Universidad Nacional de Educacion a Distancia

Trabajo Fin de Master

Detección de primer plano y construcción de fondos panorámicos a partir de cámaras en movimiento

Alumno: D. Rafael Marcos Luque Baena

Tutor: Dr. D. Ezequiel López Rubio

Trabajo Fin de Máster para el Máster de Matemáticas Avanzadas Especialidad Estadística e Investigación Operativa

11 de septiembre de 2014

Índice general

Detección de Primer Plano y Construcción de Fondos Panorámicos a partir de Cámaras en Movimiento

1. Introducción

La capacidad para identificar y analizar los objetos de fondo y de primer plano en escenas interiores y exteriores utilizando cámaras en movimiento, es un prerrequisito fundamental para el diseño e implementación de numerosos **sistemas inteligentes de análisis de vídeo**. En un mundo marcado por el afán en la seguridad, la profusión de cámaras de vigilancia ha provocado que multitud de lugares tanto públicos como privados, cuenten con un sistema de cámaras monitorizadas (*Closed Circuit Television*, CCTV) que traten de detectar y analizar cualquier comportamiento que suceda en la escena observada. Este análisis se suele hacer de forma manual mediante operadores humanos que observan cada monitor y determinan las incidencias que ocurren. Sin embargo, este proceso es inviable cuando el número de cámaras es cada vez más numeroso y un operador debe analizar más de una cámara al mismo tiempo, además de la ineficacia del resultado debido a las más que frecuentes faltas de atención.

Existen diferentes aplicaciones de este tipo sistemas, tales como video vigilancