

The optimization technique is based on Evolutionary Strategies and it has three main goals:

- First of all, to tune the parameters so that the system has the best performance in real conditions (noise, occlusions, changing lighting, etc.) taking into account the context of use. The performance is measured by means of a evaluation function that assesses and quantifies the functioning of the moving object track.
- Second, to obtain a set of parameters that cope with general situations, that is, that

the obtained parameters are useful for different scenarios of the same application environment.

- Finally, the technique must be a general frame for the tuning of the parameters of all kinds of detector/tracking systems in any environment.

The proposed technique consists in minimizing several combinations of an evaluation-per-track function. This performance evaluation function of the tracking system must allow the evaluation of multiple scenarios in an easy way. Current performance evaluation methods based on ground truth are computationally expensive and tedious. In addition, those based on non-ground truth methods do not allow the evaluation of a large number of scenarios.

This thesis overcomes this drawback by proposing a new quantitative performance evaluation function based on a minimum and easy-to-calculate ground truth. Thus, this new evaluation function allows us to compute the performance for numerous scenarios.

The strategy to prove the validation of this technique is to use the proposed methodology over a simple point tracker based on a Kalman filter and some easy association rules. The performance after the adjustment of the parameters should be better than if a more sophisticated and slow tracker had been used.

Once the object detection and tracking stages are adjusted, this thesis deals with the problem of activity recognition. There are many works in this field, but it has been proved that Hidden Markov Models solve this problem very efficiently. Nevertheless, the modeling of the HMM have usually been carried out by models that assume a certain distribution of the data (usually Gaussian).

This thesis proposes the application of a non parametric technique based on Kernel Density Estimation (KDE) for the modeling of the HMM. The assumption of an underlying distribution is avoided by using this technique, but it should be noted that predictions out of the observation data range are more difficult than in the parametric methods.

## 2.6 Outline

This document is divided as follows. Section 3 presents a review of the video technology in general, the main tracking methods, the most relevant evaluation techniques, a summary of the optimization and generalization methods and finally an introduction and review of the most important human recognition methods. Section 4 presents then our evaluation metric proposal, followed by section 5, which presents the technique for optimization and generalization suggested by the author. Section 6 presents the human activity recognition classifiers and methodology that this paper will follow. Finally, conclusions and future work are explained in section 7

# Chapter 3

# State of the Art

The section reviews the principal existing works in the topics in which this thesis focus on:

- Motion Detection and Object Classification: Brief enumeration of methods for motion detection and object classification.
- Tracking: Review of the principal tracking methods.
- Evaluation: Summary of the main performance evaluation metrics for video surveillance systems.
- Optimization and Generalization: learning techniques for a system optimization and,
- Activity Recognition: Extraction of features and methods for classification

## 3.1 Motion Detection and Object Classification

The objective of this process is to segment correctly regions corresponding to moving objects from the rest of the image, that is, to distinguish which pixels are in movement and which ones are just background. The correct detection of moving regions is a key problem to provide good results to the tracking and behavior analysis. Among many techniques, the most conventional methods used for motion detection are three:

1. Background subtraction, [7] takes the difference between the reference background and the current image. The reference background consists in a model of the scene that will be particular for each kind of scenario. For example, if the surveillance action is carried out in a room with constant-intensity lights and non-changing furniture, the background will be easy to model. On the other hand, if the surveillance

system is placed in an outdoor parking lot, the background has to consider that steady cars (which are part of the background) suddenly can move, or cars that are moving can be parked for long periods of time (and become part of the background). Moreover, the background model will have to take into account illumination changes from day to night, weather conditions (cloudy, sunny, rainy), waving plants, etc.

In particular, in [51], the authors model the color distribution of each pixel by means of a single Gaussian, which is described with a full covariance matrix. They define $\mu$ to be the mean $(Y, U, V)$ of a point a texture feature, and $K$ like the covariance of the same point.

$$Pr(O) = \frac{e^{-\frac{1}{2}(O-\mu)^T K^{-1}(O-\mu)}}{(2\pi)^{\frac{m}{2}} |K|^{\frac{1}{2}}} \tag{3.1}$$

The pixels update their statistics using a simple adaptive filter:

$$\mu_t = \alpha y + (1-\alpha)\mu_{t-1} \tag{3.2}$$

where $y$ is the vector that models the pixel (color distribution $(Y, U, V)$). This filter allows the detector to update the background according to the lighting and slow objects changes. The $\frac{1}{\alpha}$ is the time constant of the forgetting process.

In addition, another interesting paper is presented by Nir Friedman and Stuart Russell [52]. They support the idea that *each pixel must be classified before being used to update the background model.* They propose modeling each pixel by a mixture of Gaussians in order to classify among moving object, shadow or background. Moreover, the mixture of Gaussians is learned using the Expectation Maximization algorithm [53].

Nevertheless, more advanced schemes for updating the background model have been proposed in [7], [54].

2. Temporal difference takes the difference between pixels in two or three frames. It is very adaptable to the changing conditions but it is not very effective to detect all the moving regions.

   In particular, in this work [6], the authors take the difference of intensity $I$ between pixels:

$$\Delta_t(u,v) = |I_t(u,v) - I_{t-1}(u,v)| \tag{3.3}$$

   and the motion pixels $M_t$ can be detected by thresholding:

$$M_n(u,v) = \begin{cases} I_t(u,v) & if \quad \Delta_t(u,v) \geq T \\ \\ I_t(u,v) & if \quad \Delta_t(u,v) < T \end{cases} \tag{3.4}$$

3. Optical flow uses flow vectors of moving objects over the time. Most of the flow methods are very sensitive to noise. One example can be viewed in [55], where the segmentation is performed by using optical flow information; in particular, a method based on monotony operators, which is described in [56]. This method is based on the assumption that local intensities do no vary from frame to frame (*image brightness constancy constraint*). Thus, the equation that relates optical flow $v = [dx/dt, dy/dt]$ and the derivatives of the intensity function $I(x, y, t)$ is [32]:

$$(\nabla I)^T v + \frac{\partial I}{\partial t} = 0 \tag{3.5}$$

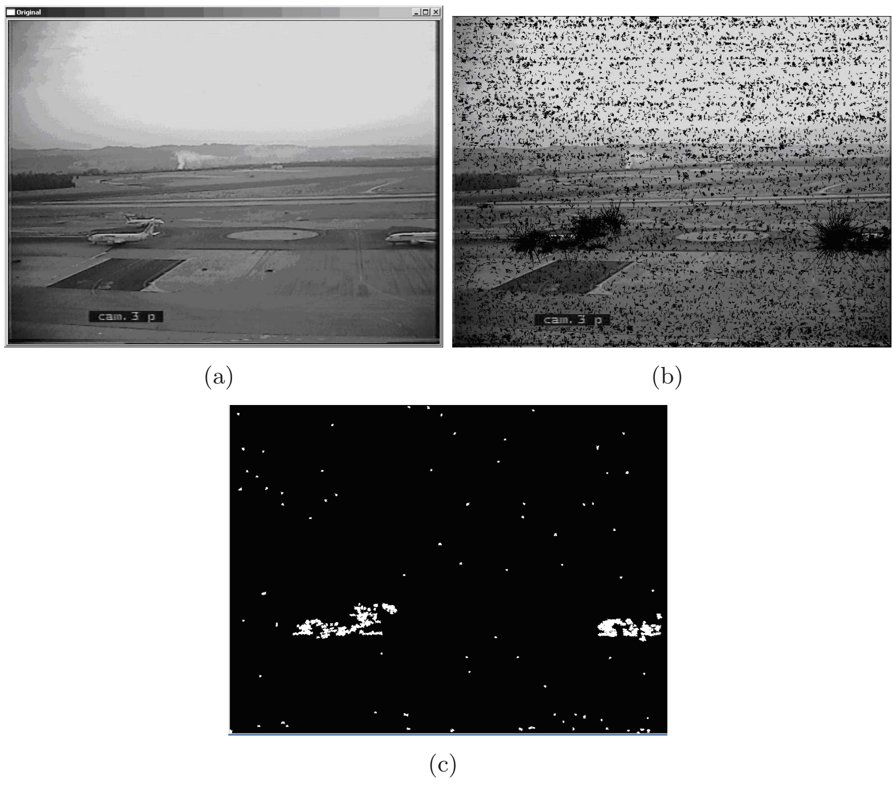


Figure 3.1: Example of optical flow: a) Original image, b) Optical flow and original image and c) Pixels with optical flow

On the other hand, there is another group of techniques that is based on spatial and temporal segmentation of the video signal. The core of these last methods takes into account the changes in the pixel spatial properties in order to detect moving regions, in addition to the common characteristics like pixel intensity and color changes [57, 58].

The target classification process classifies objects that correspond to different moving targets. For example, in people tracking contexts, some vehicles or other moving targets might be detected along with individuals. It is important to classify them so that the subsequent action recognition process analyzes the activities of each one in the correct way. This classification can be considered as a pattern recognition step that uses, for example, shape-based [6] or motion-based criteria [59]. In [6], blob density and areas are used to classify blobs into humans, vehicles or clutter; whereas the authors employ in [59] the characterization of periodic motion to classify moving objects.

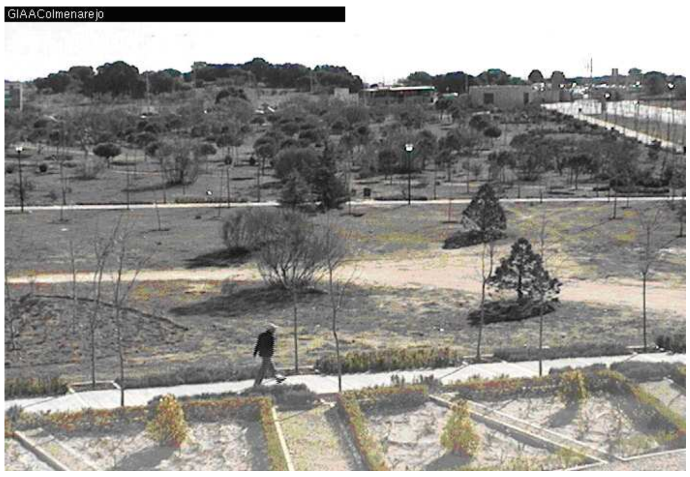


Figure 3.2: This figure shows two moving targets: a pedestrian in the first plane and a bus in the top right of the image. It will be important for the system to identify each one as different kind of targets.

## 3.2 Tracking

Tracking can be defined as the estimation of the trajectory of a moving object in the image plane [49] over time. Trackers must perform two main tasks:

- Motion problem: Usually, trackers estimate the object position in the next frame by using the position and velocity of the object in the current frame. A very well known solution for the estimation problem is the Kalman filter as we will see in the next lines [43].
- Association problem: This process of the tracker tries to find the target among several valid moving regions that might be other targets, clutter or noise.

The main design requirements that a tracker must accomplish can be enumerated as [32]:

- Stability: good performance has to be kept over time.

- Accuracy: trackers must follow the accelerations and speeds of targets.
- Robust: trackers must maintain the track even if this one is occluded temporarily. In addition, trackers must track only valid objects and ignore noise and image regions similar to the target.

Many methods have been proposed for object tracking. The main differences among them are based on the target representation used by each method, or which features of this moving target are used, or which model represents better the motion, appearance and shape of the moving object [49]. According to these criteria, there are numerous classifications [49], [32], [20] as it is explained next.

The first step is the representation of the target [49]. Objects can be represented by their shape and appearances. Then, the next paragraph will show the joint shape and appearance representations.

- Shape:
  - Points: The target is represented by a point (mass center or centroid) or a group of points [60].
  - Geometric Shapes: The target is represented by a geometric shape, usually a rectangle [30] or an ellipse.
  - Silhouettes and Contours: Contour is defined by the bounds of the target [61], whereas silhouette is the region inside the contour [62].
  - Skeletons: The target is represented by a skeleton that can be computed by applying medial axis transform to the object's silhouette [63].
  - Joints: Joint representation is usually applied to human tracking. The target is represented by body parts or joints [64]. The relation among all the parts is defined by means of kinematic motion models.
- Appearance:
  - Probability densities: The probability density of certain objects' features are computed and represented by parametric functions, non-parametric functions or histograms [44]. Moreover, the probability functions might be computed from the shapes representations defined before, for example the color inside a contour.
  - Templates: Templates are built by taking as a base the silhouettes or the geometric shapes that were enumerated before. They encode spatial and appearance information of a single view [65]. This might be a limitation if the object's poses vary significantly during the video sequence.
  - Multi-view appearance: The codification is performed by using several views of the object.

In general, there is a strong relation between object representation and tracking algorithms as it will be subsequently analyzed . The next step in the tracking process is to select the features that the tracker is going to track. This is a key point in the design of the tracker since it would be desirable to select distinguishable features so that the objects can be easily differentiated. Feature selection is closely related to the object representation; for example, color is almost always used with a histogram representation, whereas edges are employed for contour representation,

The most common features to track are:

- Color: Color is one of the most widely used features. There are several representation spaces [66], such as "'RGB"' or "'HSV"'. However, color space representations are very sensitive to noise [67].
- Optical Flow: It is defined by the displacement of pixels in a region. Optical flow is computed by supposing that brightness does not vary in consecutive frames (see equation 3.5). Optical flow is commonly used in motion segmentation and tracking systems [68].
- Edges: They are defined by target boundaries. They have proved lower sensitivity to illumination changes than colors [69].
- Texture: It measures the difference of intensity on a surface that quantifies properties such regularity, coarseness and smoothness [49]. The three principal approaches to describe texture are statistical, structural or spectral properties. Statistical properties compute the gray level of the surface. Structural properties are based on certain rules that characterize the object's surface. And finally, spectral components are extracted from the Fourier spectrum and describe periodicity of the gray levels.

Among many classifications found in the literature for tracking systems ( [32], [20]), the author follows the classifications that A. Yilmaz et al. suggested in [49]. According to A.Yilmaz, the tracker can be classified as:

- Point Tracking: Targets are represented by points and the matching of these points is based on the position and motion of the target in the previous state.
- Template Tracking: The object is represented by a geometrical region and the tracking follows the motion of this template or kernel.
- Silhouette Tracking: These tracking methods estimate the silhouette in each frame and code the information inside this region, i.e. in the form of appearance density of the object. The object's appearance density is estimated by parametric (such as Gaussians) or non parametric (such as histograms) techniques.

**Point Tracking**

Point Tracking is defined as the representation of an object by points and the correspondence of these points in each frame. This correspondence is a difficult task, especially in the presence of occlusions and misdetections. There are two types of methods for this correspondence: non-statistical and statistical.

- Non-statistical correspondence is defined as a cost association problem formulated as a combinatorial optimization. Some constraints are taken into account such as:
  - Proximity: It is assumed that the position of the target does not change considerably from one frame to the next one.
  - Velocity Change: Another assumption considers that the direction and speed of the target does not change notably from one frame to the next one.
  - Rigidity: The targets in 3D are assumed to be rigid. Therefore, the distance between any two points remains unchanged for the actual object.

  For example, Sethi and Jain [70] use a greedy approach for the points correspondence based on the proximity and rigidity constrains. More recently, Shafique and Shah [71] presented an approach to maintain temporal coherency of the speed and position. The correspondence was formulated as a graph theoretic problem.

- Statistical correspondence assumes that noise is inevitably captured when the measurements are acquired. Statistical statistical correspondence includes noise in the model used to solve the tracking problem. Statistical correspondence methods model target properties such as position, velocity and acceleration. Commonly, the measurement ($Z^t$) consists of the target position in the image in each frame.

  If we consider a moving object in the image, in which the information representing the object is the state sequence $X^t : t = 1, 2, ....$ The dynamic equation that changes the state over time is:

  $$X^t = f^t(X^{t-1}) + W^t \tag{3.6}$$

  where $W^t : t = 1, 2, ...$ is white noise. Moreover, $Z^t = h^t(X^t, N^t)$ is the relationship between the measurement and the current state, where $N^t$ is the white noise and it is independent of $W^t$.

  The aim of the tracker is to estimate the current state $X^t$ by knowing all the measurements up to the current moment: $p(X^t|Z^{1,...,t})$. The Bayes theorem provides an optimal theoretical solution in two steps: first of all, the current state is derived from $p(X^t|Z^{1,...,t-1})$. Then, the likelihood function $p(X^t|Z^t)$ is used to compute $p(X^t|Z^{1,...,t})$.

  If there is only one moving object in the image, the previous calculations can be applied directly. Otherwise, the measurements have to be associated previously with the corresponding object states. Then, we have two important cases to distinguish: Single Target and Multiple Target Methods.

- Single Target State Estimation:
  * Kalman Filter: It is used if the functions $f^t$ and $h^t$ are linear and the state $X^t$ is assumed to be Gaussian. This filter has two steps: prediction and correction. The former state model is used for predicting the new state variables:

$$\overline{X}^t = DX^{t-1} + W \tag{3.7}$$

$$\overline{\Sigma}^t = D\Sigma^{t-1}D^T + Q^t \tag{3.8}$$

where $\overline{X}^t$ and $\overline{\Sigma}^t$ are the state and covariance at $t$, the prediction time. $D$ is the state transition matrix between $t$ and $t-1$ and $Q$ is the covariance noise of $W$. In addition, the measurement $Z^t$ is used to update the target's state in the correction step:

$$K^t = \overline{\Sigma}^t M^T \left[ M\overline{\Sigma}^t M^T + R^t \right]^{-1} \tag{3.9}$$

$$X^t = \overline{X}^t + K^t \underbrace{\left[ Z - M\overline{X}^t \right]}_{v} \tag{3.10}$$

$$\Sigma^t = \overline{\Sigma}^t - K^t M \overline{\Sigma}^t \tag{3.11}$$

where $v$ is the *innovation*, $M$ is the measurement matrix and $K$ is the Kalman gain.
In the Extended Kalman Filter (EKF) [72], the functions of the state transition $f^t$ and observation models $h^t$ do not need to be linear but differentiable in order to use the Taylor series. Nevertheless, the EKF is not an optimal estimator. The filter might diverge if the initial state or the model process are wrong due to its linearization. Moreover, the estimated covariance matrix usually underestimates the true covariance matrix and then, some inconsistent problems may arise.
  * Particle Filters [73]: An alternative for the Kalman (KF) and Extended Kalman Filter (EKF) can be Particle Filters. Kalman Filter and Extented Kalman Filter assume that the state variables are Gaussians and present poor performance when the state variables do not follow Gaussian distributions. The Particle Filters model the conditional state density $p(X^t|Z^t)$ at time $t$ with a set of samples or particles $\{s_t^{(n)} : n = 1, ..., N\}$ and weights $\pi_t^{(n)}$, which represent the importance of each sample (its observation frequency). The new samples at time $t$ are derived from $S_{t-1} = \{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}) : n = 1, ..., N\}$ at $t-1$ based on sampling schemes [74].

The equation for estimating the new position will be:

$$\epsilon_t = \sum_{n=1}^{N} \pi_t^{(n)} f(s_t^{(n)}, W) \tag{3.12}$$

where $f$ is a non-linear function and $W_t^{(n)}$ is a zero mean Gaussian error. It is important to point out that Kalman filter and Particle filter assume a single measurement at each frame. If the aim is to track several targets, we would need a joint solution of data association methods.

– Multiple Target State Estimation: Association methods must be used if more of one object is tracked. There are three main algorithms:
  * Nearest Neighbor (NNS): Deterministic method of optimization for finding closest points in a metric space [75].
  * Joint Probability Data Association Filter (JPDAF): Statistical method that calculate joint probabilities between the tracks and the measurements. The algorithm assigns weights to the measurements and updates the tracks by means of weighted centroids of these measurements [76].
  * Multiple Hypothesis Tracking (MHT) [77]: Statistical method that makes the matching decision after analyzing several frames. This algorithm computes several correspondence hypothesis for each object at each frame.

The deterministic methods can make the tracker not to converge if the objects are too close and the association is done incorrectly. On the other hand, the main drawback of JPDAF is its inability in detecting new targets that appear in the field of view (FOV) or already tracked targets exiting the FOV. The MHT algorithm overcomes these problems, but it has the shortcoming of being computationally exponential in memory and time.

### Template Tracking

Following the classification that A. Yilmaz established in this work [49], Template trackers represent the target by a primitive object region and compute motion of this object. The algorithms developed differ from each other in how they represent the object, the number of them tracked by the tracking system and the method used to estimate the object motion. The division of the algorithms is done in this case according to the object's representation.

Templates and Density Appearance Models are widely used for their simplicity and fast computation. The most common approach is the template matching. First, an object template, $O_t$, is defined in the previous frame and then, a similar image, $I_w$, must be searched in the current frame using brute force search.

$$argmax_{dx,dy} = \frac{\sum_x \sum_y (O_t(x,y) \times I_w(x+dx, y+dy))}{\sqrt{\sum_x \sum_y O_t^2(x,y)}} \tag{3.13}$$

where $(dx, dy)$ is the candidate template position. Usually, templates are made of intensity or color features. The only drawback of this method is the computational cost due to the brute force search.

Other representations, such as color histograms or mixture models can be calculated by using pixel appearance in the bounds of the rectangular or ellipsoidal regions. For example, in [78], P. Fieguth and D. Terzopoulosi build object models by finding the mean color of the pixels in a rectangular region in the target. In [44], the authors employ a weighted histogram computed from a circular region representing the object. Instead of using the brute forte search, they use the Mean-Shift method [46]. The Mean-Shift tracking algorithm is an iterative scheme based on comparing the histogram of the original object in the current frame, $Q$, and the histogram of candidate regions in the next frame, $P$. The objective is to maximize the correlation between two histograms. Similarity in the histogram is defined in terms of the Bhattacharya coefficient:

$$\sum_{u=1}^{b} P(u)Q(u) \tag{3.14}$$

where $b$ is the number of bins. The mean-shift vector is calculated iteratively such that the similarity between histograms is increased. This process follows until the converge is reached.

In addition, the work developed by Jepson et al. [79] models an object by three mixture components: noise, transient features and stable appearance features. A particular version of Expectation-Maximization algorithm (EM) is used to adjust the parameters.

All these algorithms do not take into account the problem of tracking several objects at the same time. An example of tracker for multiple objects is [80], where the whole image is represented as a set of layers. There is a layer for the background and one for each object. Then, the probability of belonging to any of the layers for each pixel is computed according to motion and shape features.

### Silhouette Tracking

Finally, silhouette tracking systems provide a precise shape description for objects that have complex shapes. The most common example of a complex shape is the human body: head, hands, shoulders, etc. Silhouette trackers try to find the target region by means of a model built in previous frames. This model can be a color histogram, an object contour, etc. We can divide these trackers in two categories: shape and contour matching trackers.

- Shape Matching Tracker: These methods search for the target silhouette in the current frame. The search is carried out by computing the similarity with the the silhouette model calculated in the previous frame, whereas Template Matching Trackers (see previous section 3.2) used a template that are usually built by intensity or color features.

Thus, the target model is recalculated every frame in order to take into account the changes produced by the moving object. For example, Huttenloncher et al. [81] use the *Hausdorff* distance [82] to perform shape matching using an edge-based representation. Another example is given by Haritaoglu et al. [7] that utilize the edge information contained inside the object silhouette to model the object appearance.

- Contour Matching Tracker: These methods iteratively evolve an initial contour, which was computed in the previous frame, up to a new contour in its new position. A requirement for this evolution implies that some area of the new and old contour overlapped in the current and previous frame respectively. There are two approaches:

  - Using State Space Models: Shape and motion parameters of the contour are used to define the object's state. The update of the state is carry out at each instant by means of the maximization of the contour's a posteriori probability. This posterior probability is a function of the prior state and the current likelihood. The current likelihood is usually defined according to the distance of the contour from observe edges. Some of the most remarkable papers are those one developed by MacComick and Blake [83], Chen et al. [84] or Isard and Blake [85].
  - Using Minimization of Contour Energy: Temporal information constitutes the base for the definition of the contour energy. This temporal information appears in the form of optical flow [86] or appearance statistics generated from the object and the background [61]. For example, Mansouri [87] employed optical flow constraint for contour tracking, Cremers and Schnorr [86] also used optical flow for contour evolution, and they imposed that only homogeneous flow vectors can be inside the region of an object. In contrast, certain statistics inside and outside the object region can be defined as an alternative to optical flow. For example, the work [61] used color and texture models computed in a region around the object's boundary in order to evolve an object contour.

**Analysis of Tracking Systems**

Among all the general techniques explained above, the author is going to select specific trackers that have proved robust performance in order to compare their performance against our proposed tracker ("Rules Tracker, see sections 5.2.1 and 5.2.2).

- CC (Connected Component Tracking) [14]: This is a kind of Template tracking algorithm, where certain connected components (CC) form a 3D connection graph. Points which distance is below a threshold are considered to be *connected*. Then, the tracking procedure identifies which CC in frame $(t-1)$ matches with the corresponding ones in the frame $t$. Besides the geometrical points, the system takes advantage of the texture image to improve the tracking.

- MS (Mean Shift Tracking) [15]: This is a class of Template Based Tracking in which the Mean Shift algorithm is a non parametric estimator of density gradient: *the mean-shift vector always point towards the direction of the maximum increase in the density and it can define a path leading to the maximum* [88]. The core of this tracking system consists of mean shift iterations that find the target candidate that is the most similar to a given target model calculated in previous frames. This target model is computed by means of the color distribution of the moving target. Then, the degree of similarity is computed by a metric based on the Bhattacharyya coefficient. In other words, the aim of the tracking system pursues to maximize the correlation of the current color histogram of a moving target with the original one computed in previous frames.

- MSPF (Particle Filter based on MS weight) [16]: A hybrid model between Mean Shift and Particle Filter models (a class of Point Tracking). A particle filter samples some particles and subsequently, those particles are shifted to their respective local maximum of target searching space by mean shift.

- CCMSPF (Connected Component tracking and MSPF resolver for collision) [14]: It uses a Connect Component Tracking in addition to a MSPF in case a collision is found it.

- CGA (Association by Canonical Genetic Algorithm) [17]: This is the only tracker that do not belong to any of the categories explained before. This method uses Canonical Genetic Algorithms for tracking. The Compact GA computes a probability that represents the distribution of a hypothetical population with update rules based on the well known GA techniques of selection and recombination.

These trackers are usually designed for specific cases and particular conditions: For example, the MS (Mean Shift) tracker easily loses the object due to its intrinsic limitation of exploring local maxima, in particular, when the tracked object has quick movements. In addition, the mean shift is hard to recover from a total occlusion [16]. MSPF is restricted to applications with little changes of the target model. Finally, CGA presens slow performance, which might be a real problem when tracking several targets.

Thus, even if these trackers have proved robust performance, they present strong limitations. Therefore, this thesis proposes the use of a flexible tracker that is based on point tracking. The values of the parameters that control this tracker will be optimized by means of an optimization technique proposed by the author. This thesis intends to overcome the limitations mentioned above and to obtain better performance than the one shown by more complex trackers. The proposed tracker and the optimization technique will be introduced in the next chapters.

## 3.3 Evaluation

The usual requirement for surveillance systems is the capability for tracking interesting objects in operational conditions, with satisfactory levels of accuracy and robustness. This capability could be directly tested by a human observer by a visual inspection of the output. However, the main problem lies in the definition of an automatic and objective procedure to assess the quality of a given system and support design decisions. These techniques must assess the good performance of both the segmentation and tracking steps.

Many solutions have been proposed, the majority of them are focused on the segmentation stage or detection of moving objects [8, 89–91], others deal with the evaluation of tracking systems [9] or both [10]. Whereas the evaluation of the segmentation stage finds errors in the detection (i.e. missing parts of objects), the tracking evaluation focuses on the high level detection, that is, object detection and trajectory.

The segmentation step is important for the global performance of the surveillance system since it influences significantly in the subsequent stages [92]. The main goal of segmentation consists in efficient moving object detection.

Tracking systems are in charge of initiating, updating and deleting tracks corresponding to detected objects. Tracking performance evaluation must assess the occlusions, track loss or duplication [93], failures due to sudden movements of objects in terms of acceleration [10], detection lags [94], etc.

Although many segmentation and tracking techniques have been presented and widely studied, their performance evaluation has not focused so much attention. Nevertheless, several important works exist in the field [95] and this section pursues to briefly present the most relevant existing results and conclusions.

A first classification of evaluation techniques may be established between pixel and object based methods [92]. The first group is a binary problem that tries to detect active pixels in an image [96]. The applicability of these techniques is questionable since the surveillance purposes are not to detect points but object regions. Thus, object-based techniques seem more appropriate. According to [97, 98], we can distinguish between two techniques when applying video segmentation object-based evaluation:

- Individual object evaluation: each target identified in the segmentation stage is evaluated individually.
- Overall evaluation: all the objects are evaluated globally.

Another classification can be done according to the criterion of using some reference or ground truth:

- evaluation based on ground truth: a reference segmentation is available.

- evaluation without ground truth: a reference segmentation is not provided.

This thesis focuses on the tracking performance evaluation, assuming a good detection stage.

The next two subsections review the currently available methods for evaluation tracking performance with and without reference.

### 3.3.1 Evaluation with Ground Truth (GT)

The performance evaluation of video-based surveillance systems is a basic aspect for their appropriate design and parameterization. Today, no standard measure sets to assess algorithms quantitatively are available [93]. However, first attempts have been made to work on common data sets (PETS workshops [41]); most recently, the project ETISEO has been funded by the French government for the systematically quantitative assessment of surveillance algorithms [99]. For example, in [9, 100], the measures depend on an established correspondence between ground truth tracks and detected tracks. A distance measure has to be computed between all ground truth and detected tracks for every frame. This is a very general way of comparing trajectories, which takes the position, its first derivative and objects' bounding boxes into account. According to [100], once the assignment between the ground truth and the detected tracks is established, some error measures are computed: false positive and false negative track error rates, average of position errors, object detection lag, etc. There are similar papers that evaluate video surveillance systems against the ground truth or with synthetic images [9,30,101,102]. They usually need the extraction of ground-truth values from real images; this is a manual process of hand labeling, which requires a considerable effort to obtain.

In general terms, the metrics employed in the evaluation with reference might be classified into two categories [98]:

- Spatial accuracy: these metrics compare the selected spatial features between the real and detected tracks. The most relevant metrics are:
  - Shape fidelity: metrics based on the misclassification of shape pixels [103], edge pixels [104], etc.
  - Spatial features: comparison of bounding boxes, mass centres [30], perimeters, ...
  - Number of objects: number of detected objects: merge or fragmentation [93, 105], occlusions, loss of tracks, [30, 93], ...
- Temporal accuracy: these metrics measure temporal stability, for example, variation in the object's shape along time, if one object is missed in a specific frame (false negative), etc.

Hence, some of the most common metrics presented in the literature [9,93] for tracking performance are listed below:

- False positive track error rate ($f_p$):

$$f_p = \frac{\text{Detected tracks without corresponding GT}}{\text{Total number of GT tracks}} \tag{3.15}$$

- False negative track error rate ($f_n$):

$$f_n = \frac{\text{GT tracks without corresponding detected tracks}}{\text{Total number of GT tracks}} \tag{3.16}$$

- Distance metric [100]: Distance between the current location of the track (i.e. track's mass center) and ground truth:

$$D(GT, r) = \frac{1}{N_{GT,r}} \sum_{\exists i GT(t_i) \wedge r(t_i)} \sqrt{(x_{GT}(i) - x_r(i))^2 + (y_{GT}(i) - y_r(i))^2} \tag{3.17}$$

  where $N_{GT,r}$ is the number of frames of the matching ground truth and detected tracks.

- Area error: Difference of the areas of ground truth and detected tracks.

$$A(GT, r) = \frac{1}{N_{GT,r}} \sum_{\exists i GT(t_i) \wedge r(t_i)} (A_{GT}(i) - A_r(i)) \tag{3.18}$$

- Detection lag: It measures the difference in time between the appearance of ground truth and detected tracks.

- Track fragmentation: Number of detected tracks matched to ground truth track.

Ground truth evaluation techniques provide reliable performance metrics for tracking systems, since they are based on direct comparison of the truth and the output of the system. Nevertheless, the main drawback is the ground truth extraction process. As a consequence of this, researchers tend to carry out this extraction process once and use these videos as much as possible for design and evaluation, which implies little variety of scenarios. These arguments lead the author to propose an evaluation metric in which the ground truth is easy and fast to be extracted. In this way, the number and variety of videos will not constitute a limitation anymore, which will allow the system to be tested in different contexts and conditions.

### 3.3.2 Evaluation without Ground Truth

The number of papers dedicated to define metrics without ground truth are not so numerous [97, 106–108] as the ones with reference.

Erdem *et al.* presents a very rigorous set of metrics (for both, segmentation and tracking) and their correlation with metrics with ground truth [106]. The non-ground truth metrics exploit the colour and motion features in the surroundings of the segmented object. The first feature measures the spatial color contrast along the edge of each detected object. The second feature, which assesses the good or bad segmentation along a spatio-temporal trajectory, subtracts color histograms across segmented objects. Finally, the third and last metric consists in motion vector differences along the segmented objects. The relevance of this work is not only the definition of the non-ground truth metrics, but the proof of their validation. The correlation of these metrics with the ground-truth metrics proves the validation of this technique.

The work of Correia and Pereira [97] presents a group of metrics for evaluating the segmentation step. These metrics are divided into *Individual Object Segmentation Quality Evaluation* (each detected object is evaluated individually) and *Overall Object Segmentation Quality Evaluation* (global evaluation of a set of objects). The individual metrics rely on spatial features (shape regularity and uniformity), temporal features (temporal stability and motion uniformity) and contrasts with neighbors pixels along the border between the inside and outside of an object. On the other hand, the overall metrics assess the stability in the number of detected objects.

Finally, Wu and Zheng [108] present metrics that are based on the fact that *when tracking fails, the size and location of bounding box changes irregularly.* The authors propose 4 metrics, in which $Tr_i$ represents certain thresholds:

1. Trajectory complexity evaluation. This is perhaps the less intuitive metric. They measure the trajectory complexity *as the ratio of the trajectory path lenght $L_{p_1,p_2}$, and end points distance, $D_{p_1,p_2}$ between two tracking points $p_1 = P(t-\tau)$ and $p_2 = P(t)$*. Usually, as the ratio increases, he complexity of the trajectory also increases.

   Usually, the larger the ratio, the more complex the trajectory will be (see figure 3.3) The authors define the trajectory complexity rate as:

$$I_1(t) = \begin{cases} 0 \; if \; \frac{L_{p_1,p_2}}{D_{p_1,p_2}} \geq Tr_1 \\ 1 \; otherwise \end{cases} \tag{3.19}$$

2. Motion smoothness evaluation: It is noticed that the distance of an object to itself between two consecutive frames is higher if the tracker fails. Then, the authors define motion as the displacement of the object over two consecutive frames. Moreover, the indicator of motion smoothness is given by:

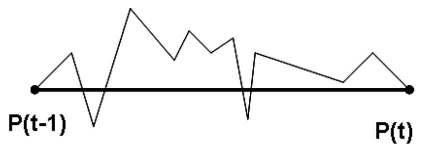


Figure 3.3: Tracking trajectory

$$I_2(t) = \begin{cases} 0\ if\ D_{p_{(t-1)},p_t} \geq Tr_2 \\ 1\ otherwise \end{cases} \tag{3.20}$$

3. Scale constancy evaluation: It is expected to have a small change in the scale factor of the target during the tracking process. The ratio between the area of the current target bounding box, $A_t$ and the initial one, $A_0$ is measured in every frame. The scale indicator is defined as:

$$I_3(t) = \begin{cases} 0\ if & \begin{cases} \left\{\frac{A_t}{A_0} \leq Tr_{31}\right\} \cup \left\{\frac{A_t}{A_0} \geq Tr_{32}\right\} \cup \\ \left\{\frac{dA_t}{A_0 dt} \leq Tr_{33}\right\} \cup \left\{\frac{dA_t}{A_0 dt} \geq Tr_{34}\right\} \end{cases} \\ 1\ otherwise \end{cases} \tag{3.21}$$

4. Shape similarity evaluation: The authors use aspect ratio $Width/Height$ of the bounding box in order to measure the shape similarity:

$$I_4(t) = \begin{cases} 0\ if\ \left\{ \left\{\frac{W_t/H_t}{W_0/H_0} \leq Tr_{41}\right\} \cup \left\{\frac{W_t/H_t}{W_0/H_0} \geq Tr_{42}\right\} \right. \\ 1\ otherwise \end{cases} \tag{3.22}$$

5. Appearance similarity evaluation: Appearance changes are usually due to failures in the tracking process according to the authors in [108]. They use three criteria to measure the appearance stability: $D_I$ is pixel by pixel difference between the current and the initial object; $D_H$ is the difference of intensity histograms; and finally, $D_M$ is the sum of weighted differences between the current appearance model and the initial appearance model. The appearance similarity indicator is given by:

$$I_5(t) = \begin{cases} 0\ if\ D_I \geq Tr_{51} \cup D_H \geq Tr_{52} \cup D_M \geq Tr_{53} \\ 1\ otherwise \end{cases} \tag{3.23}$$

The evaluation methods presented in the literature are reliable and proved the objective of reflecting the faults of the system. Nevertheless, important limitations are presented:

- The techniques presented by Erdem et al. are strongly based on color measurements. That is, if the camera provides black and white videos, the evaluation will be incorrect.
- On the other hand, Correia and Pereira present robust and reliable evaluation metrics without ground truth. Nevertheless, these metrics only include the segmentation level, and not the tracking level.
- Finally, all the metrics presented by Wu and Zheng do not take into account track losses or label changes, etc.

As a general conclusion for all the evaluation techniques cited in the literature, there is great demand for automatic and objective performance metrics, not only for the evaluation itself but for the comparison among different surveillance systems.

## 3.4 Optimization

Optimization can be defined as the process of improving a system by modifying all or part of its parameters.

On the other hand, generalization is the process of computing general concepts by abstracting common properties of instances of a system.

In mathematics, optimization seeks to minimize or maximize a real function by finding the most suitable set of variables from within an allowed set.

Practical optimization is the art of obtaining the best effect by using the less number of resources. Optimization techniques belong to our everyday questions of scheduling, industrial planning, adjustment of parameters, etc. Many of the large scale optimization techniques were developed during the Second World War (i.e. the simplex algorithm) in order to deal with the logistic problems that arose in that time. Some questions such as where the optimum location of petrol supplies were, the best search for anti-submarine patrols, ...

In the case of this thesis, it is crucial the adjustment of the parameters of the tracking system in order to obtain effective results for diverse scenarios.

Among the many articles that we could find in the literature, the author highlights the following:

- For example, the author in [109] use a gradient-descent method with local step size adaptation for optimizing a visual tracker of articulated structures.
- In [110], the authors propose a method to obtain the less number of dimensions for a particle filter using a Gaussian Process Latent Variable Model (GPLVM) framework.

- The use of particle swarms for tracking and optimizing dynamic systems is carried out in [111].

- In [112], the authors cast tracking as a parameter estimation problem and use non linear optimization techniques to find the numerous solutions.

- The article in [113] exploits an idea of kernel-based track and shows a global statistical optimization method that allows to track up to six people trajectories in a complex context (many occlusions).

- The proposal in [114] is based on a statistical measure that is minimized by means of mean shift iterations that exploit the spatial gradient of such measure. Mean Shift is a non parametric, iterative procedure introduced by Fukunaga and Hostetler [115] for seeking the mode of a density function represented by a set of samples. Then, Cheng [116] developed a general formulation of mean shift as an optimization method. According to Cheng :

  *If $S$ is a finite set embedded in the n-dimensional L Euclidean space, $X$. Let $K$ be a flat kernel that is the characteristic function of the $\lambda$-ball in X,*

  $$K(x) = \begin{cases} 1 \; if \; ||x|| \leq \lambda \\ 0 \; if|||x|| \geq \lambda \end{cases} \tag{3.24}$$

  The *sample mean* at $x \in X$ is

  $$m(x) = \frac{\sum_{s \in S} K(s-x)s}{\sum_{s \in S} K(s-x)} \tag{3.25}$$

  The difference $m(x) - x$ is called mean shift. The repeated movement of data points to the sample means is called the mean shift algorithm. In each iteration of the algorithm, $s \leftarrow m(s)$ is performed for all $s \in S$ simultaneously".

  This is the plain definition of the mean shift algorithm. For convergence proof and explanation of the algorithm as an optimization method, the articles written by Cheng [116] and Comaniciu [114] are mandatory references.

- In [44], the same authors use feature histogram-based target representation regularized by spatial masking with an isotropic kernel. This allows the authors to use a spatially similarity function for target localization. They reduce the localization problem to a search, which uses a gradient optimization technique. This method performs a faster search than the exhaustive search.

- Finally, in [117], the author use the same kernel-weighted histograms and trust-region methods for optimization.

Optimization processes (figure 3.4) start from a real problem, which allows the user to create a model. In this abstraction problem (*analysis*), the designer must choose what elements are relevant or not. Then, when passing from an abstract model to a computer implementation (*numerical methods*), we must consider aspects such as calculation accuracy, efficient matrix inversion, ... The *verification* stage assures that a computer program is actually accomplishing its tasks. Finally, the results are compared between the model and the real problem (*validation*).

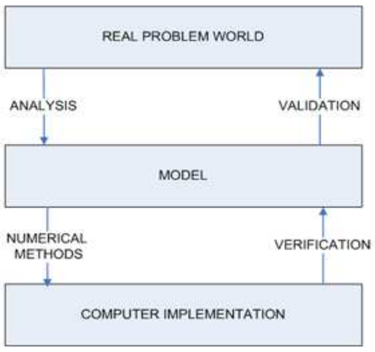


Figure 3.4: Process of optimization

Optimization could be formalized like follows. Given the function $f$

$$f : A \to \Re \tag{3.26}$$

find the elements $a$ such that

- Minimization:
$$f(a) \leq f(x) \forall a \in A \tag{3.27}$$
- Maximization:
$$f(a) \geq f(x) \forall a \in A \tag{3.28}$$

Commonly $A$ is a subset of the Euclidean space $\Re^n$ that must accomplish some constraints. The domain of $A$ is called the *search space*, while the elements of $A$ are called *feasible solutions* and the function $f$ *objective function* or *cost function. A feasible solution* that minimizes or maximizes the *objective function* is called an *optimal solution.*

If the objective function is not convex, there might be more than one local minima and maxima. The definition of local minima ($x^*$) says that there exists some $\epsilon > 0$ so that:

$$\|x - x^*\| \leq \epsilon \Rightarrow f(x^*) \leq f(x) \forall x \in A \tag{3.29}$$

The maxima or minima of a function can be *local* or *global*. It is said to be *global* when it is the minimum o maximum of the whole region of interest, whereas *local* means that it is only the maximum or minimum in some small neighbourhood of the region. According to this definition, two types of optimization algorithms can be distinguished:

1. Global optimization: Those algorithms that search for the global minimum (maximum).
2. Local optimization: Algorithms that search for local minima (maxima).

If there exists certain knowledge about position of the minimum, local optimization algorithms can be used. Otherwise, global optimization algorithms must be utilized [118]. Nevertheless it is very common to use both methods in some applications. For example, in order to adjust the parameters of a tracker, the first frames can be adjusted by a global optimizer, whereas a local optimizer is enough for the rest of frames.

### 3.4.1 Local Optimization

1. **Unidimensional Search**

   It is the simplest minimization problem in which there is only one variable. Some point $x$, among two intervals with a common point $b$, $(a, b) \cup (b, c)$, is evaluated in the function $f$. If $f(x) < f(b)$ then, $x$ replaces the point $b$. Otherwise $x$ replaces one of the end points.

2. **Multidimensional Search**

   Usually, the local minimizers use the Taylor series expansion as follows:

$$f(a+x) = f(a) + \sum \frac{\partial f}{\partial a_i} x_i + \frac{1}{2} \sum \frac{\partial^2 f}{\partial a_i \partial a_j} x_i x_j + ... \quad (3.30)$$

$$f(a+x) \approx c - bx + \frac{1}{2} x^T A x \quad (3.31)$$

   where $c = f(a)$, $b = -\nabla f$, $[A]_{ij} = \frac{\partial^2 f}{\partial a_i \partial a_j}$

   The matrix $A$ is called the *Hessian matrix* of the function at $a$.

   Thus, the gradient will be equal to zero, so $b = 0$ and

$$f(a_{min} + x) \approx c + \frac{1}{2} x^T A x \quad (3.32)$$

   where the matrix $A$ is evaluated in $a_{min}$

   There are some particular situations that can be solved like follows:

- Direction Vector [118]: If we have certain estimation of the minimum in $f(a)$, the search can be improved by selecting a direction vector along the line $a+\lambda u$. The unidimensional problem consists in finding $\lambda$ which minimizes the function $f(a+\lambda u)$. The repetition of the cycle through the $n$ unit direction vectors which span the parameters can lead to a solution or not. In order to avoid the last case, the *Conjugate Directions* can be calculated (i.e. by means of the Powell method [118]).
- First Derivatives: If gradient, $\nabla f$, can be calculated efficiently (requiring less computation than $n$ individual evaluations of the function $f$), there are some cunning methods to calculate conjugate set in a faster way than the Powell method.
- Second Derivatives: If gradient, $\nabla f$, and the Hessian matrix, $A$, can be calculated efficiently, faster and more robust algorithms can be used to find the minima (i.e. Levenberg-Marquardt (LM) Method).

### 3.4.2 Global Optimization

It is useful to use global optimizers when finding the global minimum among many minima is necessary for the problem or application. Among other methods, the author highlights the next ones:

- Simplex Algorithm: This method can get out of a local minimum in order to find better minima [118].
- Simulated Annealing: It is based on an analogy with the process of crystallization and freezing of liquids. If the liquid is cooled slowly, the molecules lose mobility and form a pure crystal, that is, the global minimum is the stable level of energy. Otherwise, if the liquid is cooled rapidly, the molecules form an amorphous state with higher energy than the pure crystal (local minima) [119–122]
- Multi-Resolution Methods and Graduated Non-Convexity: Sometimes, the function $f$ is very rough, having many sharp local minima. In this cases, it is very difficult to find the global minimum. Thus, in some situations a smoothed version of $f$, $f^*$, can be calculated and a global minimum is searched in this $f^*$. This pseudo global minimum is usually a good starting point to locate the minimum in the original function $f$ [123, 124].
- Statistical Optimization Methods [125]: They use a statistical model of the objective function to bias the selection of new points. Bayesian arguments suppose that the objective function to optimize comes from a class of functions modeled by a stochastic function. Moreover, previous samples provide information of the objective function. This information is used to estimate parameters of the stochastic function and this refined model can be used to bias the selection of points in the search domain.

- Tabu search [126]: According to Glover, the Tabu Search is "'a meta-heuristic superimposed on another heuristic"'. Manly, it is an algorithm that avoids entrainment in cycles by not allowing moves which take the solution to *solution points* already visited, that is, *tabu*.

In the presented traditional techniques, the main goal is precision and certainty [11]. That is, if it is required an increment on the solutions' accuracy, the computational cost of the algorithm will grow in correspondence.

Therefore, though there are many mathematical optimization methods, the difficulties associated to use them on large-scale engineer problems lead us to the development of alternative solutions [127]. To overcome these problems, researchers have proposed evolutionary-based algorithms for searching near-optimum solutions as part of the so called *Soft Computing Techniques*.

### 3.4.3 Soft Computing Techniques

The starting point of Soft Computing techniques is to exploit, wherever possible, the tolerance for imprecision and uncertainty [12]. Soft computing are not only optimization techniques, but a group of techniques that study, model and analyze complex problems. A good example cited in the literature is the case of parking a car. This is an easy task almost for everybody since the final orientation and position are not defined precisely. Otherwise, it would become unmanageable for humans. The main point to highlight here is that the parking task becomes easy for humans when it is defined imprecisely whereas it would be very difficult to solve by traditional methods that do not exploit tolerance to imprecision [11].

Soft computing techniques use human mind as a model and try to formalize the human cognitive process. The main soft computing methods include the fuzzy logic, neuro-computing, evolutionary computing and probabilistic computing. Moreover, the different techniques are complementary rather than competitive and exclusive [42]. Therefore, the more appropriate for this problem is the evolutionary computation (EC) or evolutionary algorithms (EA), since it is a systematic random search addressed to find an optimal solution to a problem.

Evolutionary algorithms (EAs) are stochastic search methods that are inspired in the natural selection, where the individual that survives is the one that fit the best in the biological environment. One of the main differences between classic techniques and EAs is that the last ones involve a search from a group (called *population*) of solutions. There is a competitive selection in each iteration, where the solutions with high *fitness* are recombined with other solutions. Subsequently, these new solutions are also *mutated* by changing some small element. EAs are usually considered as a global optimization methods although their convergence to a global optimum is guaranteed in a weak probabilistic sense. Nevertheless, one of the main strong points of EAS is that they perform well on noisy environments where there may be multiple local optima.

Thus, for complex domains, Evolutionary Algorithms (EA), [128, 129], have proved to be robust and efficient stochastic optimization methods, combining properties of volume and path-oriented searching techniques. The great popularity achieved by this kind of techniques is a consequence of their adequate performance at acceptable cost when applied to a wide range of problems with great amounts of data and parameters. However, it is important to take their lacks into account when apply to a problem of optimization. They are computationally expensive, therefore they are not adequate for optimizations that take place in real-time applications, require adjusting intrinsic parameters and most of all, the optimal solution in finite time is not guaranteed.

A new terminology was introduced in EA paradigm, taken from the Biology, to make reference to basic concepts of classical optimizations. The candidate solution is named "individual", the quality function is called "fitness function" and the problem to solve is the "environment". The evolution process happens in equilibrium between two antagonistic tendencies:

- increase of the genetic diversity in the population;
- decrease of this diversity for the selection mechanism.

In the following figure the evolutionary process is shown. The cycle starts generating a random population, and then, the selection operator chooses the fittest solutions to the problem. The application of some genetic operators produces new solutions, "breeds", and the replacement process generates a new population.

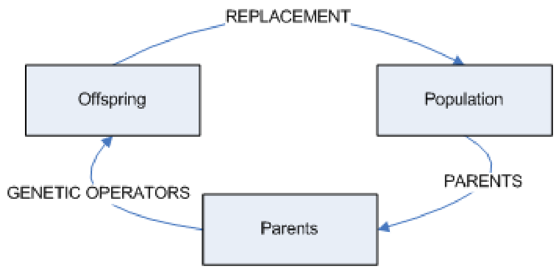


Figure 3.5: Evolutionary wheel

Then, many different types of evolutionary methods were

- Genetic Algorithms (GA) [130]
- Genetic Programming (GP) [131]
- Evolutionary Programming (EP) [132]

- Evolutionary Strategies (ES) [133]

And more rencetly, new types of EAs have been developed (in addition to the improvements in GA) in recent years in order to reduce the processing time and improve the quality of solutions. The most recent techniques are:

- Memetic Algorithms (MAs) [134]
- Particle Swarm Optimization (PSO) [135]
- Ant-colony Systems [136]
- Shuffled Frog Leaping (SFL) [137]

There are many different applications of these algorithms to the optimization problem. For example, T. Back et al. present a work [138] in which they improve the performance of a telecommunication system by means of a new approach on evolutionary algorithms. In the same line, D. Reichelt and F. and Rothlauf [139] use two new evolutionary algorithms to find the most cost-effective communication network design that satisfies a predefined reliability constraint. In some cases, the evolutionary algorithms are used in critical applications, for example, the design of aircraft by means of Genetic Algorithms [140], electromagnetic systems using also Genetic Algorithms [141], [142], the design of a nuclear reactor fuel arrangement optimization [143] or modular robotic arms [144] by means of evolutionary algorithms. Other interesting applications have been successfully solved by Evolutionary Strategies, i.e. the optimal design of shell-and-tube heat exchangers [145] and the vehicle routing problem [146].

In particular, this thesis focuses on the application of adjusting the parameters that regulate the detector and tracking system. We face a high-dimensional problem with non-linear dependence among the variables or parameters [147].

Evolutionary Strategies (ES) [133, 148, 149] are chosen for this application [147] since the function landscape is very irregular, with plenty of local optima (there are many "‘reasonable"’ combination of parameters) and they have proved to have a good performance in this type of problems. For example, in [147], the authors present an interesting application of Evolutionary Strategies for the design of tracking filter with a large number of specifications and constraints, in particular, the design of Interactive Multiple Model (IMM) filters.

Moreover, Evolutionary Strategies (ES) [133, 148, 149] are the evolutionary algorithms specifically conceived for numerical optimization, and they have been successfully applied to engineering optimization problems with real-valued vector representations [150]. They combine a search process, which randomly scans the feasible region (exploration), and local optimization along certain paths (exploitation), achieving very acceptable rates of robustness and efficiency. Each solution to the problem is defined as an individual in a population, codifying each individual as a couple of real valued vectors: the parameters and a standard deviation of each parameter used in the search process.

A general ES is defined as an 8-tuple [150]: $ES = (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda)$ Where $I = (\vec{x}, \vec{\sigma}, \vec{\alpha}) = \Re^n \times \Re_+^{n_\sigma} \times [-\pi, \pi]^{n_\alpha}$ is the space of individuals, $n_\sigma \in \{1, ..., n\}$ and $n_\sigma \in \{0, (2n - n_\sigma)(n_\sigma - 1)/2\}$, $\Phi : I \rightarrow \Re = f$, is the fitness function, $\Omega = \{m_{\{\tau, \tau', \beta\}} : I^\lambda \rightarrow I^\lambda\} \cup \{r_{\{rx, r\sigma, r\alpha\}} : I^\mu \rightarrow I^\lambda\}$ are the genetic operators, mutation and crossover operators. $\Psi(P) = s(P \cup m_{\{\tau, \tau', \beta\}}(r_{\{rx, r\sigma, r\alpha\}}(P)))$ is the process to generate a new set of individuals, $s$ is the selection operator and $\iota$ is the termination criterion. In this work, the definition of the individual has been simplified: the rotation angles $n_\sigma$ have not been taken into account, $n_\sigma = 0$.

Mutation operator generates new individuals as follows:

$$\sigma_i^{'} = \sigma_i \cdot \exp(\tau^{'} \cdot N(0, 1) + \tau \cdot N_i(0, 1)) \tag{3.33}$$

$$\vec{x}^{'} = \vec{x} + \sigma_i^{'} \cdot \vec{N}(\vec{0}, 1) \tag{3.34}$$

The general outline of ES is showed in the figure 3.6 .

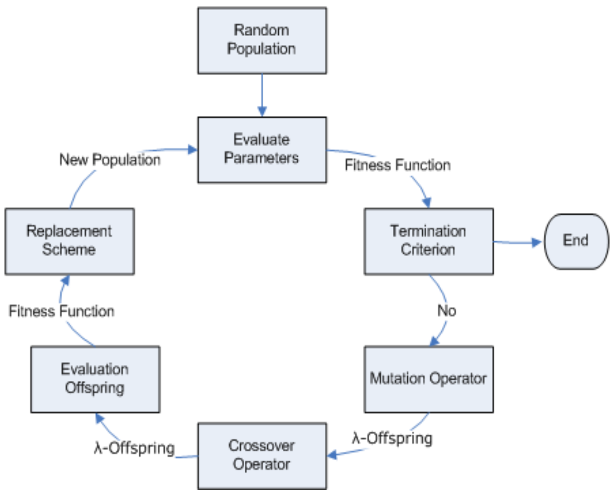


Figure 3.6: General outline of ES

ES have several formulations, but the most common form is $(\mu, \lambda) - ES$, where $\lambda > \mu \geq 1$, means that $\mu$-parents generate $\lambda$-offspring through crossover and mutation in each generation. The best $\mu$ offspring are selected deterministically from the $\lambda$ offspring and replace the current parents. ES consider that strategy parameters, which roughly define the size of mutations, are controlled by a "self-adaptive" property. An extension of the selection scheme is the use of elitism; this formulation is called $(\mu + \lambda)$-ES. In each generation, the best $\mu$-offspring of the set $\mu$-parents and $\lambda$-offspring replace current parents. Thus, the best solutions are maintained through generations. The computational cost of $(\mu, \lambda)$-ES and $(\mu + \lambda)$-ES formulation is the same.

Until here, the author has focused on the optimization problem. But this work not only searches the parameters for a effective performance of a tracking system but for the effective performance for the more general number of cases. Thus, the next section talks about the generalization problem.

### 3.4.4 Generalization

During the optimization stage, parameters are adapted to have the best results for the fitness function given the inputs in the training set. Nevertheless, this is not the only goal; another main purpose is to obtain good results for input data that are not likely in the training set, that is, generalization.

One of the indispensable requirements is to have some previous knowledge of the relevant inputs-outputs in order to know what must be learned [151], [152].

For the generalization purpose, two important conditions must be considered:

- The function to learn (the function that assigns inputs to correct outputs) must not be abrupt in the sense that it has to have similar behaviour for neighbour areas.
- The training data set must be large and representative enough in order to cover the set of cases that the system is interested in generalizing. The generalization that the system can carry out can be classified as interpolation and extrapolation. Interpolation applies to the scenarios close to one or more cases of the training cases (here we see the importance of the non abrupt or relatively smooth function to learn). Whereas extrapolation are those cases which are not similar at all to ones in the training sequences. The resultant output given by the function in the case of interpolations is considered reliable in contrast to the extrapolation outcome.

All the information regarding the function to learn must be included in the learn process. For example, if the threshold in the detector process must be above 10 units or otherwise the system would considered all noise as moving targets, then, this limitation is introduced to the function so that the system never allows forbidden inputs.

Another critical point is how to avoid over-fitting, that is, how to avoid that the optimizer select the parameters ad-hoc for the scenarios of the training set. Again, the initial set of training video sequences have a relevant role. This training set must be carefully selected in the way that the scenarios must be different enough to avoid the system from over-fitting. Thus, the set of examples used to design, adjust and train the system should produce a general solution of the tracking system. A small set of examples can lead to learning techniques that adapt exactly to these specific examples (over-fitting), with the consequent loss of generality. On the contrary, random selected examples could produce disorientation in the search. The set of data that optimize the search of the suitable parameters is defined as the *ideal trainer* [153].

As a result of this, it is important to highlight that surveillance systems must work under changing conditions: different targets of distinct shapes, distance to the focus, velocity, appearance (individually or in groups), number of targets, etc. Moreover, generalization is the capability that makes the system work regardless of the scenario. For example, in the case of tracking people, surveillance systems must track one person, two, three or a group of them, in a wide range of distances to the focus, if they are stood or on their knees, if they carry an object, ... This is what is known as generalization, in this case, applied to the surveillance systems technology.

There are many papers on the literature that deal with the problem of generalization, especially in the field of neural networks. For example, the work [154] studies how to generalize on statistically neutral problems (a problem is statistically neutral if the probability of mapping an input onto an output is always the chance value of 0.5). The authors highlight that *the size of the training set and the choice of the distribution function by which the training data are selected directly influences the performance of any learning machine.* Moreover, this work pointed that there are many methods for estimating the generalization accuracy.

- Bagging: Bagging predictors is a technique to generate several versions of a predictor. These versions are used to obtain an aggregated predictor. Each of the versions are computed by making bootstrap replicates of the learning set and using these as new learning sets.

- Stacking: Stacked generalization deduces the biases of certain generalizer(s) with respect to a provided learning set. This deduction is carried out by generalizing in a second space in which the inputs are, for example, the guesses of the original generalizers when taught with part of the learning set and trying to guess the rest of it, and whose output is, for example, the correct guess. If multiple generalizers are used, stacked generalization can be interpreted as a version of cross-validation in which a more sophisticated strategy than cross-validation's crude winner-takes-all for combining the individual generalizers is utilized. On the other hand, if a single generalizer is used, stacked generalization is a technique for estimating the error of a generalizer which has been trained on a particular learning set and then asked a particular question [152].

- Boosting: Boosting refers to a general effective method of producing an accurate prediction rule by combining rough and moderately inaccurate rules of thumb. This approach is based on the fact that learning a highly accurate rule is a very difficult task, whereas it is not hard to obtain easy and rough rules of thumb that are moderately accurate. Iteratively, the algorithm finds certain "weak" rules and after many rounds, the boosting algorithm combines this "weak" rules to generate a single prediction rule that hopefully is more accurate and precise that any of the rules of thumb.

In order to measure the quality of this generalization and not having a over-fitted evaluation, there are two well known methods:

- Cross-validation: In k-fold cross-validation, the data are divided into k subsets of approximately equal size. The classifier is trained k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute whatever error criterion is used.
- Split-sample: A single subset (the validation set) is used to estimate the generalization error, instead of k different subsets.

The optimization technique applied in this work, not only fit the values of the parameters for a proper performance, but search the solution that provides a generalization in the performance so that they system can work in the widest range of conditions for a specific environment. The generalization part of this technique can be identified as a kind of simplification of "Stacking Generalization" in which *for any real-world learning set $\omega$, there are always many possible generalizers $G_j$ one can use to extrapolate from $\omega$ [155].* Then, this work would split the training data in several parts and apply the optimization process to each of them. Subsequently, the results of each of the parts are combined and used as an input for the next iteration. This is repeated until the optimization process converges.

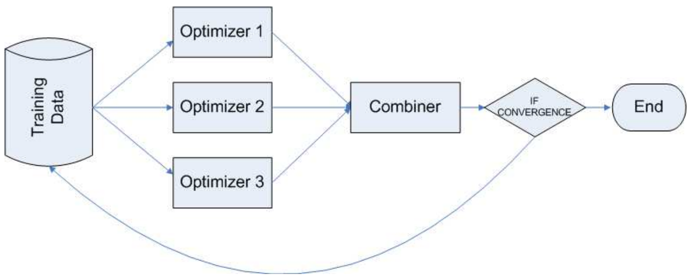


Figure 3.7: Process of generalization

Finally, once the parameters of the video tracking system have been learned for proper optimization and generalization taking as a fitness function some evaluation metric, we can trust on the output and measurements given by such tracking system. This output will be used for a specific application. In this thesis, the author proposes a recognition of human activities. The next subsection gives an overview on this topic.

## 3.5 Human Activity Recognition

The third and last aspect of this work is focused on the recognition of activities by analyzing the features extracted from a video tracking process, with a special emphasis on data modeling and classifier performance [156–161]. In particular, the author is going to focus on human activity recognition.

This process of action recognition is usually divided into three steps that are closely related one another:

1. Definition of the activities to sort depending on the application. According to the authors in [161], human movements can be considered at different temporal levels: analysis of body parts, single people activities or large-scale interactions. As a result of this, systems deal with different activities and temporal integration times.
2. According to the activities to classify, make the correct selection of measurements and computation of features.
3. Generation of a classifier by using the measurements and features as training and validation sets of examples.

Many solutions have been proposed in this field since this topic is still and open and challenging problem to be solved by the research community. The next sections briefly show some of the most highlight solutions presented in the literature.

### 3.5.1 Selection of Features

According to the classification carried out by the authors in [161], there are three major approaches for human classification:

1. 3D models: construction of 3D models of the different poses. For example, the authors in [162] build an ellipsoid as an approximation of a 3D shape of a human body. Moreover, this work detects the number of individuals in a group by locating the head of each of them. In addition, they present a hierarchical locomotion model. In a first level, the activities to distinguish (running, walking and standing) are represented by a a finite state machine where the speed of the body is the feature to model this classifier. In a second level the limbs motions of walking and running are captured taking into account that both are cyclic movements. The only feature computed for subsequent classification is the distance $D$:

$$D = \sum_{(x,y)\in\Omega} \frac{|\bar{v}_{xy} - \bar{m}_{xy}|}{\sqrt{|\bar{v}_{xy}|\,|\bar{m}_{xy}|}} \tag{3.35}$$

where $\Omega$ is the area within the bounding box of both legs, $\bar{v}$ is the optical flow, and $\bar{m}$ is a predicted motion template (the authors compute 32 motion templates).

2. Appearance models: they are based on the extraction of a 2D shape model that must be classified by comparison with a trained one.

   For example, in [7], the authors compute the silhouette and a texture template for each person. Then, bearing in mind that elbows, hands, head and feet lie on the silhouette boundary, the system localizes and tracks each of these body parts. Then, the authors observe that for four postures (standing, bending, lying and sitting), the selected body parts have very different locations. The classification is done by means of a hierarchical body posture analysis that compute the similarities of horizontal and vertical projection histograms of the detected silhouette and the main postures. The body posture that gives the highest similarity measure is chosen as the estimated posture.

   In the next equation, $S_i$ represents the similarity between the detected silhouette and the $i-th$ main posture, $H_i$ and $V_i$ the horizontal and vertical projections of the $i-th$ main posture, and $P$ and $R$ the horizontal and vertical projections of the detected silhouette. Then, the similarity is calculated as:

$$S_i = -log \sum_{h}^{128} \sum_{v}^{128} (H_h^i - P_h)^2 + (V_v^i - R_h)^2 \tag{3.36}$$

   The system determines the posture by taking the highest score.

3. Motion models: they rely on people motion measurements. The most primitive level for the recognition of any activity is the analysis of the movement, characterized by a space-time trajectory [163].

   In [164], the analysis is carried out by extracting motion features with which to compute the motion patterns to recognize a set of activities. The authors compute 29 features and subsequently filtered in order to select the most important ones for the final classification. The activities are 5: inactive, active, walking, running and fighting. Thus, the features are computed divided into 8 categories:

   - Instantaneous (i.e. speed):

$$v(t) = \|\bar{v(t)}\| = \sqrt{v_x^2 + v_y^2} \tag{3.37}$$

   - Time averaged over T frames (i.e. mean speed):

$$v(t) = \bar{v(t)} = \frac{1}{T} \sum_{i=t-T+1}^{t} v(i) \tag{3.38}$$

- Temporal energy 2nd order moments (i.e. speed) :

$$\sigma_v^2(t) = \frac{1}{T-1} \sum_{i=t-T+1}^{t} v^2 \tag{3.39}$$

- Instantaneous optical flow:

$$F(t, \bar{x}, y) = (f_x(t,x,y), f_y(t,x,y)), (x,y) \in P(t) \tag{3.40}$$

where $P(t) = (x,y) \in \Re^2 | (x,y) \in target(t)$

- Spatial energy 2nd order moments of the optical flow (i.e. motion energy):

$$f(\bar{t})_i = \frac{1}{N(t)} \sum_{(x,y)\in P(t)} f_2(t,x,y)_i, i = 1,2 \tag{3.41}$$

- Time averaged over T frames (i.e. motion energy)

$$f(\bar{T})_i = \frac{1}{T} \sum_{i=t-T+1}^{t} f(\bar{i}) \tag{3.42}$$

- Temporal energy 2nd order moments of the optical flow (i.e. mean flow)

$$\sum_{\bar{F}}(t)_i = \frac{1}{T} \sum_{j=t-T+1}^{t} F(\bar{j})_i F(\bar{j})_i^T, i = 1,2,3,4 \tag{3.43}$$

They use a Bayesian classifier and model the likelihood functions as Gaussian mixtures.

Other interesting work it is the one presented in [165], where the authors represent the motion information as a feature image by using a using an Infinite Impulse Response (IIR) filter. A weighted average at time $i$, $M_t$, is computed as:

$$M_t = \alpha I_{t-1} + (1-\alpha) M_{t-1} \tag{3.44}$$

where $I_t$ is the image at time $i$, $\alpha \in [0,1]$.

Then, the feature image at time $i$ is computed as:

$$F_t = |M_t - I_t| \tag{3.45}$$

They obtain good results for the classification of activities like walking, running, skipping, hopping, etc.

Once the features have been collected, the next step is the generation of a classifier.

### 3.5.2 Statistical Learning Classifiers

As it was said by the author in the Introduction, one important distinction among classifiers lie in the use or not of statistical learning. The principal advantage of using statistical classifiers is the knowledge of the error and success rate since this rate is modeled by the corresponding probabilities.

One of the main aspects of this probabilistic learning is the introduction of latent variables associated to the observed samples. These latent variables reduce the complexity in describing the observed samples.

In order to connect the latent and observed variables a model is needed. Again, according to [18], a model family is composed of a conditional probability distribution $P(A|B, \theta)$, where $A$ and $B$ are disjoint variables, $B$ can be an empty group and $\theta$ is a latent variable linked to the group. Model families are usually represented by groups of distribution $P(A|B, \theta)|\theta \in \Theta$, where the models are the elements represented by conditional distributions $A|B$ indexed by $\theta$.

We can distinguish between two main types of learning algorithms: supervised and unsupervised learning. Thus, in supervised learning, if the joint probability $p(x, y)$ (where $x \in X$ are the set of samples or observations and $y \in Y$ are the labels such as $(x_i, y_i)|i = 1..n)$ can be approximated by means of a parametric family model such as $p_\theta(x, y), \theta \in \Theta$, a classifier is obtained by estimating the class-conditional densities and then classifying each new income sample to the class with the highest posterior probability. This is called *generative classifiers.*

That is, the generative approach attempts to obtain the way in which the observed data $x$ are generated from given classes $y$ by specifying a prior distribution for each class and a class-conditional distribution over the features. Then, this approach makes the predictions by using Bayes Rules [166]:

$$p(x, y) = p(x|y)p(y) \tag{3.46}$$

On the other hand, *discriminative classifiers* aims to find the classification rule with the smallest error rate. This depends only on the conditional density $p(y|x)$, without assuming any knowledge about the input distribution $p(x)$. Then, discriminative classifiers model the posterior probability $p(y|x)$ or directly map from inputs $x$ to the class labels [166].

The two main examples of these classifiers are logistic regression (discriminative) and Naive Bayes (generative) classifiers.

While supervised learning has a defined goal, there are no such criteria for unsupervised learning. Here, the interest consists of finding interesting structures within a sample $x|i = 1..n$ independently formed from a unknown distribution $P(x)$. According to Occam's razor, the objective is to search structures that are inherently simple but difficult to see due to the unpredictably random noise. Then, the problem seems to be harder than the supervised problem as the algorithm must find latent variables suitable for effective compression of the function $P(x)$. Therefore, unsupervised methods performs density estimation and it is very common to use generative models for $P(x)$. For example, two important approaches for density estimation are:

1. Mixture of models
2. Kernel Density Estimation (KDE)

The next two sections will explain these common approaches of probability density estimation:

### Mixture Models

*Finite mixtures naturally model samples which are assumed to have been produced by one (random selected and unknown) of a set of alternative random sources* [167]. Then, calculating the parameters of these sources and inferring which source produces each sample leads to a clustering of a set of observations.

$$p(\mathbf{x}|\theta) = \sum_{m=1}^{M} \alpha_m p_m(\mathbf{x}|\theta_m) \tag{3.47}$$

where

- $\alpha_1, .., \alpha_M$ are the mixing probabilities which must accomplished:

$$\alpha_m \geq 0, m = 1, .., M \tag{3.48}$$

and

$$\sum_{m=1}^{M} \alpha_m = 1 \tag{3.49}$$

- each $\theta_m$ is the parameters that define the $m - th$ component $p_m$
- and $\theta_1, .., \theta_M, \alpha_1, .., \alpha_M$ needed to define the mixture of functions.

Then, log-likelihood expression is given by:

$$log(\ell(\Theta|X) = log\prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} log(\sum_{m=1}^{M} \alpha_m p_m(x_i|\theta_m)) \tag{3.50}$$

which is difficult to optimize because it contains the log of the sum. Therefore, if $X$ is considered as incomplete and assumed the existence of $Y = y_i{}_{i=1}^{N}$ whose values identify which component from the mixture generated each sample, the likelihood expression becomes simpler. Thus, we assume that $y_i \in 1..M$ for each $i$, and the $i^{th}$ sample was generated by the $k^{th}$ mixture component. If we know the values of $Y$, the likelihood will be [168]:

$$log(\ell(\Theta|X,Y) = log(P(X,Y|\Theta) = \sum_{i=1}^{N} log(P(x_i|y_i)P(y)|\Theta) = \sum_{i=1}^{N} log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})) \tag{3.51}$$

The problem is that the values of $Y$ are unknown. But if it is assumed that $Y$ is a random vector and we can derive an expression for the distribution of unobserved data by using Bayes' rule:

$$p(y_i|x_i,\Theta^g) = \frac{\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})}{p(x_i,\Theta^g)} = \frac{\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})}{\sum_{m=1}^{M} \alpha_m^g p_m(x_i|\theta_m^g)} \tag{3.52}$$

and

$$p(\mathbf{y}|X,\Theta^g) = \prod_{i=1}^{N} p(y_i|x_i,\Theta^g) \tag{3.53}$$

where $\mathbf{y} = (y_1,..,y_N)$ is the sequence of the unobserved data.

Then, the maximum complete-likelihood (ML) estimated will be:

$$\Theta_{ML} = argmax_\theta log(Y|\theta) \tag{3.54}$$

The standard method used to find the proper parameters of the mixture of function is the expectation-maximization (EM) algorithm [169] [53] which converges to a maximum-likelihood (ML) [168]. EM iteratively computes an expectation of the likelihood taking into account the latent variables as if they were observed (E step) and afterward, it calculates the maximum likelihood estimates of the parameters by maximizing the expected likelihood of the previous step (M step).

Finally and before to explain the basis of the EM algorithm, it is important to highlight that mixture models are not only useful for unsupervised applications but are capable to represent complex probability density functions (pdf's) which make them very useful for modeling, for example, complex class-conditional pdf's in Bayesian classifiers.

### Kernel Density Estimation Model (KDE)

One alternative for the training of mixtures of Gaussians without ground truth is the kernel density estimation techniques (KDE).

The aim of KDE is to approximate the true probability density function of a phenomenon from the set of measured data produced by this phenomenon [170]. A kernel function is generated around each sample $x_j$ in the training set. The probability density function is computed by adding all these kernel functions.

More formally, kernel estimators smooth the contribution of each observed data point over a local neighborhood of that data point. The contribution of data point $x_i$ to the estimate at some point $x^*$ depends on how apart $x_i$ and $x^*$ are. The extent of this contribution is dependent upon the shape of the kernel function adopted and the width (bandwidth) accorded to it. If we denote the kernel function as $K$ and its bandwidth by $h$, the estimated density at any point $x$ is:

$$f(x) = p_{KDE}(x) = \frac{1}{N \cdot h} \sum_{j=1}^{N} K(\frac{x - x_j}{h}) \tag{3.55}$$

where $h$ is the *kernel* bandwidth (or smoothing factor). More over $\int K(t)dt = 1$ to ensure that the estimates $f(x)$ integrates to 1 and where the kernel function, K, is usually chosen to be a smooth unimodal function with a peak at 0.

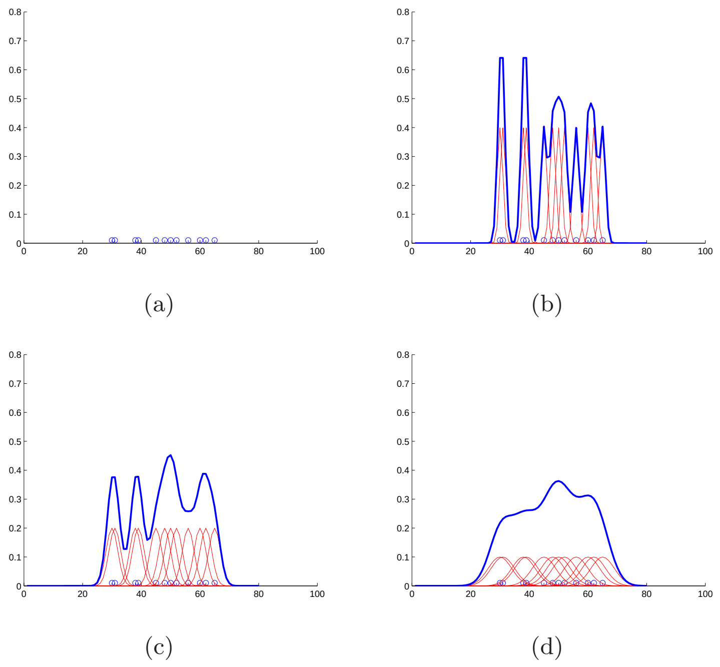
The most typical kernels are Gaussians; nevertheless some other functions can be chosen, leaving this decisition to the researcher. Some kernels might be (where $1_{(t)}$ denotes 1 when $t$ holds and 0 when $t$ is false):

- Uniform

$$\frac{1}{2} 1_{(|t| \leq 1)} \tag{3.56}$$

- Triangle

$$(1 - |t|) 1_{(|t| \leq 1)} \tag{3.57}$$



(a) (b) (c) (d)

Figure 3.8: Example of KDE where the kernel are Gaussians with h equal to b) 1, c) 2 and d) 4

- Epanechnikov

$$\frac{3}{4}(1-t^2)1_{(|t|\leq 1)} \tag{3.58}$$

- Quartic

$$\frac{15}{16}(1-t^2)^2 1_{(|t|\leq 1)} \tag{3.59}$$

- Triweight

$$\frac{35}{32}(1-t^2)^3 1_{(|t|\leq 1)} \tag{3.60}$$

- Gaussian

$$\frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}u^2} 1_{(|t|\leq 1)} \tag{3.61}$$

- Cosine

$$\frac{\pi}{4} cos(\frac{\pi}{2}u) 1_{(|t|\leq 1)} \tag{3.62}$$

The quality of a kernel estimate depends less on the shape of the $K$ than on the value of its bandwidth $h$. It's important to choose the most appropriate bandwidth as a value that is too small or too large is not useful. For small values of $h$ the estimates are very pointed whereas for higher $h$ values the estimates become smoothing.

**Expectation Maximization (EM) Algorithm**

The EM is a method to find the maximum-likelihood (ML) estimate of the parameters of an underlying distribution given a some data when the data is incomplete [168]. For the case of Gaussian mixture (the most common case), the convergence of EM is well studied.

The EM computes a sequence of estimates $\theta^i$ where $i = 1, 2, ..$ by alternatingly applying two steps:

- **E-step** finds the expected value of the complete log-likelihood $p(X, Y|\Theta)$ with respect to the unknown data $Y$ given the observed data $X$ and the current parameter estimates. The result is the so-called Q-function [168]:

$$Q(\Theta, \Theta^{(i-1)}) = E[logp(X, Y|\Theta)|X, |\Theta^{(i-1)}] \tag{3.63}$$

  where $\Theta^{(i-1)}$ are the current parameters estimates which are used to evaluate the expectation and $\Theta$ are the new parameters that we optimize to increase $Q$. In this expression, $X$ and $\Theta^{(i-1)}$ are constants, whereas $\Theta$ is a normal variable to adjust and $Y$ is a random variable defined by $f(y|X, \Theta^{(i-1)})$. In the best of cases, this distribution is a simple function of the assumed $\Theta^{(i-1)}$, but it could be also a difficult density hard to compute.

- **M-step** maximizes the expectation which was computed in the first step.

$$\Theta^{(i-1)} = argmax_{\Theta} Q(\Theta, \Theta^{(i-1)}) \tag{3.64}$$

These two steps are repeated as necessary. Moreover, the algorithm assures that each iteration increases the log-likelihood and the algorithm itself converges to a local maximum of the likelihood function.

In the particular case that Gaussian mixtures are used to model the probability density function of a random variable, $p(x)$, as:

$$p(x) = \sum_{l=1}^{M} = \alpha G(x; \mu_l; \sigma_l^2) \tag{3.65}$$

with $G(\cdot)$ the Gaussian function and $M$ the number of Gaussian components. We consider here the univariate case for the sake of simplicity of notation, but we will eventually extend results to the multivariate case. The parameterization of such a GM requires the estimate of the weight, $\alpha$ , mean, $\mu$ , and variance, $\sigma^2$, for each of the $M$ Gaussian components (the sum of weights has to be unitary). This estimate is typically performed so as to maximize the likelihood, $L$, over a set of samples, $x_i$, $i = 1, ..., N$, independently drawn from this distribution:

$$L(x_1, ..., x_N) = \prod_{i=1}^{N} p(x_i) \tag{3.66}$$

Operationally, the likelihood is conveniently computed in log form with the main advantage of avoiding rapid underflow. Maximization of 3.66 can be obtained by estimating the GM parameters through the expectation steps. The equations used to estimate the parameters at each iteration are:

$$\alpha_l^{new} = \frac{1}{N} \sum_{i=1}^{N} p(l/x_i, \Theta) \tag{3.67}$$

$$\mu_l^{new} = \frac{\sum_{i=1}^{N} x_i p(l/x_i, \Theta)}{\sum_{i=1}^{N} p(l/x_i, \Theta)} \tag{3.68}$$

$$\sigma_l^{2\ new} = \frac{\sum_{i=1}^{N} (x_i - \mu_l^{new})^2 p(l/x_i, \Theta)}{\sum_{i=1}^{N} p(l/x_i, \Theta)} \tag{3.69}$$

where $\Theta$ represents the current set of parameters and $p(l/x_i, \Theta)$ is simply the probability of the $l$-th Gaussian component at sample $x_i$:

$$p(l/x_i, \Theta) = \frac{\alpha_l G(x_i; \mu_l, \sigma_l^2)}{\sum_{k=1}^{M} \alpha_k G(x_i; \mu_k, \sigma_k^2)} \tag{3.70}$$

Equation 3.70 is often referred to as a membership function, expressing the membership of $x_i$ in each of the Gaussian components, and its computation is the expectation step of an EM iteration. The computation of update equations (3.67-3.69) is its maximization step. It is important to note that each of (3.67-3.69) provides an optimum for the respective parameter independently of the other two.

However, it is important to highlight that the EM algorithm has several drawbacks:

- it is very sensitive to initialization and as a result of this, it might converge to a local maxima of the log-likelihood.
- it might converge to the boundary of the observation space leading to meaningless estimates.

Some works have dealt with these drawbacks successfully. For example, in [171] and [172] the authors proposed splitting and merging iteratively certain components of the mixtures in order to obtain a better value of log-likelihood.

Another additional problem with the use of EM algorithm is a general problem of the unsupervised learning: how to define the quality of clustering. The next section explains a possible solution to this problem.

### Clustering and Labeling: Semi-supervised Learning

Usually, clustering is used as an exploratory method to find hidden structures not known by the researcher. These methods are in many cases pure algorithmic and do not define quality measures.

Thus, the objective, and the measure of quality, are the same as in classification: to match observations or measurements to labels. Therefore, unsupervised classification might be a bit contradictory. For example, supposing the clusters are well separated and compact, it could be thought that the classification can be done correctly by using the boundaries as delimitation among classes. But without any labeled data, it is very difficult to match which labels correspond to which cluster. Nevertheless, in some cases, unsupervised classification is in principle possible [48]. The main difficulty in matching labels to clusters is due to the different conceptions: class labels are usually arbitrary identifiers whereas a learning problem arises when the classes are not arbitrary but have intrinsic content with observable distributional reflexes [48]. In the particular case of this work, if there is a intrinsic meaning in the features or measurements used to cluster the activities, it would be possible to label the clusters once they are identified (a method of semi-supervised learning). Thus, the author will find measurements that reflects this behavior.

Then, a clustering algorithm learns a function $c(x)$ that assigns cluster labels to instances. In order to obtain a classifier $f(x)$ a mapping $\mu$ is required from cluster labels to class labels:

$$f(x) = \mu(c(x)) \tag{3.71}$$

Finally, the quality of $f$ is measured as generalization error. Moreover, the quality of $c$ is defined as the quality of the best classifier generated from $c$ over all the choices of $\mu$

This is the simplest idea of clustering by means of a semi-supervised way: cluster and label. A brief definition of semi-supervised learning (SSL) is halfway between supervised and unsupervised learning. It uses manly unlabeled data and in addition, some supervised samples. There is plenty of literature on this topic, like for example [48] and [173].

### Hidden Markov Models (HMMs) Classifiers

Human activity recognition can be thought as the classification or clustering of time varying feature data [20]. Therefore, the problem of activity recognition is a learning task in which training samples are needed in order to build a model that is able to recognize the type of activity carried out by the object and its dynamic evolution along time. The are many works on this field [163] and the most remarkable are the approaches that take into account the time-varying data:

- Dynamic time warping (DTW) [36,37]: It measures similarity between two sequences being capable of overcoming differences of velocity or accelerations between the template and the sequence [174].
- Hidden Markov Models (HMM) [175–177]: It is a stochastic state machine in which the system is modeled as a Markov process with unknown parameters [21].
- Time-delay neural network (TDNN): It adds delay units to a general static network and some of these previous samples are used to predict then next value. It has been applied very successfully to gesture recognition [178].
- Syntactic techniques: The syntactic techniques have been manly applied to patter recognition tasks. Nevertheless, some works have been carried out in the field of behavior recognition [179].
- Finite-state machine (FSN): It is a model of behaviors made up of several finite states. The states are used to match the reference with the sequence test [180].
- Non-deterministic finite automaton (NFA): In this work [181], the authors present an approach that is based on feasible assumptions about the present behaviors consistent with the input image. Behavior models are dynamically generated and verified by finding their supporting evidence in input images. This can be realized by an architecture called the selective attention model, which consists of a state-dependent event detector and an event sequence analyzer.

Human actions are represented by sequences of states which must be inferred from features extracted from images (shapes, body parts, poses, etc.) and attributes extracted from motion such as velocity, trajectory, etc. In many real cases, the values of the state variables cannot be directly measured, due to noise and other non-idealities such as occlusions and illumination changes, and have to be estimated from the available observations. Hidden Markov models and their several variations have been used extensively for activity recognition since they provide a smoothed estimate of the state values [22, 168, 182].

Thus, the two strong points of HMMs are that they build a time pattern of the activities to recognize and, they do not need the direct measurement since they can estimate them from the available estimation.

A hidden Markov model (HMM) is fully described by three sets of quantities: the state transition probabilities, $A$; the emission (or observation) probabilities, $B$; and the probabilities of the initial states, $\pi$. The states are only allowed to assume discrete values; let us say, N. Therefore, $A$ can be represented by an N x N matrix and by an N-dimensional vector. Instead, observations are often drawn from continuous variables and, as such, emission probabilities need to be modeled by probability density functions.

Thus, HMMs offer a fashion to estimate the joint probability of a sequence of time-discrete observations $O_t, t = 1..T$, and corresponding hidden states, $X_t 1..N$ [168]. The model is fully described by the set of parameters $\lambda = A, B, \pi$:

The coefficient matrix $A$ expresses the Markovian hypothesis that the value, $i$, of the current state, $X_t$, is only dependent on the value, $j$, of the previous state, $X_{t-1}$

$$A = \{a_{ij}\} = p(X_t = i/X_{t-1} = j) \forall i, j \tag{3.72}$$

$B$ quantifies the probability of observing value $o_t$ when the current state is $j$

$$B = \{b_j(o_t)\} = p(O_t = o_t/X_t = j) \forall o_t, j \tag{3.73}$$

Finally $\pi$ quantifies the probabilities of values for the initial state.

$$\pi = \pi_i = p(X_1 = i) \forall i \tag{3.74}$$

The most common approach for modeling emission probabilities is by use of mixtures of Gaussians [182]. In such a case, $B$ is fully described by the weights, means and covariances of all the Gaussian components. Once given one or more sequences of observations, the **Baum-Welch algorithm** [23] can be used for learning a corresponding HMM with maximum likelihood. This algorithm is an expectation-maximization algorithm that learns $A$, $B$ and in a simultaneous manner and is guaranteed to converge to a local optimum in the parameter space.

$$b_i(o_t) = \sum_{l=1}^{M} \alpha_{il} G(o_t; \mu_{il}, \sigma_{il}^2) \tag{3.75}$$

The Baum-Welch algorithm provides update equations for the iterative optimisation of the model's parameters. Herewith, we concentrate on $B$. First, similarly to Equation 3.70, we pose:

$$p_i(l/o_t, \Theta) = \frac{\alpha_{il} G(o_t; \mu_{il}, \sigma_{il}^2)}{\sum_{k=1}^{M} \alpha_{ik} G(o_t; \mu_{ik}, \sigma_{ik}^2)} \tag{3.76}$$

to express the probability of the $l-th$ component in the GM of state $i$. Then, the weights, means and variances of the emission probabilities are obtained in a way similar to 3.67-3.69 over the set of observed values, $o_t, t = 1..T$. The basic difference is that the terms in the numerators and denominators are multiplied by the probability of being in state $i$ at time $t$, $\gamma_i(t)$. Thus, the term $\gamma_i(t)$ is the main difference between the plain EM and the Baum-Welch algorithm.

$$\alpha_{il}^{new} = \frac{\sum_{t=1}^{T} p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \tag{3.77}$$

$$\mu_{il}^{new} = \frac{\sum_{t=1}^{T} o_t p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} p(l/o_t, \Theta)\gamma_i(t)} \tag{3.78}$$

$$\sigma_{il}^{2\ new} = \frac{\sum_{t=1}^{T}(o_t - \mu_{il}^{new})^2 p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} p(l/o_t, \Theta)\gamma_i(t)} \tag{3.79}$$

In turn, $\gamma_i(t)$ can be expressed from the current estimates of $A$ and $B$ (see [168] for details).

Alternatives to the use of mixtures of Gaussians (GMs) for modeling the emission probabilities have been proposed in the literature. Bourlard and Morgan in [183] proposed to replace the Gaussian mixtures by Artificial Neural Networks in a hybrid ANN/HMM model. A number of variations on hybrid ANN/HMM models is presented by Trentin in [184]. In a recent work [185], Krüger et al. proposed to replace the Gaussian mixtures with mixtures of Support Vector Machines. However, these approaches typically train the emission probabilities in a supervised manner, requiring knowledge of the ground-truth values of the hidden state variable. Even though they may potentially achieve higher accuracy than maximum-likelihood methods, they cannot be applied in the general case where state ground truth is not available. In the contrary, maximum-likelihood HMMs can be trained just with a sequence of observations that are, by definition, available.

Until this point, the most relevant techniques and works carried out on the fields of interests for this thesis have been reviewed in this section. Then, next chapters will present the proposals suggested by the author in this dissertation.

Chapter 4

# Evaluation with Minimum Ground Truth

## 4.1 Motivation

This thesis proposes a technique to compute detailed evaluations of video tracking systems and defines for that figures of merit for measuring accuracy and robustness. These figures of merit are based on a minimal amount of reference data (ground truth) and do not rely on color features. In addition, this method allows the assessment of conclusions supported by numerous samples, due to the possibility of analyzing a statistically significant number of recorded video sequences. This evaluation with many samples is other advantage over conventional systems, which would require a large amount of expensive ground-truth data to derive analogous conclusions about performance.

The basis of this methodology is that interesting objects many times do not move randomly but, under normal conditions, they follow quite regular patterns. This was a conclusion to which the author came after observing many of the videos used in our experiments for surveillance purposes.

Thus, the proposed evaluation technique is based on video sequences in which targets follow a regular path. That does not mean that the tracker would work only for this kind of situations. On the contrary, the tracker must follow any kind of movement or path that a target carries out, but for the tracker evaluation, the user preselects only the videos where the target follows this regularity.

Figure 4.1 shows some examples of videos used by the author for experiments in which the targets follow regular patterns. For example, in figure 4.1(a), individuals usually walk in the footpath (see figure 4.1(b)). In figure 4.1(c), the camera is set in a room where there are plenty of tables and individuals follow the paths between tables as figure 4.1(d) shows. Finally, the example 4.1(e) show an airport environment where the airplanes move along the apron (figure 4.1(f)).

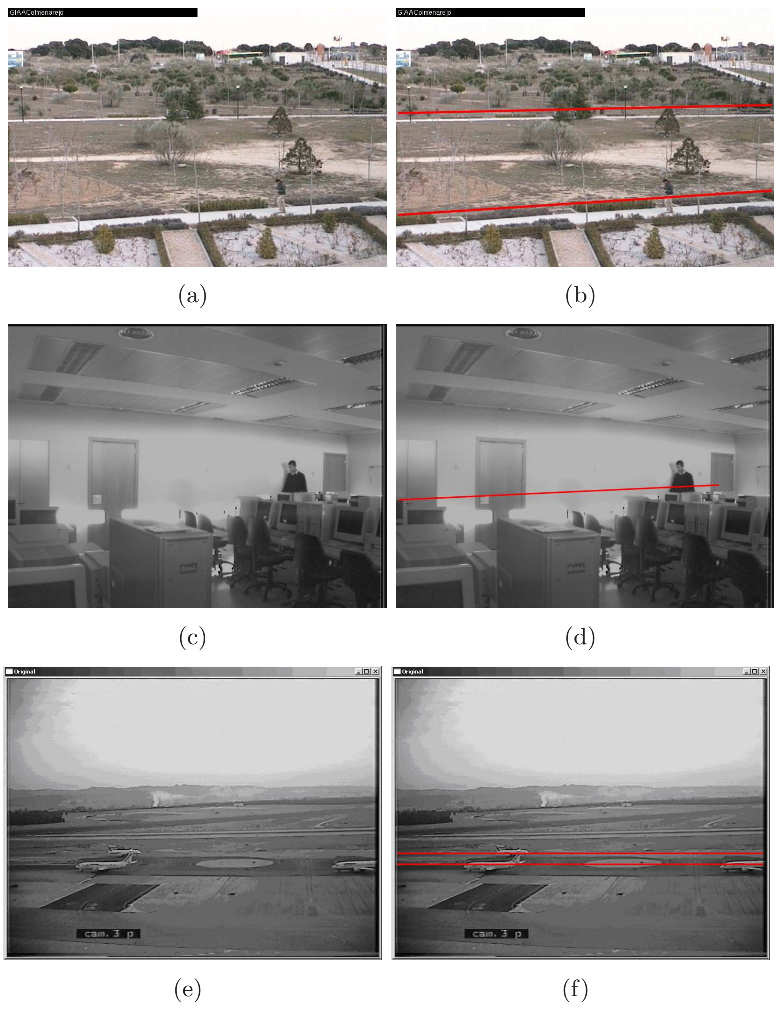


Figure 4.1: Several real video sequence where the targets follow regular paths most of the time

There are two main methods for representing this regular motion patterns:

- A priori definition: In this first case, the evaluation for a tracking algorithm contains a previous definition of the most probable trajectory (ideal trajectory) and compares the result track with it.

- Knowledge extraction from recorded data: An off-line computation computes the "average" trajectory from a database of real trajectories. Then, this it is taken as ideal trajectory for evaluation.

In this work, we follow the first method with paths predefined a priori. The scenes considered for evaluation are trajectories performed by a very regular type of mobile; typically trajectories performed by individuals on a certain path, i.e., university students arriving to the buildings entrances following a footpath. The context is useful since, for example, vehicles move on roads, following the axis direction, and in this particular work, individuals tend to move on sidewalks or specific footpaths.

Defining the context is much easier than marking all ground truth elements; the paths followed by objects (constraints on surface) can be approximated by means of straight segments. The other aspect taken into account to define the evaluation metrics will be the organization of the video samples. That is, the coherence in the selection of the video sequences to be evaluated.

Finally, the evaluation technique aims to be an objective comparison among different trackers in order to check the behavior of each of them depending on the scenario. In particular, the evaluation will compare the performance of the proposed Rules tracker that will be introduced in sections 5.2.1 and 5.2.2, and a bunch of known classifiers available in the literature.

## 4.2 Basis of Evaluation

The first question that arises is which metrics are relevant or not for the evaluation of trackers performance.

The follow up strategy was to use firstly a ground truth technique in order to prove the efficiency and relevance of each metric for the evaluation purposes. Subsequently, the metrics for the evaluation with minimum ground truth will be a natural extension from the well proved ground-truth metrics.

This work proposes, then, a ground-truth metric from which the minimum ground-truth technique will be derived. Both of these evaluation techniques will be used for carrying out experiments.

Therefore, in each scenario, the ground truth was extracted frame by frame, selecting the targets and storing the next data for each target. The targets ($T_i(n)$), individuals or moving vehicles, are represented by a bounding rectangle that surrounds the connected blobs o moving pixels. A rectangle is a suitable approach to characterize the targets in our experiments, individuals and moving vehicles (airplanes and cars), since they shape resemble a rectangle. Moreover, many papers in the literature represents also these targets by means of a rectangle such as [7] and [186].

- Number of analyzed frame: $n$ where $n = 1..N$
- Track identifier: $i$.

- Value of the minimum x coordinates of the rectangle that surrounds the target: $x_{T_i}^{min}(n)$.
- Value of the maximum x coordinates of the rectangle that surrounds the target: $x_{T_i}^{max}(n)$.
- Value of the minimum y coordinates of the rectangle that surrounds the target: $y_{T_i}^{min}(n)$.
- Value of the maximum y coordinates of the rectangle that surrounds the target: $y_{T_i}^{max}(n)$.

These ground truth was compared to the real detections by the evaluation system. First of all, the result tracks were checked to see if they match with the ground truth tracks registered in the ground truth table.

Figure 4.2 depicts the different stages of the Evaluation System in order to have a clear idea of it.

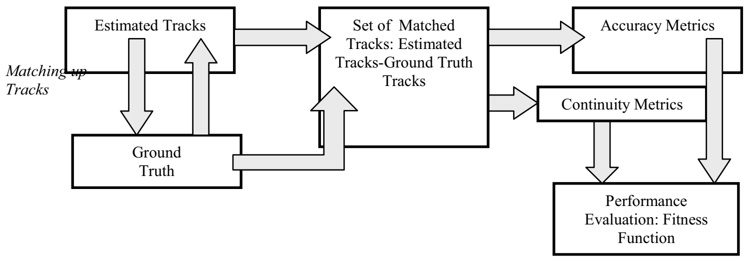

Estimated Tracks
Matching up Tracks
Ground Truth
Set of Matched Tracks: Estimated Tracks-Ground Truth Tracks
Accuracy Metrics
Continuity Metrics
Performance Evaluation: Fitness Function

Figure 4.2: Evaluation System.

If the test is passed, the evaluation system computed four parameters **per target** which were classified into 'accuracy metrics' and 'continuity metrics':

**Accurary Metrics:**

- Overlap-area (OAP): Overlap Area Percentage between the real and the detected blobs. This metric was selected in order to show how precise our tracker system was in the detection, association and track from blobs to tracks. If these processes are good enough, the overlapped area will be high, otherwise the overlapped area will be small.

$$OAP_{T_i} = Area(T_i \bigcap T_{i_{GT}}) \tag{4.1}$$

where $T_{i_{GT}}$ is the rectangle defined in the ground truth for the target $T_i$

- X-error ($E_x$) and Y-error ($E_y$): Difference in x and y coordinates between the centers of the ideal blob and the detected blobs. Again, these two metrics reflect the precision of the tracker for detection, matching and tracking the targets. If the processes are carried out efficiently, the differences will be small.

$$E_{x_{T_i}}(n) = x^g_{T_i}(n) - x^g_{T_{i_{GT}}}(n) \tag{4.2}$$

$$E_{y_{T_i}}(n) = y^g_{T_i}(n) - y^g_{T_{i_{GT}}}(n) \tag{4.3}$$

where $(x^g_{T_i}(n), y^g_{T_i}(n))$ is the center of the detected rectangle (target $T_i$) and $(x^g_{T_{i_{GT}}}(n)$, $y^g_{T_{i_{GT}}}(n))$ is the corresponding center of the ground-truth rectangle ($T_{i_{GT}}$).

**Continuity Metrics:**

- Number of Tracks per target (NT): It is checked if more than one detected track is matched with the same ideal track or with no track at all. If the first case happens, the program keeps the detected track which has a bigger overlapped area value, removes the other one and marks the frame with a flag that indicates the number of detected tracks associated to this ideal one.

  This metric shows if the blobs were fragmented and the association process could not merge them in a single track or if a bad association has been carried out.

$$\begin{aligned} if\Big(\ T_i(n) \Rightarrow T_{i_{GT}}(n)\Big) \wedge \Big(\ T_k(n) \Rightarrow T_{i_{GT}}(n)\Big) \wedge \\ ... \wedge \Big(\ T_{k+m}(n) \Rightarrow T_{i_{GT}}(n)\Big) \Rightarrow NT(n) = m+1; \end{aligned} \tag{4.4}$$

  otherwise

$$if\Big(\ T_i(n) \Rightarrow \neg\exists T_{i_{GT}}(n)\Big) \Rightarrow NT(n) = 0; \tag{4.5}$$

  where $i$ and $k, .., k+m$ are different labels for the the same target $T_{i_{GT}}$.

- Commutation (C): A commutation occurs when the identifier of a track matched to an ideal track changes. It typically takes place when the track is lost and recovered later. This metric reflects a failure of the tracker: the track is lost and then recovered after certain frames. In addition, it might be a temporal lost of the track as a consequence of occlusions.

$$if\Big(\ T_i(n) \Rightarrow T_{i_{GT}}(n)\Big) \wedge \Big(\ T_k(n+m) \Rightarrow T_{i_{GT}}(n+m)\Big) \Rightarrow C(n+m) = 1; \tag{4.6}$$

Besides these parameters, an evaluation function was defined, with the objective of extracting a number that measures the quality level of the tracking system. This number was based on the evaluation metrics specified before. Thus, the resultant number was obtained by means of a weighted sum of different terms which are computed target by target:

- The next three elements are three counters:
  - Overmatch-counter ($O_c$): how many times the ground truth track is matched with more than one tracked object data ($NT > 1$). Again, this metric reflects the fragmentation of a track into several blobs in the detector stage and the failure of the association process to merge this blobs into a single track. Whereas $NT$ is the exact number of detected tracks associated with real tracks, $O_c$ is just a count that increases in one unit anytime an over association occurs.

$$\begin{aligned} O_c &= 0; \\ for\ all\ n\ frames\ where\ n &= 1..N \\ if(\ T_i(n) \Rightarrow T_{i_{GT}}(n)) \wedge (\ T_k(n) \Rightarrow T_{i_{GT}}(n)) \Rightarrow O_c &= O_c + 1; \end{aligned} \tag{4.7}$$

  - Undermatch-counter ($U_c$): how many times the ground truth track is not matched with any track at all ($NT = 0$). This metric shows for all the video how many time the detector has failed by not detecting anything or if the tracker has carried out a very imprecise tracking of a target.

$$\begin{aligned} U_c &= 0; \\ for\ all\ n\ where\ n &= 1..N \\ if(\ T_i(n) \Rightarrow \neg\exists T_{i_{GT}}(n)) \Rightarrow U_c &= U_c + 1; \end{aligned} \tag{4.8}$$

  - Mismatch (M): A counter which stores how many times the ground truth and the tracked object data do not match up (overmatch and undermatch). This metric shows any failure produced by the tracker due to a bad detection, an incorrect association process or a non precise tracking of the target.

$$\begin{aligned} M &= 0; \\ for\ all\ n\ frames\ where\ n &= 1..N \\ if\Big(\ T_i(n) \Rightarrow \neg\exists T_{i_{GT}}(n)\Big) &\vee \qquad (4.9) \\ \vee\Big((\ T_i(n) \Rightarrow T_{i_{GT}}(n)) \wedge (\ T_k(n) \Rightarrow T_{i_{GT}}(n))\Big) \Rightarrow M &= M + 1; \\ end\ for \qquad (4.10) \end{aligned}$$

- The three next terms are the total sum of the non-overlapped areas ($\sum(1 - OAP)$) and the central errors of x ($\sum E_x$) and y axes ($\sum E_y$).

$$\sum_{n=1}^{N} \Big(1 - OAP_{T_i}(n)\Big) \tag{4.11}$$

$$\sum_{n=1}^{N} E_{xT_i}(n) \tag{4.12}$$

$$\sum_{n=1}^{N} E_{yT_i}(n) \tag{4.13}$$

- Finally, the last term is the number of commutations in the track under study ($\sum C$).
- The continuity elements are normalized by the time length of track, $T$, whereas the accuracy terms are normalized by the time length of track being continuous, $CT$ (i.e. when they can be computed).
- $\omega_{1,2,3,4,5,6,7}$ are the relative weights for the terms (detailed later). Highest values have been given to the continuity terms, since this aspect is the key to guarantee the global viability.

Thus, the evaluation per target function can be represented as follows:

$$\frac{\omega_1 M}{T} + \frac{\omega_2 \sum(1 - OAP) + \omega_3 \sum E_x + \omega_4 \sum E_y}{CT} + \frac{\omega_5 O_c + \omega_6 U_c + \omega_7 \sum C}{T} \tag{4.14}$$

The adjustment of the $\omega$ parameters is the next task to carry out. The author uses a "trial-and-error" process in which more importance is given to the failures considered more important for the good performance of the tracking system. For example, if the overlapped area value is a bit higher or lower is less important than if a track is divided in two and associated to the same ground truth or if on the contrary, the track is not detected at all.

In Trial-and-error method, one selects a possible set of parameters, applies them to the problem and if the performance is not satisfactory, selects another set of parameters. The process continues until a proper solution is found, that is, the evaluation function gives a proper value of performance.

This evaluation metric proved to be efficient for measuring the performance of tracking video systems as it will be showed in the experiments carried out in an airport environment in the section 5.3. Nevertheless, the extraction of the ground truth each time we needed a set of training video sequences was a hard task that took long. This was the principal reason why the author decided to propose a new evaluation method that could provide a minimum ground truth in a fast way.

## 4.3 Proposal of Evaluation Technique with Minimum Ground Truth

The proposed evaluation methodology extends the metrics derived from the ground truth study which proved to be effective (see section 5.3). In order to extend these metrics, some requirements must be accomplished by the new technique:

1. Equal number of targets ($O$): In the detailed ground truth technique, the metrics could evaluate the correct or incorrect association between detected and real tracks thanks to the ground truth. Nevertheless, the ground truth in the proposed evaluation technique consists in an *a priori* approximation of a path followed by a target. Then, in order to prove the correctness in the detection and tracking of targets, the training set of videos must have the same number of objects. Therefore, if the training videos have selected to have 3 targets and the final output is a different number, the tracker will have consequently failed.

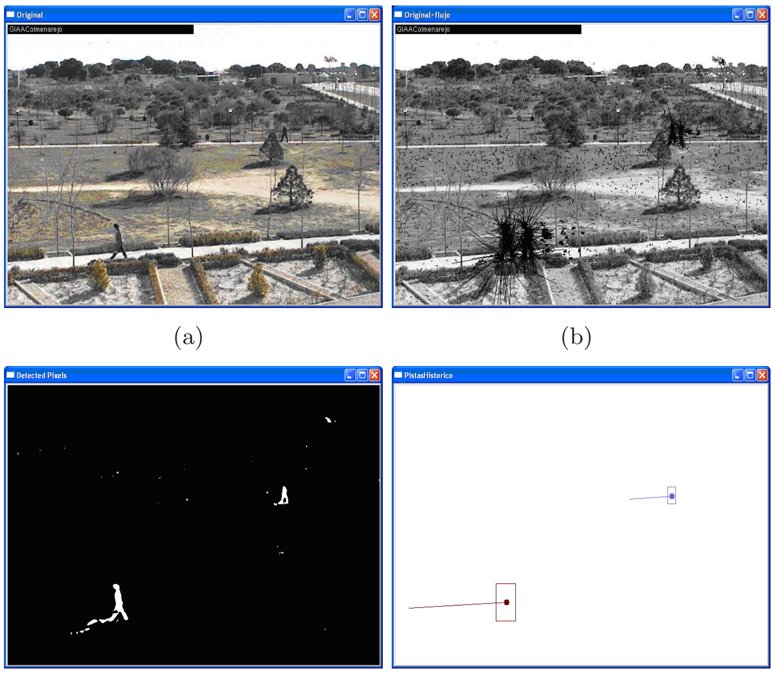
2. Regular and similar trajectories: In the previous ground truth technique, the accuracy in tracking was reflected in the overlapped areas and the $x$ and $y$ differences between the mass centers of the detected and real track. Thus, a new distance to the ground truth must be defined. As a result of this, the videos in the training set must be selected in a way that the targets follow regular paths so that the definition of ground truth is an easy task to perform. Although, as it was pointed before, the tracker must work under all kind of situations.

3. Distance to the focus: In addition, a pre requirement to have a regular path must be to have a useful trajectory. That is, if the target is too close to the camera focus, the trajectory defined could not be long enough to provide good results for evaluation. Thus, an extra requirement would be to have a target whose maximum area is $\frac{1}{5}$ with respect to the whole image.

4. No changes of direction: Finally, the videos selected for evaluation must have targets that move in the same sense of direction in all frames. The reason why this criterion is required lies on the fact that if a change of direction happens in the detailed ground truth technique, the good or bad reaction of the tracker can be evaluated directly by the comparison with ground truth by means of the Overlapped Area metric ($OAP$), the Mismatching ($M$), etc. Nevertheless, in the present case of a minimum ground truth there is not time information in the ground truh. Thus, if this criterion is accomplished, a change of direction in the prediction of the tracker will be consider as a failure.

Within this context, an extensive recording of typical scenes can be carried out, which will be automatically evaluated by the proposed method. It will allow the computation of statistical data to assess robustness, such as tracking continuity or presence of false tracks, and accuracy metrics such as transversal error, area and orientation variations. Therefore, this proposal is oriented toward an "auto-evaluation" point of view, as an alternative to other approaches based on comparisons with detailed and particular ground truth established frame by frame.

To sum up, this section presents our proposal for a tracking performance evaluation function. Thus, the method proposed here is an alternative to other methods based on extracted detailed ground-truth values from real images, with a detailed description of the contours for interesting targets frame by frame. It only uses a set of selected videos to measure the quality of the tracking system and a representation of pathways on surface. These videos are selected considering only situations where the objects follow predefined paths in their movements. It is easy to find this kind of sequences, i.e. the buses and cars in a road or individuals walking in the footpath. Figure 4.3 shows an example of how an individual walks following a straight line. Thus, the quality measure will be computed by only having as an input data the video sequences and the recorded lines followed by the selected targets.



(a) (b) (c) (d)

Figure 4.3: Information levels in the processing chain: Raw Image (a), Optical Flow (b), Detected Pixels (c) and Result Tracks (d)

### 4.3.1 Metrics of the Proposed Evaluation Technique

The general process carried out to evaluate a video tracking system is sketched in the figure 4.4. A collection of videos for targets moving in similar conditions is processed so that the tracking output can be accumulated across different videos and generate an averaged description of system performance.

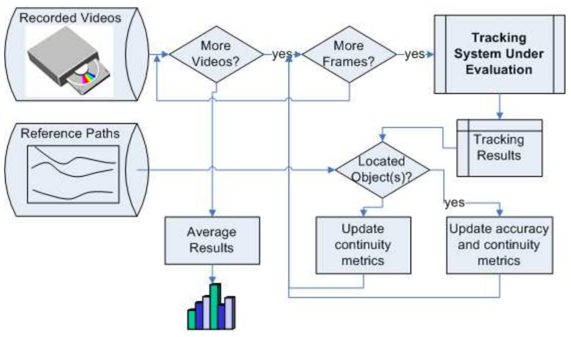


Figure 4.4: Cumulative evaluation methodology

Contextual information about the set of videos is that targets always follow certain paths on surface. These paths will be represented by an approximation of straight segments $f(x)$(see figures 4.5 and 4.6).

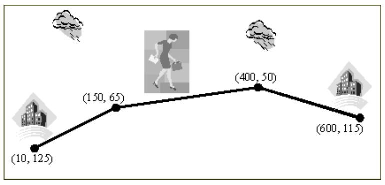


Figure 4.5: Approximation of a trajectory by means of straight segments

It is clear that the simplicity and easiness of obtaining these data contrast with other common ways of obtaining the ground truth (i.e. a human operator who must mark every contour of the targets in the screen). The evaluation system calculates the distance between the target and the set of segments. Then, the system decides by taking the shortest distance which segment is selected as the approximation of the trajectory, the ground truth. Figure 4.8 illustrates the comparison process to evaluate the accuracy and continuity metrics, detailed next in the following paragraphs.

Then, the basic of this method is the definition of the function $f(x)$, that is, the approximation by means of straight segments of the trajectory followed by a target.

In order to have a more detailed idea of the system performance, the area under study is divided into different zones. Each zone is defined as a fixed number of pixels of the x-axis, the 10% of the horizontal size of the image (see figure 4.7). The horizontal component has

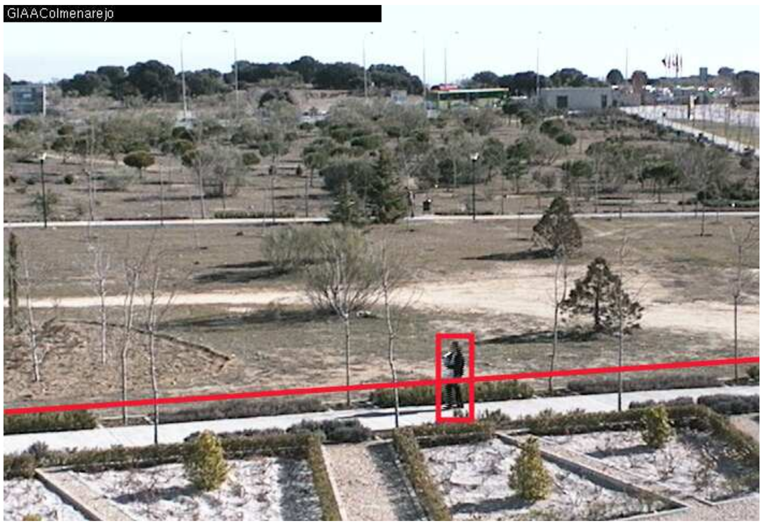


Figure 4.6: Straight line followed by an individual in a footpath

been selected in this case because it constitutes the main coordinate in which the objects move in this particular study, but they could be defined along the trajectory. As a result of this, we have a more intensive appreciation of the performance for different situations (object initialization, manoeuvres, etc.), and where they change, keep stable or have a big slope.

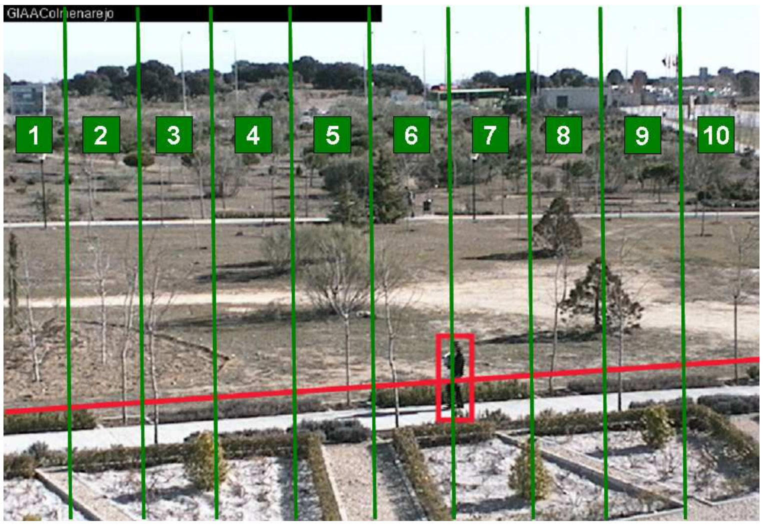


Figure 4.7: Image divided into 10 zones for the study of the metrics in each area

Each time a track is initiated or updated by the tracking system, the evaluation system selects the relevant tracks to evaluate. The straight segment considered as the ground truth in this moment is compared with the tracks to decide which of them will update

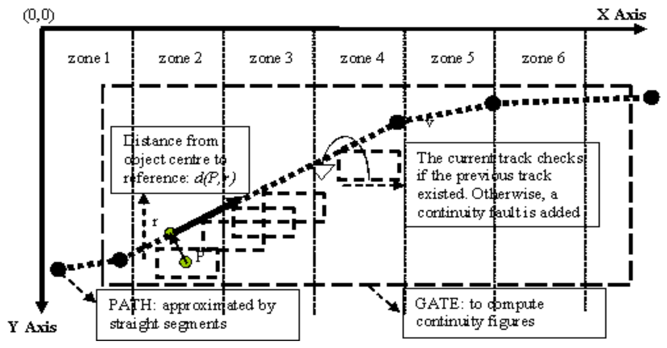

(0,0)
X Axis
Y Axis
zone 1
zone 2
zone 3
zone 4
zone 5
zone 6
Distance from object centre to reference: d(P,r)
r
P
The current track checks if the previous track existed. Otherwise, a continuity fault is added
PATH: approximated by straight segments
GATE: to compute continuity figures

Figure 4.8: Continuity and accuracy evaluation

the metrics, within a certain margin around. The evaluation metrics provide a qualitative and quantitative method for evaluating the performance of the Tracking System.

These metrics have been divided into two groups:

- The accuracy metrics assess how precise is our prediction in contrast to the expected or ideal behavior. They can give also a comparative measure between the current and the previous track to evaluate the stability of estimated vectors. The evaluation system calculates the absolute values and moreover the mean, variance, maximum and minimum values.
- The robustness metrics assess the rate of discontinuous behaviors occurring during the tracking process. These results are referred to the spatial zones indicated above The next subsections describe all the metrics in detail.

**Accuracy Metrics**

A second subdivision is done according to the previous knowledge that is needed. If the metric requires information of the previous update of the track under study, the metrics are referred as relative accuracy metrics. Otherwise, they are called absolute accuracy metrics.

- Absolute Accuracy Metrics
  - Absolute Area ($A$): It is computed by calculating the area of the detected track. It is useful to check the variability of the track area along the time. Under normal conditions (an individual walking in a normal way), the area must be almost constant. Moreover, it gives and idea about the target dimensions. If the area value is too high, the track could have considered shadows or reflexions as part of the track. This metric is an adaptation of the overlapped area (OAP)

explained in the detailed ground truth evaluation technique. In that case, it was immediate to check the quality of the track by directly comparing both areas, the detected and the real one. In the current case, the area is stored for each frame and then compared at the end of the video sequence.

$$A(n) = Area(T_i(n)) \tag{4.15}$$

where $n$ is the frame number $n = 1...N$.

– Transversal Error ($TE$): It is defined as the projected distance between mass center of the target and reference path (Figure 4.8). This metric is the corresponding to the distance between detected and real mass centers in the detailed ground truth technique (X-error ($E_x$) and Y-error ($E_y$)). In this case, the real mass center is the closest point to the real trajectory defined by the corresponding ground truth straight segment. In the current case, the minimum ground truth does not provide temporal information. For that reason, this metric only measures the transversal component of the error.

$$TE(n) = d\Bigg( (x^g_{T_i}(n), y^g_{T_i}(n)), (x_{T_{MGT}}(n), y_{T_{MGT}}(n)) \Bigg) \tag{4.16}$$

where $MGT$ stands for minimum ground truth or the reference path.

– Orientation Error ($OE$): deviation between the velocity of the output tracks and reference uniform velocity vector of the straight segment. In this case, the equivalence with any metric in the detailed ground truth case is not direct. Nevertheless, this metric tries to reflect the correct tracking: detection, association and estimation of the targets' tracks.

$$OE(n) = arctg\left(\frac{v^g_{y_{T_i}}(n)}{v^g_{x_{T_i}}(n)}\right) - arctg\left(\frac{\bar{u}_{y_{T_{GT}}}(n)}{\bar{u}_{x_{T_{GT}}}(n)}\right) \tag{4.17}$$

- Relative Accuracy Metrics

  – Inter-frame Area Variation (ΔA): Variation of area between the current and the previous update of the track under study. As it was said before, the area of the target under normal conditions should not be very different. This variation per area reflect this behavior.

$$\Delta A(n) = Area\Bigg(T_i(n)\Bigg) - Area\Bigg(T_i(n-1)\Bigg) \tag{4.18}$$

  – Inter-frame Orientation Variation (ΔOE): Variation of the direction vector between the current track and the previous update of the track under study.

$$\Delta OE(n) = arctg\left(\frac{v^g_{y_{T_i}}(n-1)}{v^g_{x_{T_i}}(n-1)}\right) - arctg\left(\frac{v^g_{y_{T_i}}(n)}{v^g_{x_{T_i}}(n)}\right) \tag{4.19}$$

**Robustness Metrics**

- Continuity Faults ($C$): This metric is only measured inside a predefined gate (Figure 4.8), which represents the area where no new tracks can appear or disappear. Each time a track is not labeled as the previous one, a continuity fault is added. Thus, this metric reflects the failures of the trackers due to the lost of the track or a mismatching in the association process.

  This metric is a direct adaptation from the Commutation ($C$) metrics computed in the detailed ground truth technique.

$$\begin{aligned} if\ T_i \subset GATE \\ C(n) = 0 \\ if\ \Big(T_i(n-1) \Rightarrow label[T_i(n-1)]\Big) \wedge \Big(\ T_i(n) \Rightarrow label[T_i(n)]\Big) \wedge \\ \wedge\Big(label[T_i(n-1)] \neq label[T_i(n)]\Big) \Rightarrow C(n) = 1; \end{aligned} \tag{4.20}$$

- Changes of direction ($D$): This metric marks when a track changes its direction through the scalar product of track displacement. This metric is also a counter where one unit is added each time a change of direction occurs.

$$\begin{aligned} D(n) = 0 \\ if\ ((x^g_{T_i}(n-2) - x^g_{T_i}(n-1))(x^g_{T_i}(n-1) - x^g_{T_i}(n)) + \\ (y^g_{T_i}(n-2) - y^g_{T_i}(n-1))(y^g_{T_i}(n-1) - y^g_{T_i}(n)) < 0) \\ \Rightarrow D(n) = 1; \end{aligned} \tag{4.21}$$

- Number of Detected Objects (O): It provides knowledge about the number of detected objects resulting from tracking. The videos are selected with a single object per one, but the system may fail and detect more objects (false positives) or fail and detect none (false negatives). Thus, each time a track is initiated, the tracking system marks it with a unique identifier in the area under study. After the evaluation of the whole set of videos, this metric is normalized by the total number of videos in the set (the output under ideal conditions would be one detected object per zone and video). This metric is an adaptation to the correct or incorrect association given by the Mismatch $M$ metric and the Overmatch/Undermatch counters ($O_c/U_c$) of the detailed GT technique.

$$\forall n\ where\ n = 1..N\ O = \sum_{i=1}^{M} T_i \tag{4.22}$$

  These two metrics are normalized by the number of videos under study.

Thus, the evaluation function is based on the previous metrics, by means of a weighted sum of different terms which are computed for each target:

$$e_{ij} = f(\bar{x}_{ij}, \theta) = \frac{\omega_1 \cdot \sum_n \Delta A(n) + \omega_2 \cdot \sum_n TE(n) + \omega_3 \cdot \sum_n \Delta OE(n)}{CT} + \frac{\omega_4 \cdot \sum_n C(n) + \omega_5 \cdot \sum_n D(n) + \omega_6 \sum_n O(n)}{T} \tag{4.23}$$

where $e_{ij}$ is the evaluation of target $i$ in the video $j$, with the terms T, CT and $\omega_{1,2,3,4,5,6}$ define as follows:

- The continuity elements are normalized by the time length of track, T, while the accuracy terms are normalized by the time length of track being continuous, CT (i.e. when they can be computed).
- $\omega_{1,2,3,4,5,6}$ are the relative weights for the terms (detailed later). Highest values have been given to the continuity terms, since this aspect is the key to guarantee the global viability. The adjustment of these parameters is carried out by the "trial-and-error" method as we saw before for the weights belonged to the evaluation with full ground-truth (section 4.2).

The current evaluation function with minimum ground-truth follows the same structure than the evaluation with ground-truth (section 4.2), that is, the sum of metrics weighted by certain coefficients and divided by some parameter related to time.

## 4.4 Experiments: People Walking Outdoors

This experiment is carried out in order to prove the effectiveness of the proposed evaluation technique. In particular, this experiment aims to check the sensibility of the evaluation metrics when the detection gets worse as a consequence of an incorrect adjustment of the detection threshold. On the other hand, the effectiveness of the proposed evaluation technique for the optimization process will be showed in the next chapter (section 5.4)

The experiments were programmed using the well known C++ [187] language program developed by Bjarne Stroustrup. In order to deal with the image processing, this work uses the Open Computer Vision Library (OpenCV) [188] developed by Intel. Finally, all the figures used to compare the metrics are generated utilizing the MATLAB environment [189].

The author presents two parallel detection and tracking processes for evaluation of certain videos. The first detection and tracking process present an inappropriate value of the detector threshold (a lower value of threshold = 20) which makes the system detect and process some blobs that should have been considered as noise. On the other hand, the second group of detection and tracking processes have a more adequate value of threshold which makes the system work properly (threshold = 25). The author aims to show the reflection of the worse performance in the first set of processes whereas performance improves for the second set. That will prove the goal of the evaluation technique to correctly evaluate a set of videos given a set of parameters.

Then, the results were obtained for two types of situations, recording 50 video samples for each one: people walking from left to right and people walking from right to left.

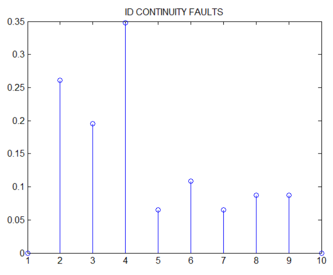
The known context for all samples is that a single interesting target is moving in the marked area (footpath) so that a single object detection and tracking should be obtained per video. Besides, the object performs a smooth movement, without jumps or sudden changes of direction, so a continuous track should exist from the object appearance until its disappearance. Global results are presented in tables, while detailed figures, depending on the horizontal position, are presented for specific situations, bus arrivals at stop and people moving from left to right.

The evaluation has been done with all available samples (100 videos) to characterize the system performance and obtain an objective comparison between very close values. This evaluation provides a detailed evaluation of a given system (how close it is to the ideal situation of maximum accuracy and robustness) and besides, it allows making comparisons supported in a significant number of samples as an alternative to conventional analysis based on a single particular sequence.

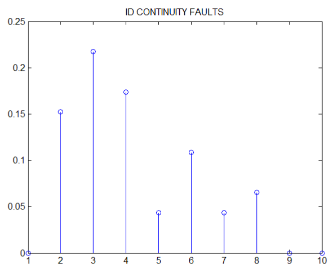
Detailed figures are presented for one of the cases (left-to-right), because some metrics can be explained with this contextual separation (all objects are initialized at the left-hand side, so the continuity and accuracy metrics are worse at this side). At the end, tables will summarize the accuracy for all available samples.

In the first place, the robustness results appear in figure 4.9. As we can see, the threshold 25 achieves a slightly more stable performance. It reduces the number of continuity faults and changes of direction, and achieves detection rates closer to 1. It can be seen, from this figure, that the robustness is worse during initialization, in this case at the left-hand side. The robustness figures can be compared with the case of a very low threshold value, 4, presented in figure 4.10, where the continuity is much worse.

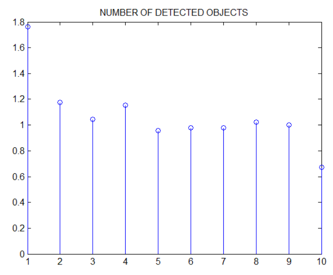
Regarding the accuracy, figures 4.11 and 4.12 present the absolute accuracy metrics (estimated area, transversal and orientation errors). The histograms have the global distribution of data (all samples), and the 2D figures present their spatial distribution. For all metrics, the average value is presented first with a solid line. A first band around mean, with dashed lines, presents the 1-$\sigma$ interval, while the total variation (from minimum to maximum) is depicted with a second, dotted-line band. In these samples, tracking people had the problem of size distortion due to shadows, producing larger regions than real



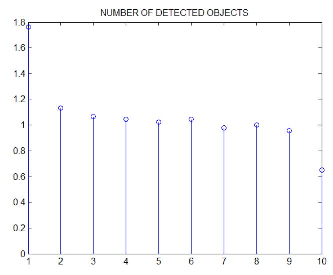
(a)



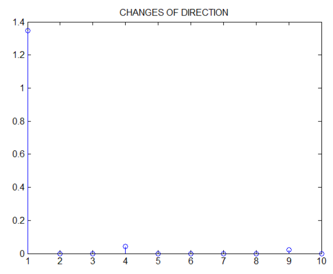
(b)



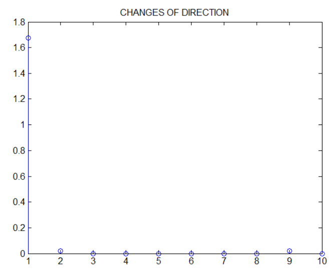
(c)



(d)



(e)

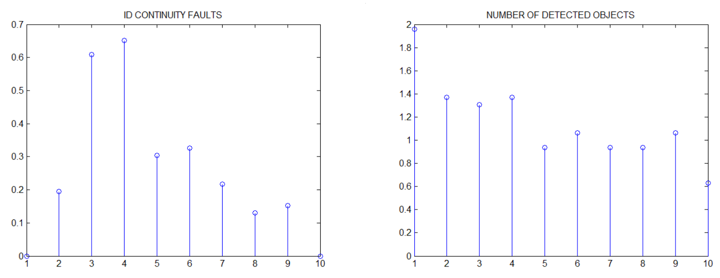


(f)

Figure 4.9: Continuity figures with 50 videos of people walking (left to right)

objects. The variance has a maximum at a position at the left-hand side, after the initialization. The orientation error has its maximum during initialization, and it is stabilized later, a typical behavior of Kalman-based tracking filters. As we can see, the 25-value threshold allowed slightly more accurate results than 20-value threshold, with more compressed intervals, especially for the case of estimated area and orientation error. So, the absolute area interval is reduced from $6 * 10^4$ to $4 * 10^4$, or the intervals for orientation error are clearly narrower. The summary tables with dispersion parameters will be presented later.

Relative accuracy metrics are presented in figure 4.13, the variation in area and orientation between consecutive frames for a same object. They represent the stability of tracks, and the behavior is similar to the corresponding absolute figures



(a) (b)

Figure 4.10: Continuity figures with 50 videos of people walking (left to right) for threshold=4

Finally, the accuracy results for the two detection thresholds are summarized in the table 4.1. As we can see, both the standard deviation and range intervals are reduced in most metrics with the 25-value threshold.

Table 4.1: Accuracy metrics with 50 videos of people walking (left to right)

| | | Transv. | Area | Orient. | I-F Area | I-F Orient. |
|---|---|---|---|---|---|---|
| Max | Th 20 | 49.7498 | 56688.5 | 89.751 | 38931 | 178.896 |
| | Th 25 | 49.9189 | 37775.7 | 89.716 | 28153.4 | 179.999 |
| Min | Th 20 | 0.0023 | 138.509 | -89.226 | -26362 | -172.76 |
| | Th 25 | 0.0063 | 154.054 | -89.898 | -17905.7 | -179.99 |
| Std | Th 20 | 9.8737 | 5558.28 | 26.359 | 2442.81 | 24.1396 |
| | Th 25 | 9.5468 | 5087.50 | 23.41 | 1895.57 | 25.0163 |

The same values were computed for the total set of people-tracking samples, presented in table 4.2. This illustrates the consistency between evaluations with different sets of data.

From the evaluations presented above, the following conclusions can be derived for this system:

- A slight increase in the detection threshold, from 20 to 25, allows a significant improvement, both in system robustness and accuracy.
- The robustness is not completely satisfactory, with a certain number of discontinuous objects detected, and a rate of detected objects separated from ideal values (one when there is a single object, zero otherwise). The main source of problems is the initialization of new objects
- The accuracy figures show a problem in size estimation with people tracking, due to the presence of shadows, not considered in the tracking algorithms.

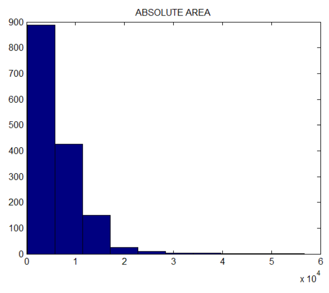
Table 4.2: Accuracy metrics with 100 videos of people walking (both directions)

| | | Transv. | Area | Orient. | I-F Area | I-F Orient. |
|---|---|---|---|---|---|---|
| Max | Th 20 | 49.7498 | 56688.5 | 89.751 | 52342.5 | 178.8969 |
| | Th 25 | 49.9189 | 37775.7 | 89.8765 | 28153.4 | 179.9998 |
| Min | Th 20 | 0.0023 | 111 | -89.226 | -26362 | -178.932 |
| | Th 25 | 0.0045933 | 154.054 | -89.898 | -17905.7 | -179.999 |
| Std | Th 20 | 9.6918 | 6263.40 | 24.075 | 2543.00 | 21.742 |
| | Th 25 | 9.3282 | 5107.7266 | 22.898 | 1929.70 | 22.0466 |

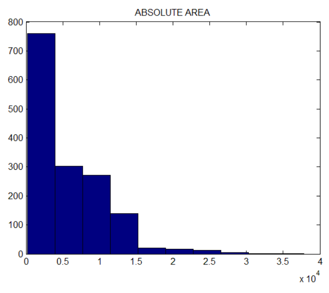
These results have been obtained to illustrate the evaluation capability only for a particular case of system and its dependence on a certain parameter, the threshold value of detector. The system could be applied to any other analysis or comparison, depending on the application of evaluation.

## 4.5 Conclusions

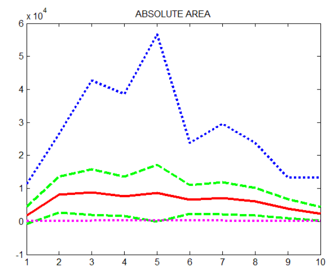
This chapter has presented an evaluation methodology based on several performance metrics for video-based tracking systems, in which the references needed are obtained at a low cost, compared with conventional approaches. Tracked objects move on surface following certain paths, and the sequences are organized into groups according to the objects behavior in different situations. The experiments show that the evaluation system reflects in an objective and quantitative way the proper or inadequate performance of a tracking system. An improper parameterization, leading to a bad performance clearly perceived by a human observer, is directly reflected in the evaluation metrics. The results offer a precise and detailed analysis of the performance, since a spatial division of sequences allows covering different situations of the tracking process, such as tracks initialisation or manoeuvres. The methodology is able to process any number of video samples so that evaluation data can be accumulated to carry out a statistical analysis (histograms, dispersion figures, etc.), and provide an accurate characterization of the tracking system. As an example, two very similar systems, with a slight difference in the thresholding levels, were compared to illustrate the process.



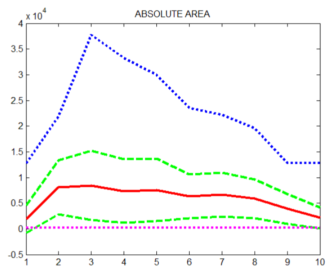
(a)



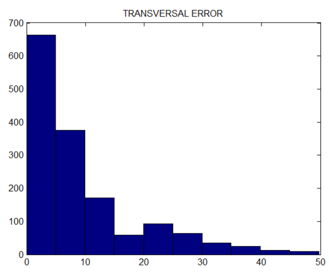
(b)



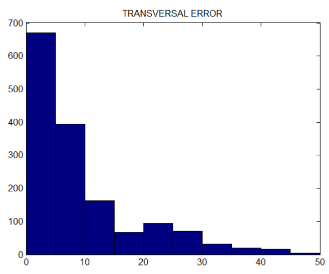
(c)



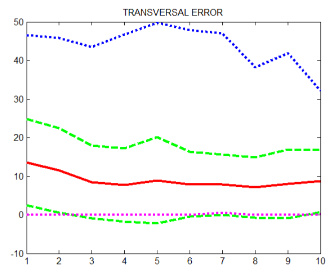
(d)



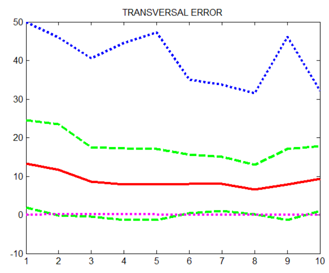
(e)



(f)

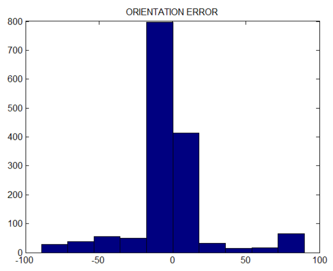


(g)

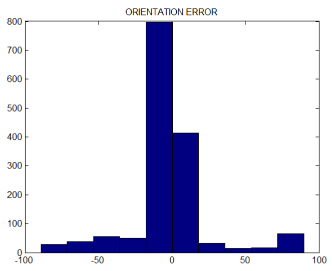


(h)

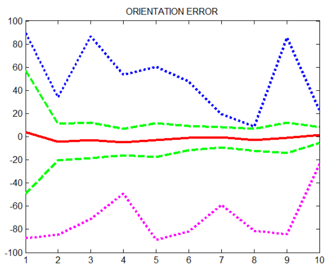
Figure 4.11: Absolute accuracy figures with 50 videos of people walking (left to right)



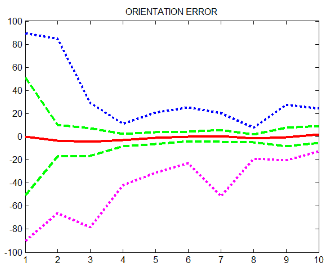
(a)



(b)

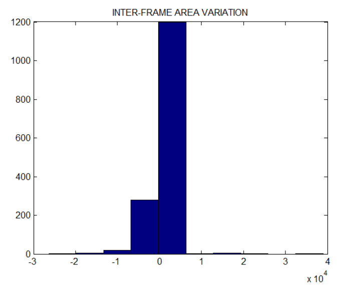


(c)

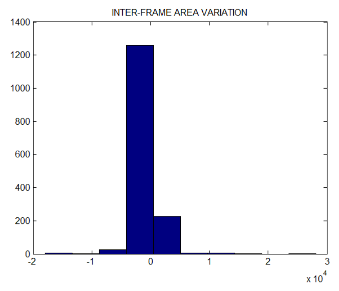


(d)

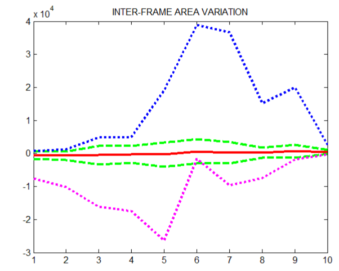
Figure 4.12: Orientation error figures with 50 videos of people walking (left to right)



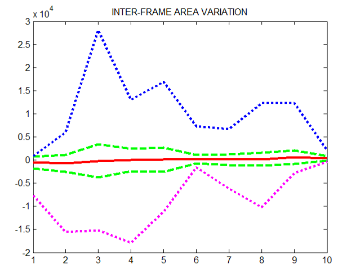
(a)



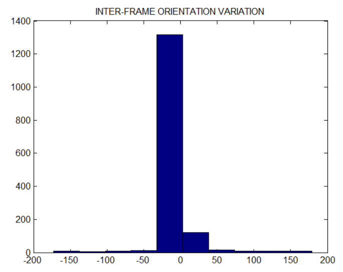
(b)



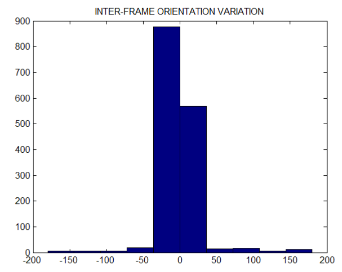
(c)



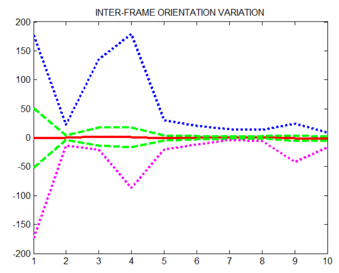
(d)



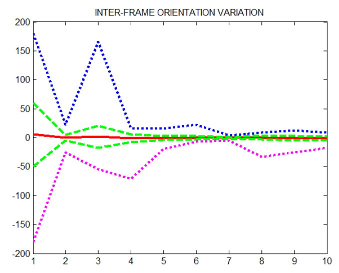
(e)



(f)



(g)



(h)

Figure 4.13: Continuity Faults (C): a) Th=20, b) Th=25

# Chapter 5

# Optimization and Generalization by Means of Evolutionary Strategies

## 5.1 Motivation

A video surveillance system depends on several inter-connected processes, and each one can be regulated by several *parameters*. The main problem of adjusting this system is that all processes must work at the same time, since their inter-relation affects severely, with non-linear dependence, to the final performance. Therefore, the adaptation of the final system performance for a specific implementation is a complex task of design. Furthermore, the set of scenarios used to design and train the system should produce a general solution of the tracking system. A small set of scenarios can lead to learning techniques that adapt exactly to these specific examples (over-fitting), with the consequent loss of generality. On the contrary, random selected examples could produce disorientation in the search. Thus, the set of data that optimize the search of the suitable parameters is defined as the *ideal trainer* [153]. In this problem, each trajectory or moving target could be considered as a sample, and the ideal trainer is a set of trajectories that represent different situations of the surveillance problem that should be reflected in the final design.

Moreover, the processing time for having a real time system is a crucial constraint in this work. In order to make this computation faster, the complexity of the algorithms employed in the tracking system make a remarkable difference in the resultant time.

In this particular case, the author is interested in adjusting the parameters of the whole detector and tracking system in order to obtain three interrelated objectives: (1) the best performance of the system, and (2) the set of parameters that make the system work properly in different conditions and scenarios and (3) the best performance by using a simple tracker based on rules (”Rules Tracker”) which computational weight is very low compared with other tracker algorithms (Mean Shift Tracking [15], Particle Filter based on MS weight [16], etcetera). The second objective must be limited in the sense that different scenarios mean distinct scenes in the same environment. That is, if the tracking system is adjusted to track people outdoors, the scenarios must belong to the same views of the same environment, but it would not be valid to put the system to track aircraft in an airport domain. The author proposes to adjust the parameters of the system in such a robust way that it works well regardless of where the cameras are deployed, the lighting conditions, etc.

To sum up, one of the purposes of this thesis is the search of the most general set of parameters to configure a whole video surveillance system and achieve an optimal performance under representative and general situations.

## 5.2 Proposal: Technique to Optimize and Generalize

To achieve our goal, this thesis follows several steps:

First of all, a set of evaluation metrics **per track** have been proposed to assess the input parameters, considering complementary terms (accuracy and continuity) in different situations to obtain the final design. Next, we take a representative set of scenarios to train the system.

The adjustment of the parameters is based on an optimization carried out by means of Evolutionary Strategies (ES). As we have said before, Evolutionary Strategies (ES) are chosen for this application since the function landscape is very irregular, with plenty of local optima. The classical techniques of optimization would be poorly suitable to these types of problems, when there is a high number of local optima, or other properties like high roughness and local flat surfaces [128, 129, 150]. Moreover, as the values to optimize will be a mixture of continuous and discrete variables, Evolutionary Strategies are particularly suited for this optimization.

One of the innovations of our proposal is that the fitness function is adjusted in different ways in order to optimize the tracking of distinct moving targets, from a single one in a specific scenario to many of them in different scenarios. Thus, whereas over-fitting means obtaining particular parameters for specific situations, generalization is the capability to adapt the parameters to be used in many different cases. Hence, the fitness function is formed by the combination of the performance of the tracking system over the whole set of trajectories. This combination is carried out with simple operators: SUM and MAX. That performance is assessed by means of the evaluation metrics per track mentioned in the previous chapter.

To sum up, in order to achieve our goals, we need to propose a technique of optimization and generalization that allows us to find the most suitable set of parameters and their values for the surveillance system for different scenarios. That means that the values of the parameters must provide the best results for different locations of the camera or environmental conditions such as illumination or weather conditions. The optimization tool is based on the use of Evolutionary Strategies, whereas the generalization method consists in combining the evaluation function of **each track** in several ways (SUM and MAX) and steps in order to gradually build a general fitness function, adapting the system for a whole group of situations.

Next sections explain the solutions adopted in this thesis for the object detector and tracking system and the parameters that must be adjusted. Then, the generalization technique is presented in detail.

### 5.2.1 Detector and Blobs Extraction

The detection algorithm is based on the detection of targets by contrasting with local background, whose statistics are estimated and updated with the video sequence. The pixel level detector is able to extract moving features from background, comparing the difference with a threshold (THRESHOLD).

$$Detection(x, y) := [Image(x, y) - Background(x, y)] > THRESHOLD \cdot \sigma \tag{5.1}$$

being $\sigma$ the standard deviation of pixel intensity. This parameter determines the first filter on the data amount to be processed in following phases.

Finally, the algorithm for blobs extraction marks with an unique label all detected and connected pixels resulted from the previous process. Previously, this image has been transformed in a binary image in order to set to 0 the background pixels and to 1 the foreground ones. Then, the rectangles which enclose the resulting blobs are built, and their centroids and areas are computed. In order to reduce the number of false detections due to noise, a minimum area, MIN_BLOB_AREA, is required to form blobs. This parameter is a second data filter which avoids noisy detections from the processing chain.

$$Rec(b_i) > MIN\_BLOB\_AREA \tag{5.2}$$

where $Rec()$ is the rectangular enclosed area around the blob $b_i$ where $i = 1..N$.

### 5.2.2 Tracking Algorithm

Broadly, the system architecture used in this thesis is a coupled tracking system where the detected objects are processed to initiate and maintain tracks. These tracks represent the real targets in the scenario and the system estimates their location and kinematic state. The detected pixels are connected to form image regions referred to as blobs. The association process assigns one or several blobs to each track, while not associated blobs are used to initiate tracks [102]. This system is based on easy rules (if-then rules) and this is the reason why we call this tracker "Rules".

In particular, the tracker can be divided in two big processes: the matching between blobs and their corresponding track and the management of tracks (initialization, updating and deletion).

### Blobs-to-track Association

The association problem lies in deciding the most proper grouping of blobs and assigning it to each track for each frame processed. Due to image irregularities, shadows, occlusions, etc., a first problem of imperfect image segmentation appears, resulting in multiple blobs generated for a single target. Blobs must be re-connected before track assignment and updating. However, when multiple targets move closely, their image regions may overlap.

As a result, some targets may appear occluded by other targets or obstacles, and some blobs can be shared by different tracks. For the sake of simplicity, first, a rectangular box has been used to represent the target. The association algorithm that this thesis proposed is based on some simple, intuitive and effective rules as the author explains in the next lines.

To sum up, in the simple case, when the regions of a specific target ($T_a$) do not overlap, this target is formed by the union of $K$ blobs:

$$T_a \Rightarrow b_i \cup ... \cup b_{i+K} \tag{5.3}$$

If the targets ($T_a$ and $T_b$) overlapped, there are some common blobs in common for each target:

$$T_a + T_b \Rightarrow (b_i \cup ... \cup b_{i+K}) \bigcap (b_j \cup ... \cup b_{j+L}) \neq NULL \tag{5.4}$$

Around the predicted position, a rectangular box with the estimated target dimensions is defined, $(xmin, xmax, ymin, ymax)$. Then, an outer gate, computed with a parameter defined as a margin, MARGIN_GATE, is defined. It represents a permissible area in which to search more blobs, allowing some freedom to adapt target size and shape.

The association algorithm analyzes the track-to-blob correspondence. It firsts checks if the blob and the track rectangular gates are compatible (overlap), and marks as conflictive those blobs which are compatible with two or more different tracks. After gating, a

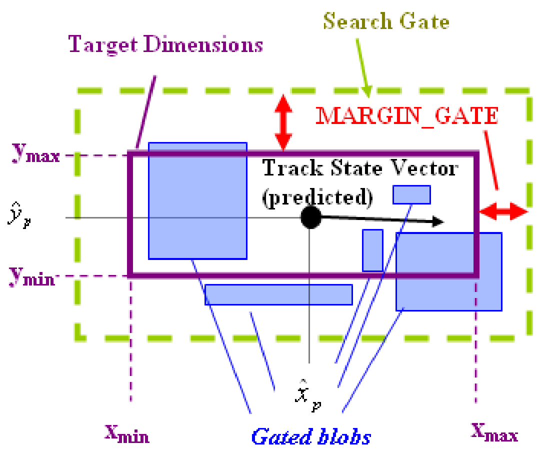


Figure 5.1: Target segmentation with estimated box

grouping algorithm is used to obtain one ”pseudoblob” for each track (see figure 5.1) . This pseudoblob will be used to update track state. If there is only one blob associated to the track and the track is not in conflict, the pseudoblob used to update the local track will be this blob. Otherwise, two cases may occur:

1. A conflict situation arises when there are overlapping regions for several targets (conflicting tracks). In this case, the system may discard those blobs gated by several tracks and extrapolate the affected tracks. However, this policy may be too much restrictive and might degrade tracking accuracy. As a result, it has been left open to design by means of a Boolean parameter named CONFLICT, which determines the extrapolation or not of the tracks.

2. When a track is not in conflict, and it has several blobs associated to it, these blobs will be merged on a pseudoblob whose bounding limits are the outer limits of all associated blobs. If the group of compatible blobs is too big and not dense enough, some blobs (those which are further away from the centroid) are removed from the list until density and size constraints are held. The group density is compared with a threshold, MINIMUM_DENSITY, and the pseudo-blob is split back into the original blobs when it is below the threshold.

The next step is to know how the tracks are initialized, updated and deleted.

## Tracks Management

A recursive filter updates centroid position, rectangle bounds and velocity for each track from the sequence of assigned values, by means of a **Kalman filter** for each Cartesian coordinate, with a piecewise constant white acceleration model [190]. The acceleration variance that will be evaluated, usually named as "plant-noise", is directly related with tracking accuracy. The predicted rectangular gate, with its search area around, is used for gating. It is important that the filter is "locked" to real trajectory. Otherwise, tracks would lose their real blobs and finally drop. Thus, this value must be high enough to allow manoeuvres and projection changes, but not too much, in order to avoid noise. As a result, it is left as an open parameter to be tuned, VARIANCE_ACEL.

Finally, tracking initialization takes blobs that are not associated to any previous track. It requires that non-gated blobs extracted in successive frames accomplish certain properties such as a maximum velocity and similar sizes, which must be higher than a minimum value established by the parameter MINIMUM_TRACK_AREA. In order to avoid multiple splits of targets, established tracks preclude the initialization of potential tracks in the surrounding areas, using a different margin than the one used in the gating search. This value which allows track initialization is named MARGIN_INITIALIZATION.

The equations for the track initialization must accomplish these equations in successive frames (usually $n = 2$ or 3 frames):

$$\bar{x}_{T_a}^{max}(n) = x_{T_a}^{g}(n-1) + MAX\_VELOCITY \cdot t + m \cdot \sqrt{2R_x} \tag{5.5}$$
$$\bar{x}_{T_a}^{min}(n) = x_{T_a}^{g}(n-1) - MAX\_VELOCITY \cdot t - m \cdot \sqrt{2R_x} \tag{5.6}$$
$$\bar{y}_{T_a}^{max}(n) = y_{T_a}^{g}(n-1) + MAX\_VELOCITY \cdot t + m \cdot \sqrt{2R_y} \tag{5.7}$$
$$\bar{y}_{T_a}^{min}(n) = y_{T_a}^{g}(n-1) - MAX\_VELOCITY \cdot t - m \cdot \sqrt{2R_y} \tag{5.8}$$

where $\bar{x}_n^{max,min}$, $\bar{y}_n^{max,min}$ are the predicted maximum coordinates of the current rectangle $n$, $x_{n-1}^g$ and $y_{n-1}^g$ is the gravity center of the non associated rectangle in the previous frame $n-1$, MAX_VELOCITY is the maximum velocity for a pedestrian, $t$ is the time different between frames, $m$ is an integer value (usually $m = 2$), and $R_x$, $R_y$ are the covariance matrix for the $x$ and $y$ coordinates.

Then, if the next conditions are accomplished, the non associated rectangle is matched with a new track if:

$$\begin{cases} x_{T_a}^{g}(n) < \bar{x}_{T_a}^{max}(n) \\ x_{T_a}^{g}(n) > \bar{x}_{T_a}^{min}(n) \\ y_{T_a}^{g}(n) < \bar{y}_{T_a}^{max}(n) \\ y_{T_a}^{g}(n) > \bar{y}_{T_a}^{min}(n) \\ Area(T(x_{T_a}^{max}(n), x_{T_a}^{min}(n), y_{T_a}^{max(n)}, y_{T_a}^{min})) > MIN\_TRACK\_AREA \end{cases} \tag{5.9}$$

where $x_n^g$ and $y_n^g$ is the gravity center of the non associated rectangle in the current frame $n$, T is the track computed by the system which rectangle is formed by $(x_n^{max}, x_n^{min}, y_n^{max}, y_n^{min})$

Then, the adjustable parameters for regulating the object detector and tracking performance are:

1. THRESHOLD
2. MIN_BLOB_AREA
3. MARGIN_GATE
4. CONFLICT
5. MINIMUM_DENSITY
6. VARIANCE_ACEL
7. MINIMUM_TRACK_AREA
8. MARGIN_INITIALIZATION

By properly tunning these parameters, the information passed to the high level process (activity recognition) will be more precise and reliable.

### 5.2.3 Technique for Optimization and Generalization

In this section, a technique is presented to search suitable parameters for the best performance of the surveillance video system under different circumstances, that is, a method to achieve both the best performance of the system by having the ability to work in diverse scenarios.

The core of this technique is to find the fitness function using as a base the evaluation function that assesses each track. Thus, we can use any of the evaluation performance technique suggested in chapter 4. This evaluation $e_{ij}$ (where $i$ is the target and $j$ the scenario or video sequence of the training set) is the initial step to carry out the two following steps. Thus, we are going to assess the performance of the surveillance video system by evaluating each resulting single track. For example, in the case of the evaluation with minimum ground truth:

$$e_{ij} = f(\bar{x}_{ij}, \theta) = \frac{\omega_1 \cdot \sum(\Delta A) + \omega_2 \cdot \sum(TE) + \omega_3 \cdot \sum \Delta OE}{CT} + \frac{\omega_4 \cdot \sum C + \omega_5 \cdot \sum D + \omega_6 \sum O}{T} \tag{5.10}$$

where

- $e_{ij}$ is the evaluation result for the $i-th$ track/target in the $j-th$ scenario;
- $\bar{x}_{ij}$ is the vector of metrics (section 4), and;
- $\theta$ is the vector of parameters to optimize.

Thus, the extension of the evaluation function must allow assessing simultaneously:

- one or various targets per scenario: scenario $j$: $\{e_{1j}, e_{2j}, e_{3j}, ..., e_{Nj}\}$, and,
- various scenarios with several targets per scenario: $M$ Scenarios: $\{e_{11}, e_{21}, ..., e_{N_1 1}, ..., e_{1j}, e_{2j}, ..., e_{N_j 1}, ..., e_{1M}, e_{2M}, ..., e_{N_M M}\}$

First of all, two benchmark tables are computed so that the proposed technique can be evaluated:

- First of all, the parameters are adjusted/optimized by ES for each target. The fitness function is directly the resultant evaluation for each track:

$$Fit_{ij} = e_{ij} \tag{5.11}$$

  And finally, there are as many sets of parameters as targets.

$$\Theta_{ij} = \theta_{11}, \theta_{12}, ..., \theta_{N_1 1}, ..., \theta_{1j}, e_{2j}, ..., \theta_{N_j 1}, ..., \theta_{1M}, \theta_{2M}, ..., \theta_{N_M M} \tag{5.12}$$

- Then, the parameters are adjusted for each scenario of the training set. As a result of this, the parameters are not learned only for the best performance of a single target, but for all the targets that are moving in a scenario. In order to build the new fitness function, two aggregations operators are used, the maximum (MAX) and the sum (SUM), of each evaluation result.

  The fitness function is the resultant of applying the aggregation operators to the evaluation for each track:

$$Fit_j = \sum_i e_{ij} \tag{5.13}$$

  or

$$Fit_j = max_i(e_{ij}) \tag{5.14}$$

  There will be as many sets of parameters as scenarios.

$$\Theta_j = \theta_1, \theta_2, ..., \theta_j, ..., \theta_M \tag{5.15}$$

Finally, the proposed optimization-generalization technique adjusts the parameters for all the scenarios. The fitness function takes into consideration all the partial evaluations of each target in all scenarios. Again, the SUM and MAX operators are used as follows:

$$Fit = \sum_i \sum_j e_{ij} \tag{5.16}$$

or

$$Fit = max_i(max_j(e_{ij})) \tag{5.17}$$

There will be a single set of parameters:

$$\Theta = \theta \tag{5.18}$$

The approach is depicted in the figure 5.2. The definition of an evaluation technique allows automating the search of parameters and adjusting the whole system to the best performance in the cases considered.

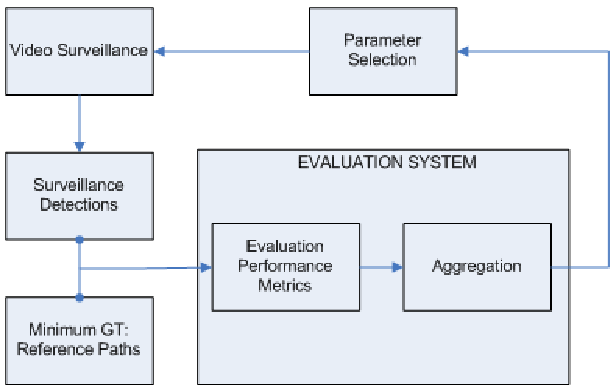


Figure 5.2: Approach for system adjustment

In fact, the maximum operator generally has a good performance for aggregation in multi-objective optimization problems with a complex trade-off required, as it was showed in a previous work related with optimization applied to a multi-constraint design problem [191]. The sum operator has been taken as a reference for comparison.

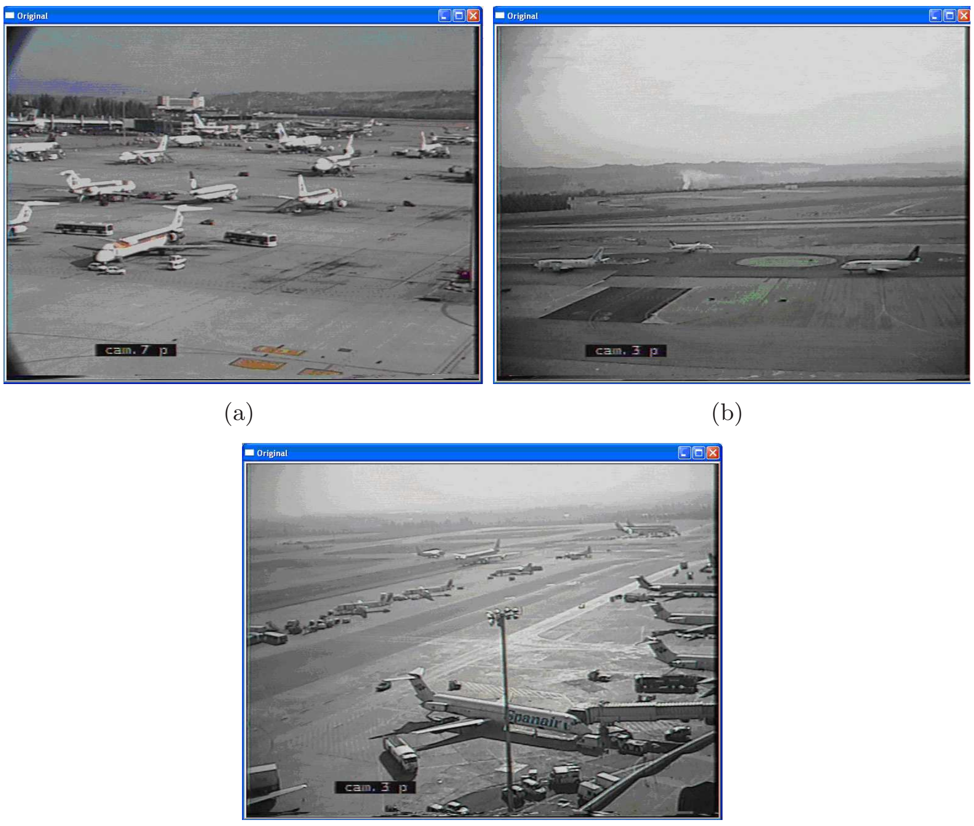
## 5.3 Optimization with Ground Truth

These experiments have been carried out with the evaluation metric with GT presented in section 4.2. As the author pointed out in that section, this evaluation technique proved to be robust and efficient. Nevertheless, the main drawback was the extraction of GT, a slow and tedious task.

The experiments in these sections were programed by means of C++ and using OpenCV library for the image processing computations.

This section shows the approached evaluation method, and the subsequent use of ES optimization, for the video tracking system. The eight parameters pointed out before are searched for the best performance of the surveillance system under different situations. In order to analyze the generalization capability, this search compares the most general optimization (the parameters for the best tracking of all targets of all videos) with two benchmark experiments (parameters adjusted for each target track and for each scenario).

The scenarios represent a good set for training the system as they are long and varied enough to cover the most common situations of surface movements in an airport.



(a) (b)

(c)

Figure 5.3: a) Video 1 b) Video 2 c) Video 3

The first scenario is a multiple-blob reconnection scenario (see Figure 5.3(a)). An aircraft is moving from left to right with partial appearance because it is hidden by other aircraft and vehicles parked in the parking places. Thus, there are multiple blobs which represent this aircraft that must be re-connected. At the same time there are four vehicles (three cars and a bus) moving on parallel roads or inner taxiways (upper taxiway and lower taxiway) whose tracks must be kept separated from this aircraft trajectory. There are five targets:

- A car which goes from the center to the left side of the screen following the lower taxiway.
- A car which heads toward the right side of the picture in the upper taxiway.
- A bus for passengers. This bus makes very slow and slight movements, practically non appreciable.
- Another car that appears in the right side of the pictures, drives all along the upper taxiway and finally disappears in the left side.
- And finally, big aircraft that heads from the left to right in the lower taxiway.

Picture 5.4 is a time-shot of the beginning, with the situation of the first three targets:

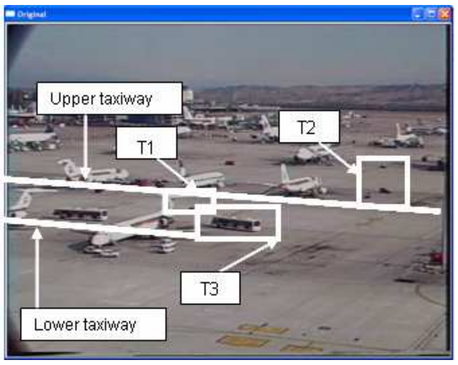


Figure 5.4: First scenario with targets number 1, 2 and 3, at the beginning

The second scenario shows three aircraft moving in parallel taxiways. The aircraft images overlap when they are crossing. This always occurs with uniform motion on straight segments. This second scenario presents three targets as shows picture 5.5.

Finally, the third scenario presents two aircraft moving on taxiways between airport parking positions. The airplanes move in different directions. The first conflict situation arises when one of the aircraft turns to the right and it is partially hidden by the other aircraft, which has already changed its direction by turning to the left. The second conflict situation happens at the end of the video sequence when another airplane appears on the right side of the taxiway. This new airplane is also partially hidden by the first aircraft that is disappearing from the field of view of the camera in that moment.

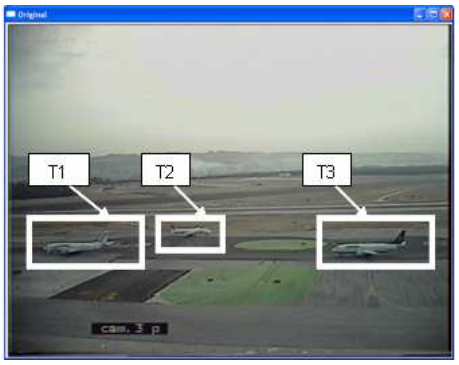


Figure 5.5: Picture of the second video for the targets number 1, 2 and 3

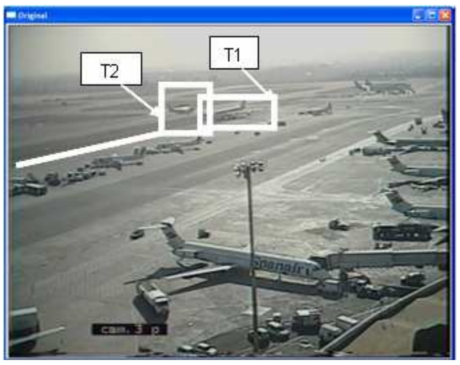


Figure 5.6: Third video with conflicting targets number 1 and 2

### 5.3.1 Tracking Evaluation and Optimization

As it was mentioned before, Evolution Strategies was the technique applied for optimizing the system performance. The fitness function is defined over the system output compared with references, and applying aggregation operators over partial evaluations to the set of trajectories considered to adapt the system. Since this function comprises error terms, the lower the evaluation function, the better the quality of the tracking system.

The size of population was reduced to the minimum in order to carry out the experiments within a reasonable time, while convergence to appropriate solutions was kept. The configuration of ES optimization is outlined below:

- Individual = $(THRESHOLD, MINIMUM\_AREA,$ $MARGIN\_GATE, MINIMUM\_DENSITY, CONFLICT,$ $VARIANCE\_ACCEL, MINIMUM\_TRACK\_AREA,$ $MARGIN\_INITIALIZATION)$;
- Size of population = 6+6;

- Adaptation factor: $\Delta\sigma = 0.5$;
- Selection scheme $(\lambda + \mu) - ES$, and,
- Maximum number of iterations = 200 (it was checked that the algorithm almost always has converged before the 200 iterations).

Regarding the evaluation function, the vector of weights was ($\omega_1$, $\omega_2$, $\omega_3$, $\omega_4$, $\omega_5$, $\omega_6$, $\omega_7$) =($10^4$, 1, 1, 1, $10^4$, $5 * 10^3$, $10^4$). Relatively high values were given to the continuity terms to enforce a robust behavior. The weights were chosen by means of a manual tuning (trial-and-error method), trying to adapt as much as possible the feedback of the evaluation system to the reality.

As an example of the tracking evaluation and direct optimization, figure 5.3.1 presents some tracking performance metrics, before and after optimization, taking a single-trajectory evaluation as a direct fitness function. This result corresponds to one of the objects (the big aircraft, target 5) in the first scenario described. As we can see, the system presents track losses (figure 5.7(a)) with an initial random set of parameters, effects that disappear after the optimization. Significantly, the X-Y errors (figures 5.7(b) and 5.7(c)) and overlap (figure 5.7(d)) degree between tracking and ground truth also are apparently improved. These errors can be measured only in situation of continuity (one-to-one correspondence with the ideal trajectory).

### 5.3.2 Benchmark Table: Optimization over a Single Target (Individual Optimization)

The optimization over single tracks, presented in previous paragraph, has been applied to every track in the three scenarios. The results are showed in table 5.1. This table is taken as the reference to compare with next experiments, which will apply the parameter optimization process over combinations of several targets/tracks simultaneously. The columns represent the target/track whose parameters are optimized to obtain the best performance and the best fitness function. The rows are the cross-evaluation over every target in each video. For each scenario, a sum row displays at the end the total error for the targets contained, applying the solution corresponding to that column.

The values in the diagonal represent the over-fitted cases, what it means that the evaluation is carried out with the parameters particularly adjusted to this case. That is the reason why this value is the minimum in each column. From now on, in some particular cases, the output parameters are not capable to detect the evaluated target (the undermatch-counter adds 1 unit to itself ($U_c$)), which is marked with the value 10,000 (a very high weigh indicating that the evaluation gives a low performance).

In order to illustrate the over-fitting effect in the results, we have selected 3 solutions from the 11 optimized sets of parameters. One target was selected per scenario, highlighted in the previous table:

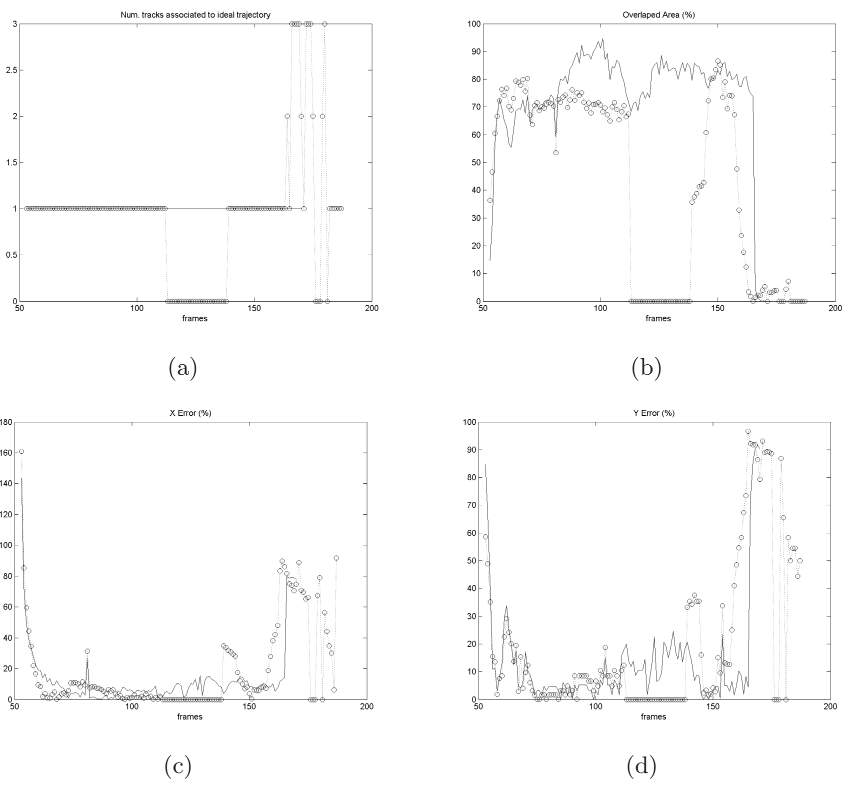


Figure 5.7: Direct optimization with a sample trajectory, continuity metrics:”-” optimized parameters, ”o...o” initial parameters

- target number 1 from video 1: param-Video1-T1;
- target number 1 from video 2: param-Video2-T1;
- target number 2 from video 3: param-Video3-T2;

In figure 5.8 we have three sets of columns, representing the performance of each solution when it is evaluated against the 11 targets. We can check that the parameters that optimize the track of target 1 (T1) in Video1 (param-Video1-T1) have the best cross evaluation value for the targets that belong to Video1 (the five first targets). In the same way, the parameters that optimize the track of target 1 (T1) in Video2 (param-Video2-T1) have the best cross evaluation value for the targets that belong to Video2 and worse in the rest of scenarios. And the same for the parameters in the optimization for a target belonging to Video3.

In order to have a reference figure with to be compared with the subsequent experiments, the figure 5.9 shows the average of all evaluations with the optimum parameters for each target. They correspond to the last column in table 5.1: each cell is the average of the cells in that row. In some way, these values represent the over-fitting effect since each

Table 5.1: Cross-evaluation of the optimized parameters for each single track

|  | Designed Scenario | Video 1 |  |  |  |  | Video 2 |  |  | Video 3 |  |  | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evaluation scenario | Track id | param-Video1-T1 | param-Video1-T2 | param-Video1-T3 | param-Video1-T4 | param-Video1-T5 | param-Video2-T1 | param-Video2-T2 | param-Video2-T3 | param-Video3-T1 | param-Video3-T2 | param-Video3-T3 |  |
| Video 1 | T1 | 1679,9 | 10493 | 10010 | 10000 | 5252,1 | 17140 | 8389,9 | 5851,7 | 2309,1 | 3068,1 | 1786,7 | 6907,31 |
|  | T2 | 2834,9 | 2816,2 | 2838,5 | 2837,9 | 2823,7 | 2838,5 | 2837,8 | 2836,9 | 2838,0 | 2838,2 | 2837,4 | 2834,36 |
|  | T3 | 1574,1 | 7686,8 | 4102,2 | 978,3 | 7987,1 | 7802,9 | 7569,4 | 6350,4 | 7486,8 | 12602 | 11494 | 6875,81 |
|  | T4 | 3753,0 | 5867,0 | 4106,5 | 257,9 | 4947,8 | 7100,0 | 4772,6 | 5155,3 | 3596,9 | 3625,1 | 2080,2 | 4114,75 |
|  | T5 | 221,4 | 6581,2 | 5451,0 | 10000 | 47,6 | 7250,1 | 5141,1 | 7293,3 | 156,4 | 4809,6 | 810,4 | 4342,00 |
|  | Sum1 | 110063 | 33444 | 26508 | 24074,2 | 21058,2 | 42131 | 28710 | 27487 | 16387 | 26943 | 19009 | 25074,26 |
| Video 2 | T1 | 1244,2 | 6879,2 | 493,5 | 2501,3 | 572,8 | 248,9 | 498,1 | 470,7 | 3757,2 | 529,7 | 592,4 | 1617,09 |
|  | T2 | 9091,7 | 8209,0 | 1343,0 | 6238,2 | 2393,5 | 1510,0 | 731,4 | 1521,7 | 6765,8 | 6905,9 | 6962,8 | 4697,54 |
|  | T3 | 2946,6 | 7946,6 | 759,8 | 768,7 | 1496,9 | 752,6 | 770,4 | 743,1 | 786,1 | 761,8 | 802,3 | 1684,99 |
|  | Sum2 | 13282 | 23034 | 2596,2 | 9508,1 | 4463,1 | 2511,6 | 1999,9 | 2735,4 | 11309 | 8197,5 | 8357,5 | 7999,62 |
| Video 3 | T1 | 2720,7 | 7029,5 | 9959,7 | 583,1 | 57546,1 | 6508,1 | 5862,6 | 6157,4 | 172,5 | 6756,8 | 8558,7 | 10168,65 |
|  | T2 | 7706,9 | 8104,1 | 13271 | 305,5 | 10189,5 | 9771,7 | 5080,6 | 12383 | 5079,4 | 91,2 | 10000 | 7452,99 |
|  | T3 | 11388 | 9460,7 | 12319 | 10000 | 14943,6 | 14421 | 11711 | 16524 | 9445,9 | 13398 | 459,8 | 11279,18 |
|  | Sum3 | 21815 | 24594 | 35550 | 10888,6 | 82679,2 | 30701 | 22654 | 35065 | 14697 | 20246 | 19018,5 | 28900,82 |
|  | Total Sum | 45161 | 81073 | 64655 | 44470 | 108200 | 75344 | 53365 | 65288 | 42394 | 55387 | 46385,1 |  |
|  | Faults | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  |

row contains the particular optimum value when the same target is used for optimization, and other ten cases corresponding to particular optimizations for other targets.

### 5.3.3 Optimization over Targets of a Specific Video: Scenario Optimization

The next step is the optimization for all the tracks included in a video, applying the addition and the maximum operators. The evaluation is carried out over all the trajectories per video. Subsequently, both operators are applied, minimization of the maximum (minimax) and minimization of the sum, in order to obtain the value that the Evolution Strategy algorithm uses to search for better individuals.

As the previous step, these calculations are computed in order to gradually build a

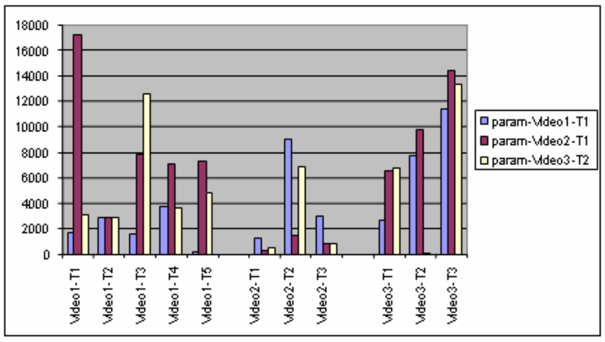


Figure 5.8: Evaluation of the obtained parameters over particular trajectories

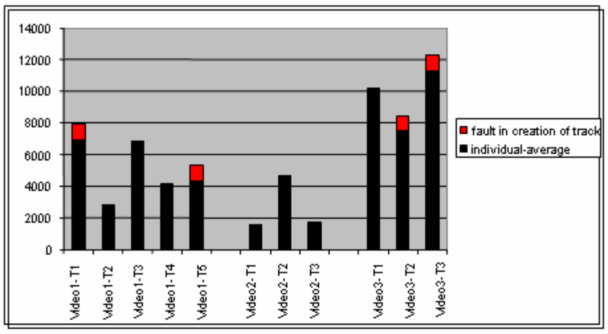


Figure 5.9: Average of evaluations over the parameters for each target

frame of comparison against the proposed optimization-generalization technique (see next section).

The table 5.2 shows the results for the maximum operator. There are three columns which correspond with the optimization carried out over each scenario or video. The over-fit effect can be checked again in the diagonal of the table. These values are the evaluation over the tracks used in the training process.

Following this method (table 5.2), the fitness value obtained is remarkably better than all previous results, which were calculated by training over a single track (table 5.1). For example, for the first video, the sum of the fitness gives a value of 6604.36. If we compare to the different sums given before for each previous solution in table 5.1, (10063.237, 33444.39, 26508.18, and 21058.22), it is easy to infer the improvement of the performance. The same argument can be followed for the rest of videos. In video number 2, the sum of fitness is 1983.73 while the values obtained in the first experiment were 2511.56, 1999.89 and 2735.41.

In an analogous way, the sum operator was applied later in order to obtain the input for the Evolution Strategy algorithm (see table 5.3)

Table 5.2: Cross evaluation for parameters optimized per scenario with minimax operator

| | Designed Scenario | Video 1 | Video 2 | Video 3 | Average |
|---|---|---|---|---|---|
| evaluation scenario | | param-Video1 | param-Video2 | param-Video3 | |
| Video 1 | T1 | 2148,38 | 6467,07 | 7118,28 | 5244,57 |
| | T2 | 2816,06 | 2838,03 | 2829,93 | 2828,00 |
| | T3 | 808,49 | 7571,34 | 7722,86 | 5367,56 |
| | T4 | 611,94 | 4296,65 | 2346,35 | 2418,31 |
| | T5 | 219,49 | 6012,59 | 6898,39 | 4376,82 |
| | Sum1 | 6604,36 | 27185,68 | 26915,81 | 20235,28 |
| Video 2 | T1 | 7761,07 | 501,94 | 498,34 | 2920,45 |
| | T2 | 4774,85 | 735,06 | 1709,53 | 2406,48 |
| | T3 | 5057,37 | 746,73 | 1901,62 | 2568,57 |
| | Sum2 | 17593,2 | 1983,73 | 4109,49 | 7895,50 |
| Video 3 | T1 | 327,80 | 8511,76 | 1321,73 | 3387,09 |
| | T2 | 352,39 | 6639,13 | 4445,21 | 3812,24 |
| | T3 | 10000 | 10007,40 | 2724,84 | 7577,41 |
| | Sum3 | 10680,19 | 25158,29 | 8491,78 | 14776,75 |
| | Total Sum | 34877,84 | 54327,70 | 39517,08 | |
| | Faults | 1 | 0 | 0 | |

Table 5.3: Cross evaluation for the parameters optimized per scenario with sum operator

| | Designed Scenario | Video 1 | Video 2 | Video 3 | Average |
|---|---|---|---|---|---|
| evaluation scenario | | param-Video1 | param-Video2 | param-Video3 | |
| Video 1 | T1 | 1691,76 | 2650,92 | 8046,10 | 4129,59 |
| | T2 | 2843,30 | 2842,96 | 4232,38 | 3306,21 |
| | T3 | 1141,46 | 1684,53 | 13112,50 | 5312,83 |
| | T4 | 541,82 | 10000 | 5826,94 | 5456,25 |
| | T5 | 177,96 | 10000 | 2024,31 | 4067,42 |
| | Sum1 | 6396,30 | 27178,41 | 33242,23 | 22272,31 |
| Video 2 | T1 | 1744,78 | 284,92 | 10000 | 4009,9 |
| | T2 | 9155,52 | 501,81 | 10000 | 6552,44 |
| | T3 | 4403,98 | 784,92 | 8095,01 | 4427,97 |
| | Sum2 | 15304,28 | 1571,64 | 28095,01 | 14990,31 |
| Video 3 | T1 | 7608,18 | 10000 | 631,79 | 6079,99 |
| | T2 | 6906,09 | 10000 | 2354,02 | 6420,03 |
| | T3 | 10000 | 10000 | 6432,97 | 8810,99 |
| | Sum3 | 24514,2 | 30000 | 9418,78 | 21311,01 |
| | Total Sum | 46214,85 | 58750,06 | 70756,02 | |
| | Faults | 1 | 5 | 2 | |

The values obtained by the sum operator show worse performance of the tracking system than previous case, as it can be observed in figure 5.10 comparing both operators. This figure shows the evaluation average over each track for the three experiments (last column of tables 2,3), instead of comparing the particular solutions obtained for each one of the three scenarios.

### 5.3.4 Optimization over Targets Belonging to the Whole Set of Videos: Global Optimization

This section shows the performance obtained by the proposed optimization technique in which the best set of parameters for all the targets in all videos is searched. As in the previous experiment, we will use the addition of every single fitness values (the results of the evaluation of the set of parameters over every single target) and the maximum among all the single fitness values.

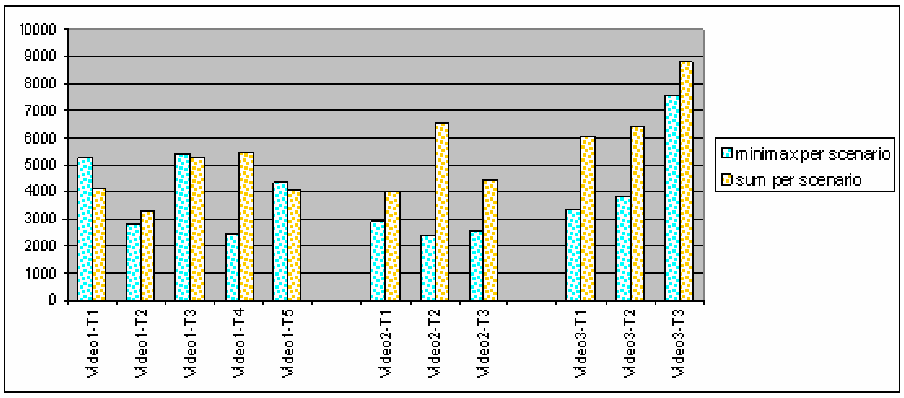


Figure 5.10: Comparison of the evaluations carried out with the maximum or addition operator over all the tracks that belong to a specific video

Table 5.4: Cross evaluation for parameters optimized for all scenarios with maximum operator

| | Designed Scenario | All Videos |
|---|---|---|
| evaluation scenario | | param-Videos |
| Video 1 | T1 | 2347,60 |
| | T2 | 2820,85 |
| | T3 | 1280,23 |
| | T4 | 3416,05 |
| | T5 | 1146,61 |
| | Sum1 | 11011,34 |
| Video 2 | T1 | 494,70 |
| | T2 | 2095,89 |
| | T3 | 787,59 |
| | Sum2 | 3378,18 |
| Video 3 | T1 | 5766,68 |
| | T2 | 5136,36 |
| | T3 | 3168,68 |
| | Sum3 | 14071,72 |
| | Total Sum | 28461,24 |

The results are shown in table 5.4 (for maximum operator) and table 5.5 (for the sum operator). The best aspect of this global optimization lies in the validity of the parameters to the whole set of videos, avoiding the over fitted values obtained in the previous experiments.

Table 5.4 shows the results for the case in which the maximum fitness has been taken in each iteration loop. The results applied to a specific scenario are good, but not better than the results applied to each target track obtained in the previous section. Nevertheless, the overall fitness sum for all the scenarios results much better considering evaluations over all situations. This fact indicates that the parameters fit well in the general case.

Finally, figures 5.3.4 and 5.3.4 show a comparison with all the cases and steps that we have presented in this work: individual optimization, scenario optimization and global optimization. As it has been mentioned, instead of comparing particular solutions for

Table 5.5: Cross evaluation for parameters optimized for all scenarios with sum operator

| | Designed Scenario | All Videos |
|---|---|---|
| evaluation scenario | | param-Videos |
| Video 1 | T1 | 2243,12 |
| | T2 | 2855,57 |
| | T3 | 7683,49 |
| | T4 | 1676,22 |
| | T5 | 105,63 |
| | Sum1 | 14564,03 |
| Video 2 | T1 | 7506,24 |
| | T2 | 10970,60 |
| | T3 | 4523,21 |
| | Sum2 | 23000,05 |
| Video 3 | T1 | 3465,03 |
| | T2 | 6181,07 |
| | T3 | 4363,25 |
| | Sum3 | 14009,35 |
| | Total Sum | 51573,43 |

individual targets or scenarios, each case is represented with the average evaluation over all cases. So, values in tables 5.4, 5.5 are compared with the last columns in tables 5.1, 5.2, 5.3. Furthermore, the total sum of all scenarios is shown below as the aggregated summary. It can be checked that the main goal of the work is achieved: the more general solution (more cases considered in the design), the better is the average performance. Moreover, it is relevant the good performance of the maximum operator in this process.

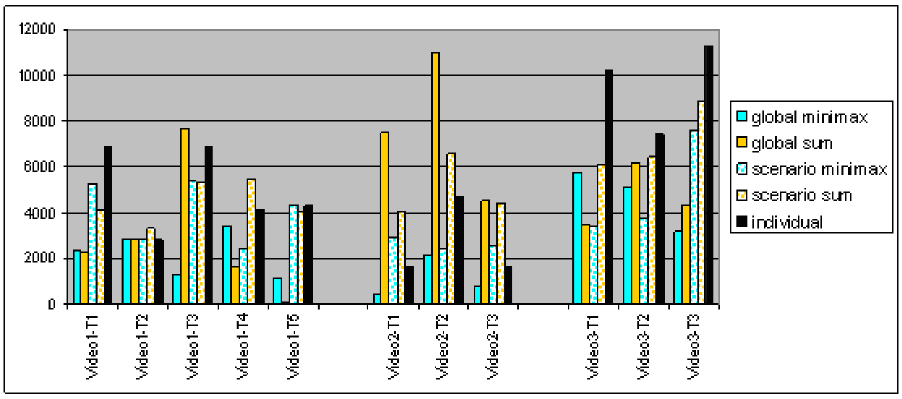


Figure 5.11: Comparison with all the cases and steps that we have presented in this work: individual optimization, scenario optimization and global optimization

### 5.3.5 Comparison with Other Existing Methods

In order to demonstrate the validation of our design technique, we compare our tracking system, tuned after the generalization process, against some existing methods. All the next tracking systems are available in the open software of Open Computer Vision Library [188]: CC (Connected Component Tracking), MS (Mean Shift Tracking), MSPF (Particle Filter based on MS weight), CCMSPF (Connected Component tracking and MSPF resolver for

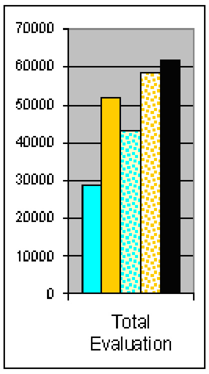


Figure 5.12: Sum of all the comparison components: individual optimization, scenario optimization and global optimization

collision), MSFG (Mean Shift Tracking with FG mask using) and CGA (Association by Canonical Genetic Algorithm).

As it can be checked in table 5.6 and we have pointed before, our method yields better generalization, obtaining a similar performance for all the cases. CCMSPF, CC, MS and MSFG have a remarkable behavior in the second scenario, that is, the easiest to analyze since there are only three big aircraft and no cars or buses. Nevertheless, all the new trackers present a bad performance when tracking more difficult scenarios that combine big aircraft and small moving vehicles. We can check how our optimized tracking system has a performance between 11000 and 14564 for these difficult cases, whereas the rest of systems present much higher values. As a result of this, we can conclude that the proposed optimization give us a trade-off in performance between simple and more complex scenarios, that is, similar performance is obtained for both cases; we obtain a set of parameters that provide good performance for different scenarios in an airport environment. In addition, we could highlight that good results are obtained with a very feasible tracker after tuning it by means of the optimization design technique that we propose. On the other hand, more sophisticated trackers give good performance for easy scenarios, whereas they can not make it so good for difficult situations where aircraft and small moving vehicles share the taxiway.

In this experiment, the author has proposed the application of Evolution Strategies to improve the global performance of a whole video tracking system with respect to real situations. The tracker is a flexible system based on easy rules. The Evolution Strategies has been used to adjust the parameters that control the operation of the tracking system in order to obtain a better performance under a wide range of common scenarios in an airport domain (big aircraft interacting with small vans, occluded trajectories, different manoeuvres, etc.).

The design technique that we propose consists in adjusting the parameters for all scenarios. To do that, the fitness function uses two aggregation operators: the sum and

Table 5.6: Comparison of our tracking system (Rules) after optimization against other tracking systems. The Rules tracking results are used as benchmark for comparison.

| Evaluation Scenario | Rules I (minimax) | Rules II (sum) | CCMSPF | CC | MS | MSPF | CGA |
|---|---|---|---|---|---|---|---|
| V1-T1 | 2347.60 | 2243.12 | 10095.70 | 10098.70 | 80127.80 | 80186.30 | 10063.00 |
| V1-T2 | 2820.85 | 2855.57 | 67.84 | 65.41 | 68.68 | 140120.00 | 81.70 |
| V1-T3 | 1280.23 | 7683.49 | 10302.90 | 11227.10 | 10145.70 | 10144.60 | 11425.70 |
| V1-T4 | 3416.05 | 1676.22 | 10081.10 | 10000.00 | 10000.00 | 10000.00 | 10000.00 |
| V1-T5 | 1146.61 | 105.63 | 58.29 | 73.63 | 78.75 | 11316.60 | 49.38 |
| Sum1 | 11011.34 | 14564.03 | 30605.83 | 31464.84 | 100420.93 | 251767.50 | 31619.78 |
| V2-T1 | 494.70 | 7506.24 | 63.63 | 66.40 | 72.90 | 13130.10 | 480.60 |
| V2-T2 | 2095.89 | 10970.60 | 66.70 | 65.26 | 84.47 | 13556.90 | 5770.31 |
| V2-T3 | 787.59 | 4523.21 | 65.12 | 66.60 | 76.76 | 11728.20 | 4568.79 |
| Sum2 | 3378.18 | 23000.05 | 195.45 | 198.26 | 234.13 | 38415.20 | 10819.70 |
| V3-T1 | 5766.68 | 3465.03 | 8362.22 | 1479.64 | 16231.30 | 7341.82 | 9959.01 |
| V3-T2 | 5136.36 | 6181.07 | 6526.68 | 6811.23 | 5195.48 | 7430.58 | 10284.40 |
| V3-T3 | 3168.68 | 4363.25 | 7145.38 | 6816.50 | 291.32 | 2728.06 | 3798.42 |
| Sum3 | 14071.72 | 14009.35 | 22034.28 | 15107.37 | 21718.10 | 17500.46 | 24041.83 |

the maximum. In addition, certain computations are calculated beforehand in order to have certain criteria to compare the final results. These computations represent a gradual analysis of the generalization or the capacity of the system to be generalized. The calculations are carried out in two steps:

- First of all, the parameters are adjusted for each target. That means that there are as many sets of parameters as targets.
- Then, the parameters are adjusted for each scenario. The fitness function uses two aggregations operators: the maximum and the sum of each evaluation result.

Thus, the study tests several scenarios and shows the improvement of the results from the most particular case to the general situation.

As a result, a significant improvement of the global vision system is achieved, in terms of accuracy and robustness. With this design technique based on the optimization, the inter-relation of parameters at different levels allows a coherent behavior under different situations. A generalization analysis has shown the capability to overcome the over-adaptation when particular cases are considered and a continuous improvement when additional samples are aggregated in the training process, comparing two different operators: sum and worst-case aggregation. Moreover, the author has compared our optimized system with other tracking systems in order to evaluate the correctness of the proposed design technique. The results show that the optimization allows obtaining the set of parameters that provides us stable performance in different scenarios in the airport domain, whereas more sophisticated trackers show great performance for easy scenarios and very bad outcomes for complex scenarios where there are aircraft and little moving vehicles.

## 5.4 Optimization with an Heuristic Fitness Function and Minimum Ground Truth

This section shows other ways of evaluating and obtaining a fitness function by using the same adjusting method, that is, Evolutionary Strategies. Moreover, although the context is still surveillance systems, the application changes: improving the detector stage by removing shadows and obtaining more compact blobs. In addition, the validation of the Evolutionary Strategies for the learning optimization proposed problem will be tested by means of the evaluation technique with minimum ground truth presented in section 4.3.

This section explains then an experiment in which Evolutionary Strategies are used to improve the quality in the detector, concretely in the segmentation stage. The parameters whose values are optimized in this case are the THRESHOLD of the detector and some extra parameters related to morphological operations.

This experiment aims to demonstrate the validation of Evolutionary Strategies to solve efficiently problems related to some of the parts of the surveillance process chain. As long as the fitness function and the parameters are carefully selected, the Evolutionary Strategies form a reliable tool for optimizing the performance of the surveillance system.

The experiments are programmed in C++, using again OpenCV library for the image processing computations. The figures displayed to compare the evaluation metrics were generated by MATLAB.

### 5.4.1 Introduction to the Problem

People tracking video-based systems present particular problems such as the multi fragmentation or low level of compactness of the resultant blobs due to the human shape or movements. This experiment shows how to improve the segmentation stage of a video surveillance system by adding morphological post-processing operations so that the subsequent blocks increase their performance. The adjustment of the parameters that regulate the new morphological processes is tuned by means of Evolution Strategies. Finally, the experiment evaluates the performance with the metrics proposed by the author. After the evaluation over a high number of video sequences, the results show that the shape of the tracks match up more accurately with the parts of interests. Thus, the improvement of segmentation stage facilitates the subsequent stages so that global performance of the surveillance system increases.

The detection of moving objects can be difficult for several reasons. We need to account for possible motion of the camera, changes in illumination of a scene and shadows, objects such as waving trees, objects that come to a stop and move again such as vehicles at a traffic light, etc. Once the moving objects have been identified, tracking them through the video sequence can also be difficult, especially when the objects being tracked are occluded by buildings or moved in and out of the frame due to the motion of the camera.

By segmentation stage, we mean the task of detecting regions that correspond to moving objects such as people and vehicles in video. This is the first basic step of almost every vision system since it provides a focus of attention and simplifies the processing on subsequent analysis steps. As we have said above, due to dynamic changes in natural scenes such as sudden illumination, shadows, weather changes, motion detection is a difficult task to process reliably.

Frequently used techniques for moving object detection are background subtraction, statistical methods, temporal differencing and optical flow. In this experiment, the author uses background substraction. Although background subtraction techniques perform well at extracting most of the relevant pixels of moving regions, they are usually sensitive to dynamic changes when, for instance, repetitive motions (tree leaves moving in windy day, see Figure 5.13), or sudden illumination changes occur.

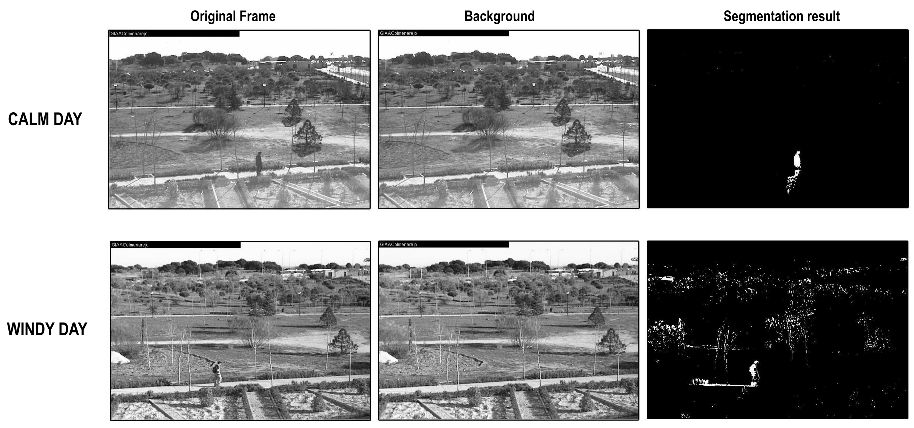


Figure 5.13: Different segmentation results obtained in different condition. The first row shows the excellent segmentation results in a calm day. However in the second row, due to the tree leaves in a windy day, we observe brightness changes almost everywhere in the image. Thus, the segmentation stage obtains worse performances.

### 5.4.2 Background Subtraction

The approach used for the background subtraction is the same than the one explained in section 5.2.1.

$$Detection(x, y) = [Im(x, y) - Back(x, y)] > THRESHOLD * \sigma \qquad (5.19)$$

In order to prevent targets from corrupting background statistics, the update is just performed for pixels not too near of a tracked target, using the tracking information in the detector. So, the statistics for k-th frame are updated as:

$$\begin{aligned} Back(x,y,k) &= \alpha Im(x,y,k) + (1-\alpha)Back(x,y,k-1) \\ \sigma^2(x,y,k) &= \alpha[Im(x,y,k) - Back(x,y,k-1)]^2 + (1-\alpha)\sigma^2(x,y,k-1) \end{aligned}$$

being x and y pixels out of predicted tracks.

In Figure 5.14 some segmentation results are depicted following this approach.

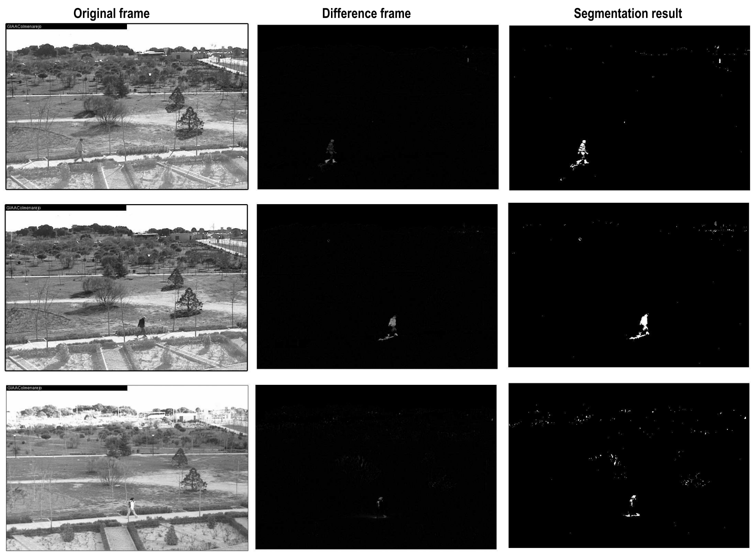


Figure 5.14: Instances of the segmentation stage. Although the results are good enough (third column), notice that in the third row, the object detected is rather difficult to track

### 5.4.3 Morphological Post-processing

As we can see in Figure 5.14, the last step (labeled as 'Segmentation result') obtains outstanding results. However, it seems obvious that we can improve the segmentation stage. A zoom of an individual of Figure 5.14 are depicted in Figure 5.15. The white pixels make up the individual and set up the foreground pixel map, in which there are unconnected and missing areas. Furthermore, in all over Figure 5.14 there is a lot of noise which can confuse later processing. Finally, the shadows are considered part of the moving target. This can lead to have false information about targets size and mass centers.

The goal of the segmentation stage is not only to produce foreground pixel maps as accurately as possible, e.g. by removing the special types of noise and shadows, but rather to make the individuals segmentation more visible and easier to process in the classification stage.

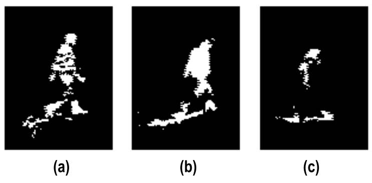


Figure 5.15: Zoom of an individual of Figure 5.14. It is clear that we can improve the segmentation stage. In the images appear unconnected and missing areas

Morphological operators have been implemented in order to improve segmentation results. The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. The most basic morphological operations are dilation and erosion. In a morphological operation, the value of each pixel in the output image is based on a comparison between the corresponding pixel in the input image and its neighbors. By choosing the size and shape of the neighborhood, a morphological operation can be tuned to be sensitive to specific shapes in the input image. In our case, an erosion operator has been chosen as first post-processing step in order to remove the noise. Then, we apply a dilation operator to improve the size and shape of the pedestrian.

Now, our problem is concerned with the selection of the size of the suitable structuring element and the number of iterations of the erode and dilate operations. We define the rectangular size of structuring elements and the number of iteration of erosion and dilate process by the next parameters: HORIZONTAL-SIZE-ERODE, VERTICAL-SIZE-ERODE, HORIZONTAL-SIZE-DILATE,
VERTICAL-SIZE-DILATE, ITERATIONS-NUMBER-ERODE and
ITERATIONS-NUMBER-DILATE. Besides, we have to establish another parameter involving in the segmentation stage: the THRESHOLD in Equation 5.19. The election of the values of these parameters makes a big difference in the performance of the system. Thus, in the next section, we show how to use Evolution Strategies in order to optimize these parameters.

### 5.4.4 Optimizing Morphological Parameters by Means of Evolution Strategies

We have implemented an Evolutionary Strategy (ES) for improving the segmentation stage by adjusting the parameters listed above. Regarding the operators, the type of crossover used in this work is the discrete one and the replacement scheme which is used to select the individuals for the next generation is $(\mu + \lambda) - ES$.

After the morphological post-processing of an image, its foreground pixel map consist of several blobs (i.e. coherent connected regions). In order to simplify the process, we represents the blobs by its bounding rectangle. Let $NB$ be the Number of Blobs in a foreground pixel map. In our experimentation we have been working with videos where there is only a pedestrian, and therefore we expect to found a short number of blobs in our ideal segmentation stage.

Let $Im$ and $\widehat{Im}$ be the image before and after the morphological post-processing, respectively. We define $Im(x, y)$ and $\widehat{Im}(x, y)$ as `true`, if and only if the pixel $(x, y)$ belongs to a moving object, respectively. We define the Density ratio, $D(B)$, of a blob, $B$, as:

$$D(B) = \frac{1}{n} \quad Card\{Im(x, y) \quad \wedge \quad \widehat{Im}(x, y)\}; \qquad \forall (x, y) \in B. \tag{5.20}$$

where $n$ is the number of pixels in the blob $B$ and *card* stands for the cardinality (i.e. number of pixels) of a set. The operator $'\wedge'$ (*and*) is applied to assess which part of the processed image contains detected pixels in the original image.

Let $AR(B)$ the Aspect Ratio of a blob, $B$. A blob is represented by its bounding rectangle. We define $AR(B)$ as:

$$AR(B) = \frac{width(B)}{height(B)} \tag{5.21}$$

where *width* and *height* stands for the bounding rectangle's width and height of a blob, respectively. Since, in our system, pedestrians are the object that we have to track, in contrast of shadows or noise, we expect to get a small value for the AR(B) ratio in every blob.

At last, the fitness function that we have to minimize is:

$$fitness = \alpha NB + \beta \sum_{\forall B \in Im} AR(B) - \gamma \sum_{\forall B \in Im} D(B) \tag{5.22}$$

where $\alpha$, $\beta$ and $\gamma$ are normalization coefficients.

### 5.4.5 Methodology and Results

Our general procedure for processing the video sequences was as follows:

1. Take a set of 5 random videos from the two video sequences groups;

Table 5.7: Optimization results. Notice that the structuring element shape rewards high and thin objects according to the pedestrians' shape.

| | |
|---|---|
| HORIZONTAL-SIZE-ERODE | 1 |
| VERTICAL-SIZE-ERODE | 4 |
| HORIZONTAL-SIZE-DILATE | 1 |
| VERTICAL-SIZE-DILATE | 4 |
| ITERATIONS-NUMBER-ERODE | 2 |
| ITERATIONS-NUMBER-DILATE | 2 |
| THRESHOLD | 15 |

2. Use the evolution strategies for adjusting the parameters of the morphological operators added to the segmentation stage. We implemented ES with a size of 10+10 individuals. This population is the minimum that assures the same result as if we had taken a higher number of individuals. The mutation factor of $\triangle\sigma = 0.5$ and the initial seed was fixed at 100;
3. Repeat the experiment with at least three different seeds;
4. If the results are similar, fix the parameters of the morphological algorithms for using them in all videos;
5. Take one video sequences set and the parameters obtained by the evolution strategy. Make the surveillance system work and collect all the people's tracks for each frame of each video sequence;
6. Evaluate these tracks and compare the results (with and without morphological algorithms in the segmentation stage), and,
7. Repeat the process for the second set of videos sequences from step 5.

In order to compare the effect of the morphological operators, we show some pictures before and after the application of the algorithms (Figure 5.16). The results of the optimization parameters are shown in Table 5.7. We can observe that the shadows and noises were removed so that subsequent stages of the surveillance system created more appropriate tracks according to the parts of interests. That is, the results had a real correspondence between the people we were interested in and the resulted tracks of the tracking system. This affected directly the track size, which was smaller as a consequence of the shadow elimination. This effect is displayed in the Table 5.8.

Finally, the effect on the whole surveillance system is showed in Figure 5.16. In order to have a more detailed idea of the system performance, the area under study is divided into 10 zones. Each zone is defined as a fixed number of pixels of the x-axis, the 10% of the horizontal size of the image. The absolute area and the transversal error show the mean, variance and maximum values for each of these two metrics.

Table 5.8: Numerical statistics of the Absolute Area (A) and Transversal Error (TE).

| | Before morphological operators | | | After morphological operators | | |
|---|---|---|---|---|---|---|
| | Mean | Max | Min | Mean | Max | Min |
| A | 7033 | 44890 | 111 | 3057.7 | 25636 | 203 |
| TE | 10.5 | 49.5 | 0.009 | 7.8 | 45.15 | 0.00055 |

All the metrics presented a remarkable improvement on the behavior of the total surveillance system. The absolute area decreased its mean value from 7033 to 3057.7 (see Table 5.8 and Figure 5.16(a) and 5.16(b)) due to the better adjustment of the tracks to the pedestrian shape. Second, the transversal error improved from a mean value of 10.5 to 7.8, which means that the gravity center of the people's track is closer to the ground truth function $f(x, y)$. Moreover, the last figures show that the number of losses for the tracks and the changes of direction decreased by a factor of 2.

As a final conclusion, we are able to confirm that the improvement in the segmentation stage provides more compact and accurate blobs to the subsequent blocks of the video surveillance system so that the performance of the surveillance system does increased.

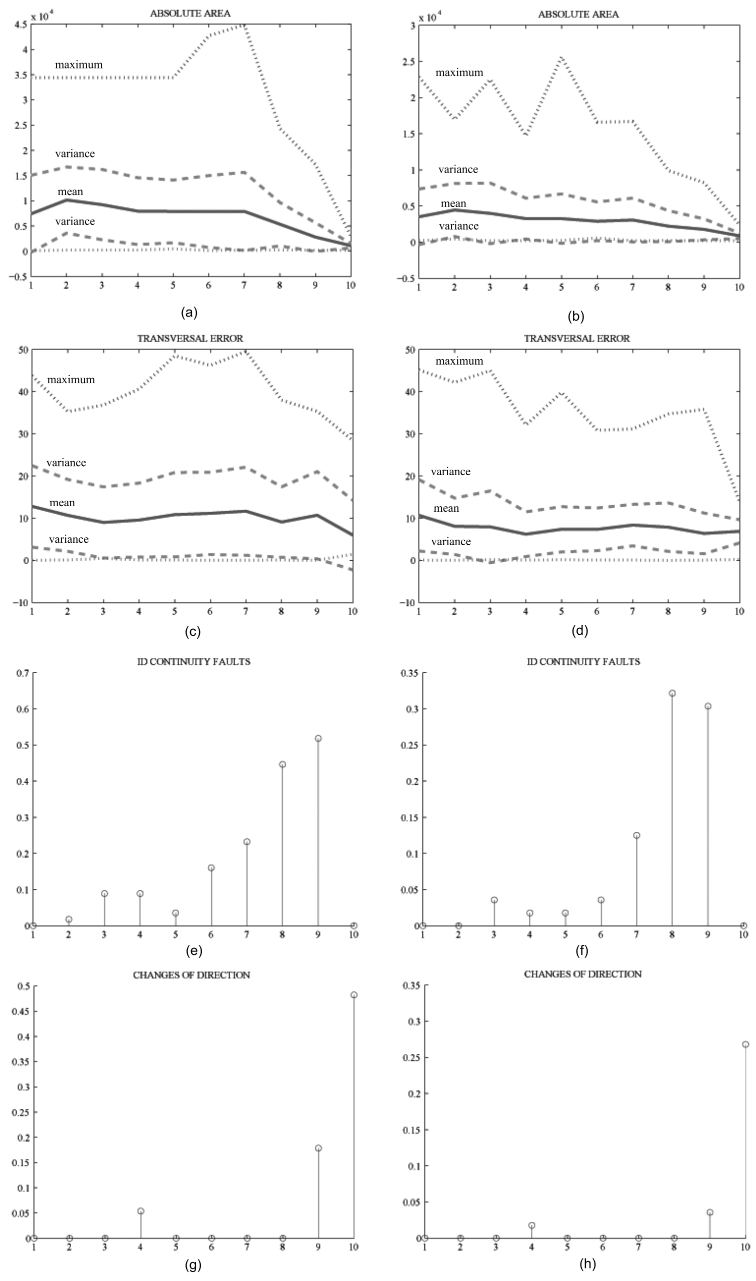


Figure 5.16: Metrics for people walking from right to left before and after the morphological process (left and right column respectively)

# Chapter 6

# Activity Recognition

## 6.1 Motivation

Once it is assured that the detector and tracking system are reliable and robust by having adjusted their parfameters for obtaining the best performance, we can trust that the final measures given by the system are suitable for application purposes.

The activity recognition process might be thought as an inverted triangle. First, it is important to determine what kind of activities must be classified. Then, according to these activities, the most representative features to distinguish these activities are computed by means of the measurements collected from the tracking system. And finally the selection of techniques to classify the activities takes into account these two points and choose the most suitable type of classifier.

In order to rate the importance of the features two processes must be carried out. The first one is a subjective study about what features might be important for the classification. Afterward, an objective technique must be applied to show the most relevant motion features for the activity recognition. The approach of this thesis consists of using a wrapper method that produces empirical evaluations for a classifier. The wrapper method is based on a machine learning (ML) method combining with a search method - the search is driven by the performance of the induced concepts [192]- [193].

Subsequently, the selection of the classifier is another key point. If a numerous set of labeled data is provided, the most appropriate decision is to use supervised classifiers such as decision trees, neural networks, support vector machines, etc. On the other hand, if labeled data is not provided, there are plenty of unsupervised learning classifiers such as those based on clustering techniques, for example, the Expectation Maximization algorithm (EM). But it might happen that only a few labeled data is provided and it would be desirable to use it for classification. Then, the most common solution is a semi-supervised learning in which the classifier is trained as unsupervised. Labels are assigned to the clusters and the validation is carried out in a supervised way by means of the labeled data. Therefore, taking into account the usual difficulty of generating labeled data, this thesis proposes the generation of a classifier in a semi-supervised way by labeling the generated clusters with the small number of labeled data.

In addition, the selection of classifiers can be facilitated by the study of the activities; for example, if they present a temporal pattern, then, any type of classifier that takes these states into account would be appropriate (Hidden Markov Models (HMMs), Finite State Machine (FSM), etc.).

This work proposes a statistical learning approach for generating a classifier for activity recognition. In particular, this thesis proposes the use of a new modeling of the emission probabilities for Hidden Markov Models by means of a non statistical parametric technique based on kernels; that is, kernel density estimation (KDE) applied to model HMMs by using an adaptation of the well known Baum-Welch algorithm to adjust the values of the parameters. These kernels are probability density functions of any type that are centered in each sample date of the training set. The non-assumption of probability functions for forming a pattern of the data allows us generating a classifier that can model any kind of data distribution.

In particular, this classifier will be tested in the environment of human activity recognition.

Finally, the results are compared with well known classifiers in order to have an idea of the good or bad performance of the proposed classifier.

Therefore, this section is divided as follows:

- Activity and Features Selection
- Kernel Density Estimation for HMMs: KDE-HMM Classifier
- Experiments

## 6.2 Activities and Features Selection

The feature selection constitute a huge field in which many type of measurements and computations can be carried out in order to extract the precise information for the proper activity classification. For example, features related to color, movement, shapes, silhouettes or contours and all the mathematical transforms such as Fourier transform, Hough, etc. can be computed. Nevertheless, this thesis is going to focus on a particular case which will be presented below.

This work is concentrated in low-level activities, that is, short-term human activities as those studied in the CAVIAR project [19]. These activities can be classified from short time sequences but more complex activities can be formed from this basic ones in case long sequences are considered [161].

The activities to classify are three:

- Inactive or if a person is not moving, just move his/her arms or very small translation is carried out.
- Walking, that is, if a person performs certain displacement.
- Running, if these displacements are larger.

The next step is the selection of features; in this case, we are going to extract low-level features that can be measured easily in short periods of time.

Thus, this thesis proposes the classification of activities by using only motion features described and calculated by a sequence of displacements of the 2D centroid and the height and width of each person's blob. This motion features can be provided by a simple tracker with no need of locating other parts of the body like head or hands.

The measures selected are the basic data that any simple tracker can provide: the 2D centroid position *(x,y)*, height and width of each person's surrounding box *(h,w)*. Then, we can compute a set features divided into two groups:

1. Velocity and Speed for a frame window of $(f)$ frames. These measurements are useful since they reflect the target translation among different frames.
   - velocity for the x-axis and y-axis:
   $$v^x_{f,i} = (x_i - x_{i-f}) \tag{6.1}$$
   $$v^y_{f,i} = (y_i - y_{i-f}) \tag{6.2}$$
   - speed:
   $$speed_{f,i} = \frac{1}{f}\sqrt{(x_i - x_{i-f})^2 + (y_i - y_{i-f})^2} \tag{6.3}$$
   - mean speed:
   $$mean\_speed_{f,i} = \frac{1}{f}\sum_{j=0}^{f-1}\sqrt{(x_{i-j} - x_{i-(j+1)})^2 + (y_{i-j} - y_{i-(j+1)})^2} \tag{6.4}$$
2. Height, Width and Area for a frame window of $(f)$ frames: These measurements reflect the height, width or area occupied by a target when carrying out the activities. For example, if a person is standing inactive, usually this person has the legs close to each other and the area is smaller than if the person is walking or running. Moreover, if a person is running the steps are longer and the arms form certain angle with the torso that makes the target area bigger than in the walking or inactive cases.

- difference of height, width and area:

$$diff\_height_{f,i} = (h_i - h_{i-f}) \tag{6.5}$$

$$diff\_width_{f,i} = (w_i - w_{i-f}) \tag{6.6}$$

$$diff\_area_{f,i} = |(height_i \cdot width_i) - (height_{i-f} \cdot width_{i-f})| \tag{6.7}$$

- mean difference of height, width and area:

$$mean\_diff\_height_{f,i} = \frac{1}{f}\sum_{j=0}^{f-1}(h_{i-j} - h_{i-(j+1)}) \tag{6.8}$$

$$mean\_diff\_width_{f,i} = \frac{1}{f}\sum_{j=0}^{f-1}(w_{i-j} - w_{i-(j+1)}) \tag{6.9}$$

$$mean\_diff\_area_{f,i} = \frac{1}{f}\sum_{j=0}^{f-1}\left|((h_{i-j} \cdot w_{i-j}) - (h_{i-(j+1)} \cdot w_{i-(j+1)}))\right| \tag{6.10}$$

Then, in order to select the more important features for classification, this thesis proposes to use a wrapper method based on a machine learning and a search method. In this particular case, J.48 and a Genetic Algorithm are the suggested ML algorithm and the search method respectively. To carry out this wrapper we employed WEKA 3.5.2 [194]. Subsequently, we decided to select a group of classifiers in order to compare their performance against the semi-supervised proposed classifier (KDE-HMM):

- J.48, Bayesian classifiers and a Classifier based on Rules (these are found in the WEKA software tool [194])
- Neuro-fuzzy classifier (the software tool this thesis used is found in NEFCLASS [195])
- HMMs modeled by Mixture of Gaussians

## 6.3 Kernel Density Estimation for HMMs: KDE-HMM Classifier

The author proposes a new classifier based on Hidden Markov Models (HMM). The novelty of this proposal will be the model used for the emission probabilities of the HMM, a density kernel estimation (KDE).

In order to find the unknown parameters of a Hidden Markov Model (HMM) it is very common the use of the Baum-Welch algorithm. Among all the probability density functions, the usual solution is to utilize the Baum-Welch algorithm with Mixture of Gaussians (GMs).

Then, based on the calculations of the Baum-Welch algorithm with Mixture of Gaussians, the author adapted the equations for the new KDE model.

This section explains this adaptation in three steps:

- First, it briefly introduces the similarities and differences between the KDE model and the traditional mixture of Gaussians. Moreover it gives the drawbacks of using GMs and the motivations to substitute them for KDE. Section 6.3.1.
- Second, in this work, the author deduces and adapts the traditional and plain Expectation Maximization algorithm with Mixture of Gaussians to the proposed Kernel Density Estimation model. This adaptation consists of the iterative calculation of the only parameter to estimate in the new model, the covariance $\Sigma^2$ (section 6.3.2)
- And finally, this work extends this adaptation to the Baum-Welch algorithm (EM for HMMs) with the Kernel Density Estimation model (KDE) (see section 6.3.3)

### 6.3.1 Similarities and Differences Between GMs and KDE

Kernel density estimation models a pdf (probability density function) as:

$$p_{KDE}(x) = \frac{1}{N \cdot h} \sum_{j=1}^{N} K\left(\frac{x - x_j}{h}\right) \tag{6.11}$$

where $K(\cdot)$ is a function with particular properties, called kernel, and h is the *kernel* bandwidth (or smoothing factor). By using the Gaussian kernel and noting the kernel bandwidth with $\sigma$, 3.55 becomes:

$$p_{KDE}(x) = \frac{1}{N} \sum_{j=1}^{N} G(x; x_j, \sigma^2) \tag{6.12}$$

Although 6.12 reduces KDE to another Gaussian mixture, the similarity between 3.65 and 6.12 is mainly apparent: first, in 3.65 the number of Gaussian components, $M$, is typically very low compared to the number of samples, $N$. Moreover, the weight, position and width $(\alpha, \mu, \sigma)$ of each component are determined as a trade-off over the sample set. In 6.12, instead, each Gaussian component is firmly located on a sample. The only parameter to be estimated is the variance, $\sigma^2$ (or, equivalently, the standard deviation, $\sigma$), common to all the Gaussian components.

This new modeling arises for the limitation of Mixture of Gaussians (GMs) and other kind of probability density functions in modeling certain distributions.

For example, one limitation is in the modeling of distributions which show more modes than the Gaussian components. In this case, one single Gaussian component has to be fit over multiple modes, thus leading to poor modeling based on eye judgment and relatively low likelihood. Although estimating the "right" number of modes is possible through procedures such as the mean-shift vector, it is often unfeasibly and time consuming.

The same poor modeling occurs when the distribution shows uniform regions which are inaccurately modeled by means of only a few Gaussians. KDE can overcome both these limitations. We argue that in some cases feature values obtained from human activities in videos such as speed and positions exhibit such uniform regions. Moreover, we argue that KDE could also lead to improved hidden Markov models of such activities. Figures 6.1 and 6.2 show an example of density estimation with a GM with two components based on 3.67 - 3.70 and with KDE based on our estimator for $\sigma$. For the latter case, the fitting of the distribution over the samples seems generally very good. As an obvious consequence, the likelihood obtained for KDE has always been greater or equal than that for GM in all our experiments.

For the latter case, the fitting of the distribution over the samples seems generally very good. As an obvious consequence, the likelihood obtained for KDE has always been greater or equal than that for GM in all our experiments. Obviously, this comes with an increased computational cost for the evaluation of equation 6.12 with respect to equation 3.65.

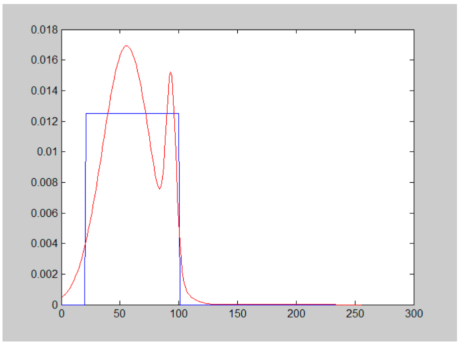


Figure 6.1: GM density estimation of a seemingly uniform distribution. The estimation seems generally inaccurate (initial parameters: $\alpha_1 = \alpha_2 = 0.5$; $\mu 1 = 85$, $\mu_2 = 170$; $\sigma_1 = \sigma_2 = 16$).

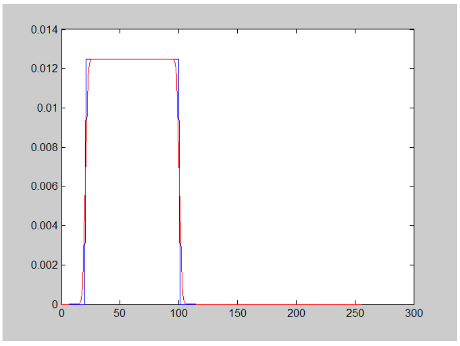


Figure 6.2: KDE from the same set of samples as Figure 6.1 with the proposed estimator for $\sigma$. The estimation seems generally accurate (initial parameter: $\sigma = 16$).

### 6.3.2 Estimation of Covariance $\Sigma^2$ in KDE by Means of Expectation Maximization

This section explains the problem of estimating an optimal value for $\sigma$ for the KDE model. All the calculations are carried out first for one dimension in order to simplify the formulation. Finally, the formula employed for the classifier will be extended to the multi-dimensional case (see equation 6.20) This work develops a mathematical algorithm based on the Expectation Maximization (EM) algorithm in order to estimate this parameter.

The estimation of the optimal variance, $\Sigma^2$, can be performed according to different criteria (see [24] for a comprehensive review). It is interesting to note that maximizing the likelihood for the KDE case leads to an obvious but impractical solution:

$$L_{KDE}(x_1, ..., x_N) = \prod_{i=1}^{N} p_{KDE}(x_i) = \prod_{i=1}^{N} \left( \frac{1}{N} \sum_{j=1}^{N} G(x_i; x_j, \sigma^2) \right) \to \infty \text{ as } \sigma \to 0 \qquad (6.13)$$

While several criteria could be chosen to determine an optimal value for $\sigma$, here we are interested in retaining the maximum likelihood framework so that our results can be more easily transferred to the estimation of parameters of a hidden Markov model. Thus, we use the pseudo-likelihood defined as [24]:

$$PL_{KDE}(x_1, ..., x_N) = \prod_{i=1}^{N} \left( \frac{1}{N} \sum_{j=1, j \neq x_i}^{N} G(x_i; x_j, \sigma^2) \right) \qquad (6.14)$$

Essentially, the probability of each $x_i$ sample is computed by excluding the Gaussian component centred on $x_i$ itself. Maximization of 6.14 can be performed in various ways.

Duin in [196] suggested computing the first derivative of 6.14 with respect to $\sigma$ and iteratively calculating its zero crossings. He reported that, by using a specially adapted version of the regula falsi algorithm, 5-20 iterations were needed to reach an accuracy of $10^{-3}$ in the value of $\sigma$ over a set of experiments. Here, we use the expectation-maximization algorithm by adapting update equation 3.69 to the case of a common value for $\sigma$ for all the Gaussian components. From [168], it can be easily proven that:

$$\sigma^{2\ new} = \frac{\sum_{l=1}^{M}(\sum_{i=1}^{N}(x_i - \mu_l^{new})^2 p(l/x_i, \Theta))}{\sum_{l=1}^{M}(\sum_{i=1}^{N} p(l/x_i, \Theta))} \tag{6.15}$$

provides the optimal value for $\sigma$ when such a value is constrained to be the same for all the $M$ Gaussian components. Again, if we use 6.15 directly for estimating $\sigma$ in the KDE case, we will converge to the undesired value $\sigma = 0$. Thus, for KDE we adjust 6.15 to reflect the definition of pseudo-likelihood given in 6.14 as:

$$\sigma^{2\ new} = \frac{\sum_{i=1}^{N}\sum_{j=1,x_j\neq x_i}^{N}(x_i - x_j)^2 p(j/x_i, \Theta)}{\sum_{i=1}^{N}(\sum_{j=1,x_j\neq x_i}^{N} p(j/x_i, \Theta))} \tag{6.16}$$

Equation 6.16 can be derived as follows: for GM, the derivation of (3.67- 3.69) is possible under the simplifying assumption that the generative model of each sample $x_i$ is not the whole GM, but only the best Gaussian component indicated by an unobserved indicator variable. For KDE, under the further assumption of pseudo maximum likelihood, the probability at 3.70 is assumed null when $x_j = x_i$; thus, 6.16 derives from 6.15.

Then, once the value of $\sigma$ has been derived, the next step will be to extend these equations to the well known Baul-Welch algorithm for the KDE case, instead of the GMs classical approximation.

### 6.3.3 Baum-Welch for KDE Models

This section shows the theoretical funds of the proposal of this work: KDE/HMM. That is, this work proposes kernel density estimation of the emission probabilities simultaneously with modeling the parameters by an adapted Baum-Welch algorithm.

The adaptation of the Baum-Welch algorithm is based on an extension of the equations derived from the Mixture of Gaussians case.

This adaptation will allow using the benefits of the maximum likelihood provided by Baum-Welch together with the non assumption of probability distribution of the data and the adaptability to any kind of distribution of data given by the Kernel Density Estimation Model.

Therefore, from 3.77-3.79 and the considerations addressed in Section 3.5.2, we can finally derive the update equations for the KDE case:

$$\alpha_{il}^{new} = \frac{\sum_{t=1,o_t \neq o_l}^{T} p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \tag{6.17}$$

$$\mu_{il}^{new} = o_l \tag{6.18}$$

$$\sigma_i^{2\ new} = \frac{\sum_{t=1}^{T} \sum_{t=1,o_t \neq o_l}^{T} (o_t - o_l)^2 p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} \sum_{t=1,o_t \neq o_l}^{T} p(l/o_t, \Theta)\gamma_i(t)} \tag{6.19}$$

where $o_t$ are the observed values in $t = 1..T$, $l$ is the $l - th$ kernel and $\gamma_i(t)$ is the probability of being in state $i$ at time $t$. It is important to highlight again that $sigma^2$ is the same value for all the kernel probabilities and this is the reason why it does not have the $l$ component.

In 6.18, the centers of the Gaussian components are not subject to update and sit, as usual, on the samples. 6.19 is the re-writing of 6.16 integrated by $\gamma_i(t)$. Again, we exclude the Gaussian component centered on the sample itself to prevent convergence to $\sigma^2 = 0$. We conveniently obtain this by setting $p_i(l/o_l, \Theta) = 0$ at the beginning of the iteration. Weight adjustment is needed also in the KDE case since observations need to be "dispatched" to the states in any case. To this aim, 6.17 is identical to 3.77 and just follows the way EM updates the GM weights. The only difference in 6.17 is that $p_i(l/o_l, \Theta)$ is, again, set equal to 0. In this way, the weights are essentially defined by the neighboring kernels, not the one centered on the point itself, like in update equation 6.19. Alternatives for weight assignment are possible, such as a simple $\alpha_{il} = \gamma_i(l)$, but they have not been experimented yet in real data. Overall, equations at 6.17-6.19 define the KDE/HMM proposed in this paper. Merely to prove that these results obviously extend to the multivariate case, we conclude this section by showing 6.19 for the case of multivariate observations:

$$\Sigma_i^{new} = \frac{\sum_{t=1}^{T} \sum_{t=1,o_t \neq o_l}^{T} (o_t - o_l)(o_t - o_l)^T p_i(l/o_t, \Theta)\gamma_i(t)}{\sum_{t=1}^{T} \sum_{t=1,o_t \neq o_l}^{T} p(l/o_t, \Theta)\gamma_i(t)} \tag{6.20}$$

where $o_t$ are the observed values in $t = 1..T$ and $\gamma_i(t)$ is the probability of being in state $i$ at time $t$.

## 6.4 Experiments: Human short-Term Activities Recognition in an Indoor Environment

This thesis takes a public video dataset to collect the 2-D measurements provided by a tracker. The dataset is the one of the CAVIAR project [19] and the author has selected two videos with the criterion of having the maximum number of activities: *Fight_RunAway1.mpg* and
*Fight_OneManDown.mpg.* These videos were recorded by the CAVIAR team with a wide angle camera lens in the entrance lobby of the INRIA Labs at Grenoble, France. The sequences have half-resolution PAL standard (384 x 288 pixels, 25 frames per second) and were compressed using MPEG2. The activities to classify are the ones mentioned before: Inactive (IN), Walking (WK) and Running (R). These are short-term activities, from which more complex activities can be formed such as "Fighting and leaving someone lying behind", etc.

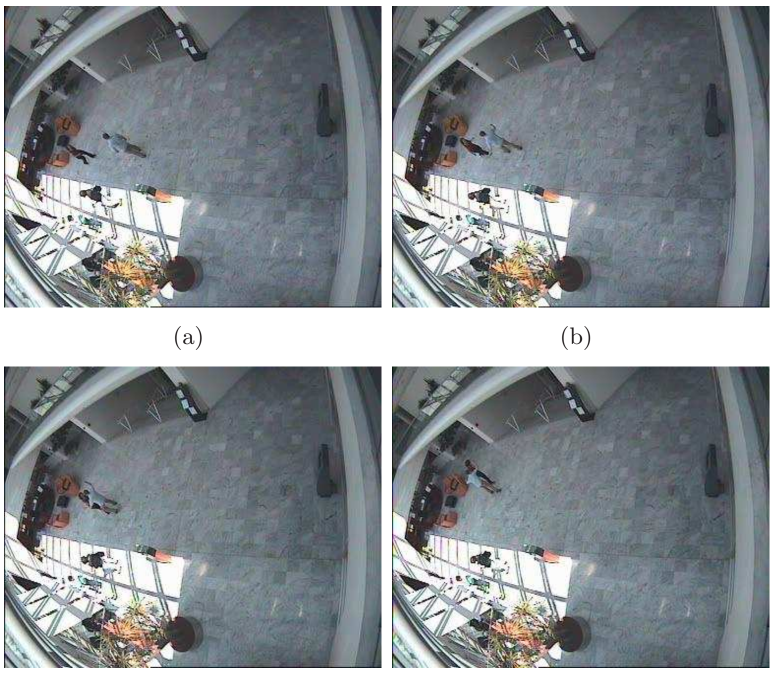


Figure 6.3: Frames 277 (a), 289 (b), 302 (c) and 368 (d) extracted from the video sequence *Fight_Chase.mpg*

The experiments are carried out into two parts:

1. The first one uses known classifiers and two sets of features in order to have certain benchmark to compare the proposal of the new classifier based on KDE/HMM.

2. The proposed KDE-HMM classifier is generated and tested with the same data set as in the previous step. Apart from the previous classifiers, all the results are directly compared with the HMM model with Mixture of Gaussians (GM-HMM) since this is the most similar classifier to our proposal.

The software tools employed in these experiments are:

- WEKA software [194] for the J.48, Bayesian classifiers and the Classifier based on Rules
- NEFCLASS [195] for the neuro-fuzzy classifer.
- MATLAB [197], [189] HMMs modeled by Mixture of Gaussians

We expect to overcome the classification performance by means of the use of the proposed classifier.

### 6.4.1 Selection of Features

Then, according to the extraction of features explained in Section 6.2, we compute a set of 40 features that are divided into two groups:

- Velocity and Speed for a frame window ($f$) of 3, 5, 15 and 25 frames.
- Height, Width and Area for a frame window ($f$) of 3, 5, 15 and 25 frames.

It is important to reduce the number of features as much as possible since the Kernel Density Estimation constructs the model from these features by centering a kernel probability function in each of these data samples. As the number of dimensions is bounded to the amount of features, an incremental of this number of features will imply that more resources are spent and more time is employed in building the classifier.

The most important of the 40 motion features defined above are selected then in two steps, discarding the ones not relevant. This selection of features is carried out by means of a wrapper that employs a J.48 and a GA as a classifier and search method respectively with 10 cross fold validation. The outcome of this filter (called *importance weight* from now on) is the number of times that each feature is used in each of the rounds of the 10 cross fold validation. The two steps are:

1. We decided to extract and choose the features whose importance is above or equal 70%.

2. Among all these features, we selected the two features with the most different histogram, $speed_5$ and $speed_{25}$, in order to train the selected classifiers with these two features.

The two features that presented the most separable histogram were:

$$speed_5 = \frac{1}{5}\sqrt{(x_i - x_{i-5})^2 + (y_i - y_{i-5})^2} \tag{6.21}$$

$$speed_{25} = \frac{1}{25}\sqrt{(x_i - x_{i-25})^2 + (y_i - y_{i-25})^2} \tag{6.22}$$

The histograms of these features (Figure 6.4) show the difficulty of this problem's resolution due to the subjectivity in the manual process carried out when labeling the activities. The challenging goal of the classifiers is to sort the classes taking these features with such a low level of separability.

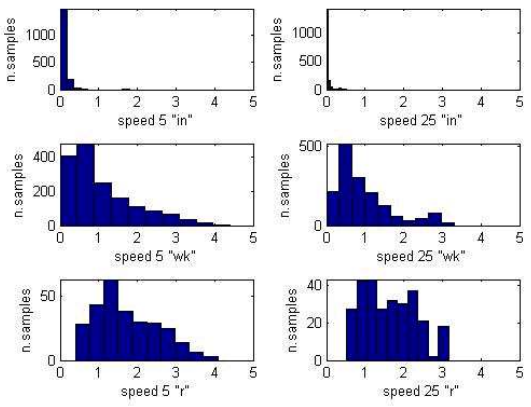


Figure 6.4: Histograms of features $speed_5$ (left column) and $speed_25$ (right column) for states "in" (inactive), "wk" (walking) and "r" (running)

Next table 6.1 shows the features (above the 70%) weight and their correspondent weight in pertecentage:

### 6.4.2 Initialization of Classifiers

Subsequently, we had to choose the set of the parameters for each of the classifiers.

The parameters for the different classifiers are adjusted as follows:

Table 6.1: Features selected by the feature selection algorithm: 13 Features above the 70% weight; 4 features equal the 100% weight

| | |
|---|---|
| $v^x_{3,i}$ | 100% |
| $v^y_{3,i}$ | 100% |
| $v^y_{5,i}$ | 80% |
| $v^x_{15,i}$ | 100% |
| $v^x_{25,i}$ | 80% |
| $v^y_{25,i}$ | 90% |
| $speed_{3,i}$ | 100% |
| $speed_{5,i}$ | 70% |
| $speed_{15,i}$ | 90% |
| $speed_{25,i}$ | 90% |
| $diff\_height_{15,i}$ | 80% |
| $diff\_height_{25,i}$ | 70% |
| $mean\_diff\_area_{25,i}$ | 90% |

- HMMs:

  The initialization of the parameters for both HMMs, KDE and GMs, are set as follows:

  - Number of Gaussians per state (only GMs case): $M = 2$. Where each $l$ Gaussian will be identified as: INACTIVE: $l = 1, 2$, WALKING $l = 3, 4$, RUNNING, $l = 5, 6$
  - Means (only for the GMs case). Two different values of means for two different experiments.
    $\mu^1_{l=1,2} = (0.2, 0.2)$ $\mu^1_{l=3,4} = (0.8, 0.8)$ $\mu^1_{l=5,6} = (1.5, 1.5)$,

    $\mu^2_{l=1,2} = (0.1, 0.1)$ $\mu^2_{l=3,4} = (2, 2)$ $\mu^2_{l=5,6} = (4, 4)$
  - Covariance. Three different values for three different experiments. Each covariance initializes all Gaussians modes or all the kernels.
    $\sum_1 = \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix}$,
    $\sum_2 = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}$,
    $\sum_3 = \begin{pmatrix} 9 & 8 \\ 8 & 9 \end{pmatrix}$
  - Weight for each of the Gaussian components: random in the case of GMs, and 1 for KDE.
  - State transition matrix (3 x 3): $A = \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{pmatrix}$

  - Initial probabilities: $\Pi = (100)$
  - Maximum number of iterations for the EM algorithm: $max - iter = 50$

- Neuro-fuzzy:

  - Number of variables in each fuzzy set: 2.
  - Form of the set: triangular.
  - Size of the rule base: automatically determine.
  - Rule learning Procedure: best per class
  - Learning Rate: 0.01
  - Maximum Number of epochs: 500
  - Minimum Number of epochs: 0
  - Number of Epochs after optimum: 100
  - Admissible Classification errors: 0

- Rest of classifiers: The default parameters of WEKA.

### 6.4.3 Results

We divided the data in training and validation data by following the criterion of having a representative number of active and running activities in both groups (since they are the less numerous that inactive). The data are divided into two sets of sequences for training and testing: one of 7 sequences containing 1975 data in total and another of 6 sequences and 1605 data. The criterion for this division was to have a representative number of "walking" and "running" activities in both groups (since they are less numerous than "inactive"). The first set is used for training while both are separately used for validation. Table 3 shows the total classification error on the training and validation sets for the GMs and the KDE.

Thus, the results of the experiments are showed in the following tables:

- Table 6.2 shows the results for both the HMMs classifiers: two GM-HMMs (HMM modeled by Mixture of Gaussians) and the proposed classifier KDE-HMM (HMM modeled by KDE).

- Table 6.3 shows the results for the J.48, Bayes Net, Naive Bayes, PART and Neuro-Fuzzy classifiers.

The first conclusion taken from the comparison between tables 6.2 and 6.3 is the out performance of the HMM classifiers whose classification errors in the validation stage are between the 15% and the 16.5%. Thus, table 6.3 shows that for all the cases, except for

Table 6.2: Total classification error for the training and validation sets

| Error (%) | | GM($\mu_1$) | GM($\mu_2$) | KDE |
|---|---|---|---|---|
| $\sum_1$ | Training | 23.59 | 16.86 | 14.48 |
| | Validation | 17.32 | 15.82 | 16.45 |
| $\sum_2$ | Training | 18.38 | 17.37 | 14.17 |
| | Validation | 15.07 | 15.32 | 16.01 |
| $\sum_3$ | Training | 23.59 | 23.59 | 14.48 |
| | Validation | 17.32 | 17.32 | 16.26 |

Table 6.3: Results for 2 features and 3 activities (inactive, walking and running)

| | J.48 | Bayes Net | Naive Bayes | PART | Neuro-Fuzzy |
|---|---|---|---|---|---|
| Training | 10.23% | 10.33% | 11.64% | 10.23% | 31.75% |
| Validation | 26.80% | 26.23% | 22.00% | 26.79% | 59.63% |

the Neuro-fuzzy classifier, the training stage shows an error around the 10% and the 12%. But, the validation stage presents an error between 22% and 27% of misclassifications. As a result of this, we can infer that these classifiers overfit for the training set, whereas they are not able to show good performance for another different set of data. This is clear in the confusion matrix for these classifiers. For example, the J.48 confusion matrix is chosen and it can be checked that many "walking" activities are classified as "inactive" (345) and all the "running" activities are classified as "walking" (78).

Table 6.4: Confusion matrix for J.48 and 2 features

| J.48 | Predicted | | | |
|---|---|---|---|---|
| Actual | | IN | WK | R |
| | IN | 569 | 7 | 0 |
| | WK | 345 | 606 | 0 |
| | R | 0 | 78 | 0 |

Moreover, if we fix our attention to Table 6.2, the experiments show the stable results of KDE/HMM independently of the initialization of its covariance parameter. It appears that the parameter space is very simple to search and the learning converges to the same value of $\Sigma$ irrespectively of very different initial values. Conversely, the GMs HMM obtains significantly different error rates depending on the initial values of its means and covariance parameters. This shows the limitation of the GM model as a highly parametric technique of difficult initialization. The error on the training and validation sets for the KDE model remains around 14-16% while for the GMs model varies between 15 and 24% depending on the different combinations of initial covariances and means. Other important point to highlight is that the experiment of Mixture of Gaussians with $\mu_1$ (GM($\mu_1$)) shows a peculiar behavior since the validation error is smaller than the the training error. To provide further detail into these results, Tables 6.5, 6.6 and 6.7 show the confusion matrix for the GMs and the KDE cases for experiments GM($\mu_1, \Sigma_1$), KDE($\Sigma_1$) and KDE($\Sigma_2$) in Table 6.2. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class (ground truth). Tables 6.5, 6.6 and 6.7 show that the better overall results of KDE also correspond to improved inter-class errors with respect to the GMs model.

Table 6.5: Confusion matrix for GM($\mu_1, \Sigma_1$)

| GM($\mu_1, \Sigma_1$) | Predicted | | | |
|---|---|---|---|---|
| Actual | | IN | WK | R |
| | IN | 560 | 16 | 0 |
| | WK | 108 | 719 | 124 |
| | R | 0 | 30 | 48 |

Table 6.6: Confusion matrix for KDE($\Sigma_1$)

| KDE($\Sigma_1$) | Predicted | | | |
|---|---|---|---|---|
| Actual | | IN | WK | R |
| | IN | 567 | 8 | 1 |
| | WK | 135 | 736 | 80 |
| | R | 0 | 40 | 38 |

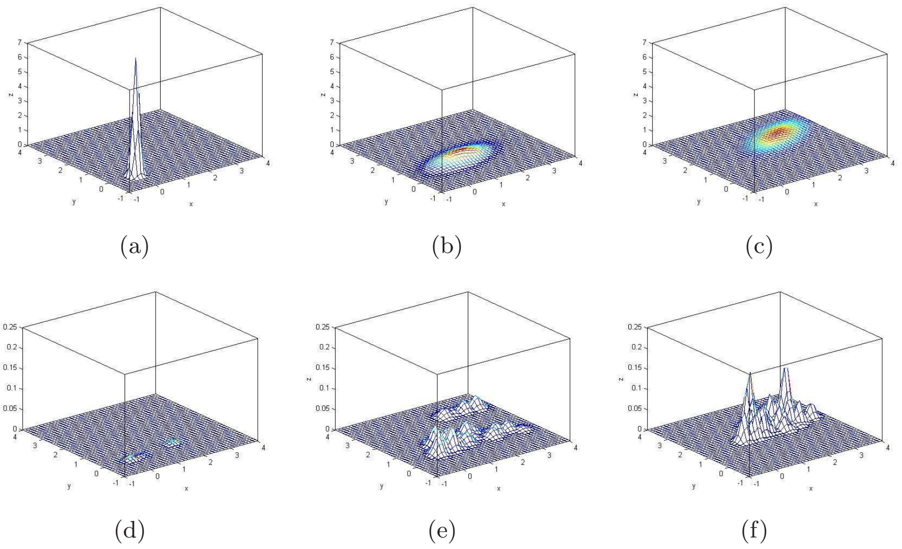
Table 6.7: Confusion matrix for KDE($\Sigma_2$)

| KDE($\Sigma_2$) | Predicted | | | |
|---|---|---|---|---|
| Actual | | IN | WK | R |
| | IN | 569 | 6 | 1 |
| | WK | 143 | 743 | 65 |
| | R | 0 | 42 | 36 |

Finally, Figure6.4.3 shows the pdf's of the emission probabilities for the GMs and KDE for experiments GM($\mu_2, \Sigma_2$) and KDE($\Sigma_2$) for each of the states. The pdf's show the intuitively different modelling of GMs and KDE. In particular, the KDE emission probabilities are not required to be compact and spontaneously adjust to model non-clustered data and with data with uniform regions.

### 6.4.4 Repetition of the Learning Classifiers for a Higher Number of Features

In order to check if the performance of the classifiers can be improved by means of adding more features, new experiments are carried out for J.48, Bayes Net, Naive Bayes, PART



(a) (b) (c)

(d) (e) (f)

Figure 6.5: GMs derived from the EM algorithm, experiment GM($\mu_2, \Sigma_2$) (in a scale 0-7) for inactive (a), walking (b) and running (c). KDE derived from the modified EM algorithm, experiment KDE($\Sigma_2$) (in a scale 0-0.25) for inactive (d), walking (e) and running (f)

and Neuro-Fuzzy. In addition, the experiment were repeated for HMMs too, but the performance did not improved. A possible reason why the performance decreased for the HMMs might be due to the difficulty of the probability density estimation techniques to model high-dimensional sets of data. The higher the dimension, the more difficult to model the data.

Following with the experiments for the rest of classifiers, the new features to use were selected like this:

- Four features that correspond to the ones with 100% of importance after the wrapper filter (see table 6.1): $v^x_{3,i}$, $v^y_{3,i}$, $v^x_{15,i}$ and $speed_{3,i}$.
- Thirteen features that correspond to the ones with at least 70% of importance after the wrapper filter (see table 6.1): $v^x_{3,i}$, $v^y_{3,i}$, $v^y_{5,i}$, $v^x_{15,i}$, $v^x_{25,i}$, $v^y_{25,i}$, $speed_{3,i}$, $speed_{5,i}$, $speed_{15,i}$, $speed_{25,i}$, $diff\_height_{15,i}$, $diff\_height_{25,i}$ and $mean\_diff\_area_{25,i}$.

The results for the 4 and 13 features are shown in tables 6.8 and 6.9 respectively. The training errors are very low for all the classifiers. Nevertheless, the validation errors remained high for almost all the classifiers. If we study the results individually taking into account the error rates for the validation stage:

- The J.48 present an error around the 25-28% for all the cases (2, 4 and 13 features) for the validation case.

- The Naive Bayes and Bayes Net improve or keep their performance increasingly to the number of features. It is remarkable the error rate achieve by the Naive Bayes classifier for 13 features: 15.82%.
- The classifier based on rules (PART) decreased its performance when the features increase. This is the most clear example of over-fit, since the training errors are much better as the number of features increases.
- Finally, the Neuro-Fuzzy classifier gives a good performance for the 13 features case. Nevetherless, the error rate is very high (32.26%).

Table 6.8: Results for 4 features and 3 activities (inactive, walking and running)

| | J.48 | Bayes Net | Naive Bayes | PART | Neuro-Fuzzy |
|---|---|---|---|---|---|
| Training | 4.91% | 8.81% | 10.33% | 5.46% | 32.31% |
| Validation | 25.36% | 21.12% | 23.00% | 20.81% | 50.60% |

Table 6.9: Results for 13 features and 3 activities (inactive, walking and running)

| | J.48 | Bayes Net | Naive Bayes | PART | Neuro-Fuzzy |
|---|---|---|---|---|---|
| Training | 0.40% | 4.55% | 8.71% | 0.25% | 12.09% |
| Validation | 28.84% | 20.93% | 15.82% | 43.17% | 32.26% |

In this section, it has been presented a modified hidden Markov model with KDE emission probabilities (HMM-KDE) and its use for activity recognition in videos. In the proposed approach, kernel density estimation of the emission probabilities occurs simultaneously with that of all the other model parameters thanks to an adapted Baum-Welch algorithm. This has allowed us to retain maximum-likelihood estimation while overcoming the known limitations of mixture of Gaussians in modeling certain data distributions such as uniform and non-clustered data. Experiments on activity recognition have been performed on the CAVIAR video surveillance database. Overall, the error on the training and validation sets with kernel density estimation remains around 14-16% while for the conventional Gaussian mixture approach varies between 15 and 24%.

The main advantage that we identify in the proposed KDE/HMM model is that its accuracy seems substantially independent from the choice of the initial value of its only parameter, the covariance matrix common to all its kernel components. On the contrary, the conventional GMs modelling of emission probabilities is a highly parametric technique and proves of challenging initialisation. Obviously in a way, the increased and more stable accuracy obtained by KDE comes at higher computational costs for both model estimation and evaluation as the number of kernels in KDE is much greater than that typical of GM components. However, this does not seem to represent a significant issue in applications such as activity recognition in videos as they are however dominated by the heavy low-level processing of foreground extraction and tracking.

Moreover, it has been proved that the classifiers based on HMMs obtain a better performance for this sort of short-term activities than the rest of classifiers tried in this section (J.48, Bayes Net, Naive Bayes, PART, Neuro-Fuzzy).

Future work will address the use of more defined motion features which provide more information about this kind of short-term activities. In addition, the KDE-HMM classifier will be improved for a obtaining a model less dependent of the number of features and dimensions (by using, for example, a mean shift algorithm to localize the more important peaks of the pdf of the classifier). Finally, the system will be improved to classify higher level activities derived from the short term ones.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

This dissertation proposed several intertwined objectives for the design of a surveillance system whose performance is thought for a wide range of situations in a specific environment.

First, two performance metrics for the detector and tracking system have been proposed in order to provide objective evaluation of these parts of the system. The first one is a ground-truth technique whose main drawbacks are the tedious and hard work of obtaining the ground-truth against which to compare which do not allow taking in a rapid way new videos for the training of the system in new locations. Nevertheless, this technique is a base from which to extend the metrics for the minimum ground truth performance metrics. The minimum ground truth performance metric is proposed as a response to the demand of adjusting the tracking system in a rapid and easy way depending on the location of this system. This adjustment demands new training video sequences from which to extract a ground truth, in this particular case, a minimum ground truth based on the regular pattern followed by pedestrians and vehicles in many circumstances (i.e. vehicles in a road, pedestrians in footpaths, etc.). The ground truth is defined as the union of straight lines that approximate the ideal pattern that the moving objects follow in normal conditions. Moreover, the proposed minimum ground technique allows obtaining statistical results since many videos can be evaluated at a time.

The experiments carried out in section 4.4 showed that the evaluation metrics reflect the behavior in the detection and tracking processes in an objective way. Thus, when more jitter and noise were detected and tracked by the system, the metrics showed a worsening in their measurements. Following the same line, in section 5.4, the detector was improved by means of certain morphological operations in the detected pixels. As a result, the evaluation technique showed an improvement in all the techniques.

The limitations of this technique lie on the constraint of defining the ground truth by means of the regular pattern that the moving objects must follow. Thus, the technique is applicable only in scenarios where the conditions are given.

Second, this work proposed an optimization technique based on Evolutionary Strategies, the selection of a varied set of training videos that represent the environment in which the surveillance system is going to work and the combination of the fitness function in several steps. The objective of this optimization is to obtain the fittest values of the parameters that adjust the tracking system for its best performance in the widest range of situations possible. Moreover, the literature showed that there are plenty of tracker systems. Nevertheless, many of them present complex algorithms which results in a slow functionality and for example they are not able to follow sudden movements. This thesis proposed to apply this optimization technique to an *agile* tracker based on easy rules, which are specified by . Then, another challenge of this thesis consisted of proving the good performance of this tracker after optimization against well known tracker algorithms.

The experiments carried out in the airport domain (section 5.3) showed how the optimization technique is able to adjust the values of the selected parameters in order to obtain a good performance for the widest range of variability. The range of variability is given by the distinct scenarios in which the system must work; these scenarios are represented by a set of training videos that are carefully selected so that they show the widest number of conditions in which the system must work. The combination of the fitness function in the proposed optimization technique is done by means of the aggregation operations of the sum and the maximum, which showed the best performance between both operators. Finally, the comparison of the tracker based on rules against other trackers (see section 5.3.5) showed that the rules tracker presented a stable performance for simple tracking cases (the only targets are aircraft of the same size) and complex scenarios (mixture of cars, buses and several airplanes), whereas the rest of tracker presented a good performance just for the simple scenarios. Thus, in this difficult cases, the rules tracker surpasses to the rest ones, whereas for the simple cases, the most elaborated trackers gave better performance.

Finally, this work had as the last objective the generation of a classifier in which a non-parametric statistic technique could model the distribution of data no matter the source generation of such data. Moreover, the thesis is focused on short-term activities that followed a time pattern that could be easily modeled by Hidden Markov Models classifiers. The proposal of this thesis consisted of a modification of the Baum-Welch algorithm in order to model the emission probabilities by means of a kernel density estimation model (KDE).

The results of the experiments carried out for a set of videos included in the public data of CAVIAR [19] (see section 6.4) showed stable results independently of the initialization of the model and good performance for the proposed algorithm KDE-HMM. If the comparison is carried out against the traditional Baum-Welch with Gaussian mixtures (GM-HMM) the results are not significantly better, but it is remarkable the stable results independently of the initialization. If comparing with the rest of classifiers (J48, Bayes Net, Naive Bayes, PART and Neuro-Fuzzy), the KDE-HMM surpassed all of them with the advantage of computing only 2 movement features. The performance of the rest of classifiers improved in a maximum of a 5% (for the Naive Bayes case) when the number of features is increased to 13.

The only drawback of the proposed classifier is the poor scalability, that is, if more features had to be considered, the model would become very slow due to the increase of the number of dimensions. The model is built by placing probability density functions (called kernels) in each of the data points which would derive in an intractable classifier for more than 6 features.

Then, the final aim of obtaining a robust surveillance system was achieved by means of the optimization technique and the generation of a classifier capable to model difficult distribution of the data. In addition, the system was trained and validated to work under the widest range of conditions for a given environment.

## 7.2 Future Work

The different objectives have been tested separately and the validation of them has been proved for specific problems. Thus, one of the immediate future works consist of testing the whole chain of proposed techniques in a single surveillance problem. The first step is to test the optimization/generalization method by using as fitness function the proposed evaluation function with minimum ground truth. The minimum ground truth was built by means of the union of straight segments. One extension may consist in the extraction of the ground truth by means of the union of straight or curved functions. These functions can be built by means of polynomial equations. Then, once the parameters are adjusted, the new and innovative Kernel Density Estimation algorithm for classification must be tested for the human activity recognition stage.

In addition, the shortcoming of intractability for the classifier when the number of dimensions increases can be solved by choosing carefully the main peaks of the probability density function and discarding the rest ones. This can be done by means of the Mean Shift algorithm [88] which is a method to find modes in a set of data samples that manifest an underlying probability density function (PDF). That would allow increasing the number of features when necessary. Then, the Mean Shift is applied as a complementary method to the KDE-HMM in order to select the most remarkable peaks of the probability density function built by the Kernel estimation.

Moreover, in order to improve the performance for the classification of the short-term activities, further extraction of features (i.e. optical flow) must be tested in order to decide the importance of the new features against the existing ones . Subsequently, the classification must be extended to complex activities based on these short-term activities ones.

Finally, new activities should be proposed in order to test the adaptability and performance of the classification model proposed in this work.

# Bibliography

[1] Claudio Sacchi and Carlo S. Regazzoni. A distributed surveillance system for detection of abandoned objects in unmanned railway environments. *IEEE Transactions on Vehicular Technology*, 49(5):2013–2026, 2000.

[2] N. Haering and K. Shafique. Automatic Visual Analysis for Transportation Security. *Technologies for Homeland Security, 2007 IEEE Conference on*, pages 13–18, 2007.

[3] S. A. Velastin J. P. Deparis and A. C. Davies. Cromatica project: A collection of telematic tools for improvement of safety in public transport. In G. Fabri C. S. Regazzoni and G. Vernazza, editors, *Advanced Video-Based Surveillance Systems*, pages 201–212. Eds. Norwell, MA: Kluwer, 1999.

[4] C.S. Regazzoni, G. Fabri, and G. Vernazza. *Advanced Video-Based Surveillance Systems*. Kluwer Academic Publishers, 1999.

[5] Yung-Sheng Chen Jun-Wei Hsieh, Shih-Hao Yu and Wen-Fong Hu. Automatic traffic surveillance system for vehicle tracking and classification. *IEEE Transactions on Intelligent Transportation Systems*, 7(2):175–187, 2006.

[6] A.J. Lipton, H. Fujiyoshi, and R.S. Patil. Moving target classification and tracking from real-time video. *IEEE Workshop on Applications of Computer Vision*, 14, 1998.

[7] I. Haritaoglu, D. Harwood, LS Davis, I.B.M.A.R. Center, and CA San Jose. W 4: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):809–830, 2000.

[8] E.D. Gelasca, T. Ebrahimi, M. Farias, M. Carli, and S. Mitra. Towards Perceptually Driven Segmentation Evaluation Metrics. *CVPR 2004 Workshop (Perceptual Organization in Computer Vision)*, page 52, 2004.

[9] J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation. *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 125–132, 2003.

[10] T. Schlogl, C. Beleznai, M. Winter, H. Bischof, A.C.V. GmbH-ACV, and A. Vienna. Performance evaluation metrics for motion detection and tracking. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 4, 2004.

[11] L.A. Zadeh. Fuzzy logic, neural networks, and soft computing. *World Scientific Series In Advances In Fuzzy Systems*, pages 775–782, 1996.

[12] LA Zadeh. Soft computing and fuzzy logic. *Software, IEEE*, 11(6):48–56, 1994.

[13] I. Rechenberg. *Evolutionsstrategie*. Friedrich Fromman Verlag, Stuttgart, Germany, 1973.

[14] D. da Silva Pires, R.M. Cesar-Jr, M.B. Vieira, and L. Velho. Tracking and Matching Connected Components from 3D Video. *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, 1530:05, 2005.

[15] D. Comaniciu and V. Ramesh. Real-time tracking of non-rigid objects using mean shift, July 8 2003. US Patent 6,590,999.

[16] B. Zhang, W. Tian, and Z. Jin. Joint tracking algorithm using particle filter and mean shift with target model updating. *Chinese Optics Letters*, 4(10):569–572, 2006.

[17] F. Cupertino, E. Mininno, and D. Naso. Elitist Compact Genetic Algorithms for Induction Motor Self-tuning Control. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3057–3063, 2006.

[18] M. Seeger. Learning with labeled and unlabeled data. *Inst. for Adaptive and Neural Computation, technical report, Univ. of Edinburgh*, 2001.

[19] R. Fisher. Caviar project, 2005. *Ground truth labelled video sequences.*

[20] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, 2004.

[21] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. *Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[22] Mohiuddin Ahmad and Seong-Whan Lee. Hmm-based human action recognition using multiview image sequences. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, 2006.

[23] M. Piccardi and O. Perez. Hidden Markov Models with Kernel Density Estimation of Emission Probabilities and their Use in Activity Recognition. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8, 2007.

[24] B.A. Turlach. Bandwidth Selection in Kernel Density Estimation: A Review. *CORE and Institut de Statistique*, 1993.

[25] A. Teschioni and C. S. Regazzoni. Performance evaluation strategies of an image processing system for surveillance applications. In G. Fabri C. S. Regazzoni and G. Vernazza, editors, *Advanced Video-Based Surveillance Systems*, pages 76–90. Eds. Norwell, MA: Kluwer, 1999.

[26] C.H. Chuang, J.W. Hsieh, and K.C. Fan. Suspicious Object Detection and Robbery Event Analysis. *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 1189–1192, 2007.

[27] S. Park and JK Aggarwal. Segmentation and tracking of interacting human body parts under occlusion and shadowing. *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 105–111, 2002.

[28] P.J. Figueroa, N.J. Leite, and R.M.L. Barros. Tracking soccer players aiming their kinematical motion analysis. *Computer Vision and Image Understanding*, 101(2):122–135, 2006.

[29] D. Thirde, M. Borg, J. Ferryman, F. Fusier, V. Valentin, F. Bremond, M. Thonnat, and O. Team. A Real-Time Scene Understanding System for Airport Apron Monitoring. *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems*, 2006.

[30] O. Perez, J. Garcıa, A. Berlanga, and J.M. Molina. Evolving parameters of surveillance video systems for non-overfitted learning. *Proceedings of the 7th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP'05)*, pages 386–395.

[31] K. Yuji, T. WATANABE, Y. SHIMOSAKODA, and S. NAKAGAWA. Automated Detection of Human for Visual Surveillance System. *Proceedings of ICPR*, 96:865.

[32] E. Trucco and K. Plakas. Video Tracking: A Concise Survey. *Oceanic Engineering, IEEE Journal of*, 31(2):520–529, 2006.

[33] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worrall, and K. Baker. An integrated traffic and pedestrian model-based vision system. *Proceedings of the Eighth British Machine Vision Conference (BMVC97)*, pages 380–389.

[34] N.T. Nguyen, S. Venkatesh, G. West, and H.H. Bui. Multiple camera coordination in a surveillance system. *Acta Automatica Sinica*, 29(3):408–422, 2003.

[35] H.H. Bui, S. Venkatesh, and G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden Markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):177–195, 2001.

[36] T.M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition*, pages 512–527, 2003.

[37] M.D. Schmill T. Oates and P.R. Cohen. A method for clustering the experiences of a mobile robot with human judgements. In *Proc. of the 17th National Conf. on Artificial Intelligence and Twelfth Conf. on Innovative Applications of Artificial Intelligence, AAAI Press*, pages 846—-851, 2000.

[38] S. Venkatesh N.T. Nguyen, H.H. Bui and G. West. Recognising and monitoring high-level behaviour in complex spatial environments. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1–6, 2003.

[39] Y. Ivanov and A. Bobick. Recognition of visual activities and interaction by stochastic parsing. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 852–872, 2000.

[40] N. Rota and M. Thonnat. Activity recognition from video sequences using declarative models. In *Proc. European Conference on Artificial Intelligence*, pages 673–680, 2002.

[41] F. Bashir and F. Porikli. Performance Evaluation of Object Detection and Tracking Systems. *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), June*, 5, 2006.

[42] LA Zadeh. Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 2(1):23–25, 1998.

[43] G. Welch and G. Bishop. An Introduction to the Kalman Filter. *ACM SIGGRAPH 2001 Course Notes*, 2001.

[44] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[45] M.J. Black and A.D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[46] 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, 2002.

[47] Z. Ghahramani. Unsupervised Learning. *Advanced Lectures On Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003: Revised Lectures*, 2004.

[48] S. Abney. *Semisupervised Learning for Computational Linguistics*. Taylor and Francis Ltd, 2007.

[49] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4), 2006.

[50] S. Kumar and M. Hebert. Discriminative random fields: a discriminative framework for contextual interaction in classification. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1150–1157, 2003.

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

[52] N. Friedman and S. Russell. Image Segmentation in Video Sequences:A Probabilistic Approach. *Annual Conference*, pages 175–181, 1997.

[53] AP Dempster, NM Laird, and DB Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[54] 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, 2003.

[55] D. Meyer, J. Denzler, and H. Niemann. Model based extraction of articulated objects in image sequencesfor gait analysis. *Image Processing, 1997. Proceedings., International Conference on*, 3, 1997.

[56] D. Koller, K. Daniilidis, T. Thórhallson, and H.H. Nagel. Model-Based Object Tracking in Traffic Scenes. *Computer Vision–ECCV'92: Second European Conference on Computer Vision, Santa Margherita Ligure, Italy, May 19-22, 1992, Proceedings*, 1992.

[57] Y.F. Ma and H.J. Zhang. Detecting motion object by spatio-temporal entropy. *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 265–268, 2001.

[58] R. Souvenir, J. Wright, and R. Pless. Spatio-Temporal Detection and Isolation: Results on PETS 2005 Datasets. *Urbana*, 51:61801.

[59] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000.

[60] D. Serby, EK Meier, and L. Van Gool. Probabilistic object tracking using multiple features. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2, 2004.

[61] X. Li. Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[62] I. Haritaoglu, D. Harwood, and LS Davis. Hydra: multiple people detection and tracking using silhouettes. *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*, pages 6–13, 1999.

[63] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice Hall Professional Technical Reference, 1982.

[64] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Articulated Body Posture Estimation from Multi-Camera Voxel Data. *Proc. of CVPR*, 2001.

[65] K. Toyama and A. Blake. Probabilistic Tracking with Exemplars in a Metric Space. *International Journal of Computer Vision*, 48(1):9–19, 2002.

[66] KN Plataniotis and AN Venetsanopoulos. *Color Image Processing and Applications.* 2000.

[67] KY Song, J. Kittler, and M. Petrou. Defect detection in random colour textures. *Image and Vision Computing*, 14(9):667–683, 1996.

[68] JL Barron, DJ Fleet, and SS Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[69] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical ROC curves. *Computer Vision and Image Understanding*, 84(1):77–103, 2001.

[70] IK Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, 1987.

[71] K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(1):51–65, 2005.

[72] Y.B. Shalom and TE Fortmann. Tracking and Data Association. *Academic-Press, Boston*, 1988.

[73] G. Kitagawa. Non-Gaussian State-Space Modeling of Nonstationary Time Series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987.

[74] D.J.C. MacKay. Introduction to Monte Carlo methods. *Learning in Graphical Models*, pages 175–204, 1998.

[75] R. Li. Tracking in clutter with nearest neighbor filters: analysis and performance. *IEEE Transactions on Aerospace and Electronic Systems*, 32(3):995–1010, 1996.

[76] Y.L. Chang and JK Aggarwal. 3D structure reconstruction from an ego motion sequence usingstatistical estimation and detection theory. *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 268–273, 1991.

[77] D. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, 1979.

[78] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at videoframe rates. *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 21–27, 1997.

[79] AD Jepson, DJ Fleet, and TF El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003.

[80] H. Tao, HS Sawhney, and R. Kumar. Object tracking with Bayesian estimation of dynamic layerrepresentations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):75–89, 2002.

[81] DP Huttenlocher, JJ Noh, and WJ Rucklidge. Tracking non-rigid objects in complex scenes. *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 93–101, 1993.

[82] F. Hausdorff. Set Theory. 1957.

[83] J. MacCormick and A. Blake. A Probabilistic Exclusion Principle for Tracking Multiple Objects. *International Journal of Computer Vision*, 39(1):57–71, 2000.

[84] Y. Chen, Y. Rui, and T.S. Huang. JPDAF Based HMM for Real-Time Contour Tracking. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1:543–550, 2001.

[85] M. Isard and A. Blake. CONDENSATION—Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[86] D. Cremers and C. Schnörr. Statistical shape knowledge in variational motion segmentation. *Image and Vision Computing*, 21(1):77–86, 2003.

[87] A.R. Mansouri. Region tracking via level set PDEs without motion computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):947–961, 2002.

[88] D. Comaniciu and P. Meer. Mean shift analysis and applications. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2, 1999.

[89] V.Y. Mariano, J. Min, J.H. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer. Performance Evaluation of Object Detection Algorithms. *International Conference on Pattern Recognition, ICPR02*, pages 965–969, 2002.

[90] P. Villegas and X. Marichal. Perceptually-weighted evaluation criteria for segmentation masks in video sequences. *Image Processing, IEEE Transactions on*, 13(8):1092–1103, 2004.

[91] J. Aguilera, H. Wildernauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of Motion Segmentation Quality for Aircraft Activity Surveillance. *Proc. Joint IEEE Int. Workshop on VS-PETS, Beijing*, 2005.

[92] J. Nascimento and J. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Transactions on Multimedia*, 8:761–774, 2006.

[93] S. Muller-Schneiders, T. Jager, HS Loos, and W. Niem. Performance evaluation of a real time video surveillance system. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 137–143, 2005.

[94] C.J. Needham and R.D. Boyle. Performance evaluation metrics and statistics for positional tracker evaluation. *International Conference on Computer Vision Systems (ICVS'03), Graz, Austria*, pages 278–289, 2003.

[95] L.M. Brown, A.W. Senior, Y.L. Tian, J. Connell, A. Hampapur, C.F. Shu, H. Merkl, and M. Lu. Performance Evaluation of Surveillance Systems Under Varying Conditions. *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (WAMOP-PETS)*, pages 1–8, 2005.

[96] T.H. Chalidabhongse, K. Kim, D. Harwood, and L. Davis. A Perturbation Method for Evaluating Background Subtraction Algorithms. *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Nice, France, Oct 2003.*

[97] P.L. Correia and F. Pereira. Stand-Alone Objective Segmentation Quality Evaluation. *EURASIP Journal on Applied Signal Processing*, 2002(4):389–400, 2002.

[98] PL Correia and F. Pereira. Objective evaluation of video segmentation quality. *Image Processing, IEEE Transactions on*, 12(2):186–200, 2003.

[99] Video understanding evaluation http://www.silogic.fr/etiseo/.

[100] A. Senior, A. Hampapur, Y.L. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance Models for Occlusion Handling. *IEEE Workshop on PETS, Kauai, Hawaii*, 2001.

[101] J.H. Piater and J.L. Crowley. Multi-modal tracking of interacting targets using gaussian approximations. *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 14:58, 2001.

[102] D. Pokrajac and L.J. Latecki. Spatiotemporal Blocks-Based Moving Objects Identification and Tracking. *IEEE Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 70–77, 2003.

[103] M. Wollborn and R. Mech. Refined procedure for objective evaluation of video object generation algorithms. *Doc. ISO/IEC JTC1/SC29/WG11 M*, 3448, 1998.

[104] IE Abdou and WK Pratt. Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE*, 67(5):753–763, 1979.

[105] K.C. Strasters and J.J. Gerbands. Three-dimensional image segmentation using a split, merge and group approach. *Pattern Recognition Letters*, 12(5):307–325, 1991.

[106] CE Erdem, B. Sankur, and AM Tekalp. Performance measures for video object segmentation and tracking. *Image Processing, IEEE Transactions on*, 13(7):937–951, 2004.

[107] CE Erdem, A.M. Tekalp, and B. Sankur. Metrics for performance evaluation of video object segmentation and tracking without ground-truth. *IEEE Intl. Conf. on Image Processing (ICIP)*, 2001.

[108] H. Wu, Q. Zheng, MARYLAND UNIV COLLEGE PARK DEPT OF ELECTRICAL, and COMPUTER ENGINEERING. *Self-Evaluation for Video Tracking Systems*. Defense Technical Information Center, 2004.

[109] M. Bray, E. Koller-Meier, NN Schraudolph, and L. Van Gool. Fast Stochastic Optimization for Articulated Structure Tracking. *Image and Vision Computing*, 25(3):352–364, 2007.

[110] T.P. Tian, R. Li, and S. Sclaroff. Tracking Human Body Pose on a Learned Smooth Space. *IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, 2005.

[111] RC Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 1, 2001.

[112] J. Paterson and A. Fitzgibbon. 3d head tracking using non-linear optimization. *Proc. BMVC*, pages 609–618, 2003.

[113] J. Berclaz, F. Fleuret, and P. Fua. Robust People Tracking with Global Trajectory Optimization. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 744–750, 2006.

[114] D. Comaniciu and V. Ramesh. Mean shift and optimal prediction for efficient object tracking. *Image Processing, 2000. Proceedings. 2000 International Conference on*, 3, 2000.

[115] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[116] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[117] H.T. Chen and T.L. Liu. Trust-region methods for real-time tracking. *Proc. 8th Intl. Conf. on Computer Vision*, pages 717–722.

[118] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical Recipies in C. *The Art of Scientific Computing*, 1988.

[119] S. Kirkpatrick, CD Gelatt Jr, and MP Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671, 1983.

[120] MP Vecchi. Optimization by simulated annealing. *Science*, 1995.

[121] L. Davis. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1987.

[122] DT Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.

[123] A. ROSENFELD. Multiresolution image processing and analysis. 1983.

[124] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press Cambridge, MA, USA, 1987.

[125] J. Mockus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994.

[126] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[127] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43–53, 2005.

[128] D.B. Fogel T. Back and Z.Michalewicz. *Evolutionary Computation: Advanced Algorithms and Operators*. Institute of Physics, London, 2000.

[129] D.B. Fogel T. Back and Z.Michalewicz. *Evolutionary Computation: Basic Algorithms and Operators*. Institute of Physics, London, 2000.

[130] J.H. Holland. Adaptation in natural and artificial systems, University of Michigan press. *Ann Arbor, MI*, 1975.

[131] J.R. Koza. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. 1992.

[132] L.J. Fogel. Evolutionary programming in perspective: The top-down view. *Computational Intelligence: Imitating Life*, pages 135–146, 1994.

[133] H.P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, 1977.

[134] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826, 1989.

[135] J. Kennedy and R. Eberhart. Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4, 1995.

[136] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 26(1):29–41, 1996.

[137] M.M. Eusuff and K.E. Lansey. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. *Journal of Water Resources Planning and Management*, 129:210, 2003.

[138] T. Bäck, C. Hillermeier, and J. Ziegenhirt. Routing optimization in corporate networks by evolutionary algorithms. *Natural Computing Series*, pages 739–753, 2003.

[139] D. Reichelt and F. Rothlauf. Reliable Communication Network Design with Evolutionary Algorithms. *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE AND APPLICATIONS*, 5(2):251, 2005.

[140] J. Periaux, M. Sefrioui, B. Stoufflet, B. Mantel, and E. Laporte. Robust genetic algorithms for optimization problems in aerodynamic design. *Genetic Algorithms in Engineering and Computer Science*, pages 370–396, 1995.

[141] E. Michielssen and D.S. Weile. Electromagnetic system design using genetic algorithms. *Genetic Algorithms in Engineering and Computer Science*, pages 345–369, 1995.

[142] C. Wang, Q. Wang, H. Huang, S. Song, Y. Dai, and F. Deng. Electromagnetic optimization design of a HTS magnet using the improved hybrid genetic algorithm. *Cryogenics*, 46(5):349–353, 2006.

[143] T. Back, J. Heistermann, and C. Kappler. Evolutionary algorithms support refueling of pressurized waterreactors. *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 104–108, 1996.

[144] F. Montes-Gonzalez. The Coevolution of Robot Behavior and Central Action Selection. *LECTURE NOTES IN COMPUTER SCIENCE*, 4528:439, 2007.

[145] BV Babu and SA Munawar. Differential evolution strategies for optimal design of shell-and-tube heat exchangers. *Chemical Engineering Science*, 62(14):3720–3739, 2007.

[146] D. Mester and O. Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers and Operations Research*, 34(10):2964–2975, 2007.

[147] J.G. Herrero, J.A.B. Portas, A.B. Jesus, J.M.M. Lopez, G.M. Vela, and J.R.C. Corredera. Application of Evolution Strategies to the Design of Tracking Filters with a Large Number of Specifications. *EURASIP Journal on Applied Signal Processing*, 2003(8):766–779, 2003.

[148] I. Rechenberg. *Evolutionsstrategie'94*. Frommann-Holzboog, 1994.

[149] H.P.P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc. New York, NY, USA, 1993.

[150] T. Back. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[151] D.H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[152] D.H. Wolpert. *The Mathematics of Generalization: The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning*. Perseus Books, 1995.

[153] B.W. Wah. Generalization and generalizability measures. In *IEEE Transaction on Knowledge and Data Engineering*, volume 11, pages 175–186, 1999.

[154] A.A. Ghorbani and K. Owrangh. Stacked generalization in neural networks: generalization on statistically neutral problems. *IEEE Proc. IJCNN, Washington, DC, USA*, pages 1715–1720, 2001.

[155] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM, 1990.

[156] O. Masoud and N. Papanikolopoulos. Recognizing human activities. In *Transactions. IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 157–162, 2003.

[157] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions. Pattern Analysis and Machine Intelligence*, 22(8):844–851, 2000.

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

[159] P. Pandit J. Ben-Arie, Z. Wang and S. Rajaram. Human activity recognition using multidimensional indexing. *IEEE Transactions on Pattern Anaysis and Machine Intelligence*, 24(8):1091–1104, 2002.

[160] Ju Han and B. Bhanu. Human activity recognition in thermal infrared imagery. *Transactionsgs IEEE CS Computer Vision and Pattern Recognition*, 3:17–17, 2005.

[161] Pedro Canotilho Ribeiroa and J Santos-Victor. Human activities recognition from video: modeling, feature selection and classification architecture. In *HAREM 2005 - International Workshop on Human Activity Recognition and Modelling*, Oxford, UK, September 2005.

[162] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1208–1221, 2004.

[163] AF Bobick and JW Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[164] JC Nascimento, MAT Figueiredo, and JS Marques. Segmentation and Classification of Human Activities. *Proc. HAREM International Workshop on Human Activity Recognition and Modelling*, 2005.

[165] Osama Masoud and Nikos Papanikolopoulos. A method for human action recognition. In *Department of Computer Science and Engineering University of Minnesota 2003*, 2003.

[166] A.Y. Ng and M.I. Jordan. On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. *log*, 1:1.

[167] MAF Figueiredo and AK Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002.

[168] J. Bilmes. *A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models.* Tech. Rep. ICSI-TR-97-021, University of California Berkeley, 1998.

[169] F. Dellaert. The Expectation Maximization Algorithm. *College of Computing, Georgia Institute of Technology, Technical Report No. GIT-GVU-02-20, Feb*, 2002.

[170] C.A. Peters and F. Valafar. Comparison of three nonparametric density estimation techniques using Bayes' classifier applied to microarray data analysis. *Proc. of the Int'l Conf. on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 119–125, 2003.

[171] K. Blekas and I.E. Lagaris. Split–Merge Incremental LEarning (SMILE) of Mixture Models. *LECTURE NOTES IN COMPUTER SCIENCE*, 4669:291, 2007.

[172] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM Algorithm for Mixture Models, 2000.

[173] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning.* MIT Press, 2006.

[174] AF Bobick and AD Wilson. A state-based approach to the representation and recognition of gesture. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(12):1325–1337, 1997.

[175] J. Ohya J. Yamato and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of Computer Vision and Pattern Recognition 1992*, pages 379–385, 1992.

[176] A.D. Wilson and A.F. Bobick. Parametric hidden markov models for gesture recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 21, pages 884—-900, 1999.

[177] M. Bicego and V. Murino. Investigating hidden markov models' capabilities in 2d shape classification. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 281–286, 2004.

[178] M.H. Yang and N. Ahuja. Extraction and classification of visual motion patterns for handgesture recognition. *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 892–897, 1998.

[179] YA Ivanov and AF Bobick. Recognition of visual activities and interactions by stochasticparsing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):852–872, 2000.

[180] F. Bremond and G. Medioni. Scenario recognition in airborne video imagery. *DARPA Image Understanding Workshop 1998*, pages 211–216, 1998.

[181] T. Wada and T. Matsuyama. Multiobject behavior recognition by event driven selectiveattention method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):873–887, 2000.

[182] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. IEEE*, volume 77, pages 257–286, 1989.

[183] H. Bourlard and N. Morgan. *Connectionist Speech Recognition. A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[184] E. Trentin. Nonparametric hidden markov models: Principles and applications to speech recognition. In *Computer Science*, volume 2859, pages 3–21. Lecture Notes in Computer Science Vol. 3449, 2003.

[185] M. Katz E. Andelic S.E. Kruger, M. Schaffoner and A. Wendemuth. Mixture of support vector machines for hmm based speech recognition. In *In Proc. 18th Int. Conf. on Pattern Recognition*, volume 4, pages 326–329, 2006.

[186] F. Lv, J. Kang, R. Nevatia, I. Cohen, and G. Medioni. Automatic tracking and labeling of human activities in a video sequence. *Proceedings of the 6th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS04).*

[187] B. Stroustrup. *The C++ Programming Languge.* Addison-Wesley Publishing Company, 1997.

[188] Opencv (2007) http://intel.com/technology/computing/opencv/index.htm.

[189] U. Matlab. The Mathworks Inc. *Natick, MA*, 2003.

[190] S. Blackman and R. Popoli. Design and analysis of modern tracking systems(Book). *Norwood, MA: Artech House, 1999.*, 1999.

[191] A. Berlanga J. M. Molina G. de Miguel J. R. Casar J. Garcia, J. A. Besada. Application of evolution strategies to the design of tracking filters with a large number of specifications. 8:766–779, 2003.

[192] DM Santoro, ER Hruschska Jr, and M. do Carmo Nicoletti. Selecting Feature Subsets for Inducing Classifiers Using a Committee of Heterogeneous Methods. *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, 1, 2005.

[193] R. Kohavi and D. Sommerfield. Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 192–197, 1995.

[194] S. Garner. WEKA: The Waikato Environment for Knowledge Analysis. *Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[195] D.D. Nauck. Fuzzy data analysis with NEFCLASS. *International Journal of Approximate Reasoning*, 32(2-3):103–130, 2003.

[196] R.P.W. Duin. On the Choice of Smoothing Parameters for Parzen Estimators of Probability Density Functions. *IEEE Transactions on Computers*, 25(11):1175–1179, 1976.

[197] K. Murphy. Hidden Markov Model (HMM) Toolbox for Matlab. *online at http://www.ai.mit.edu/murphyk/Software/HMM/hmm. html.*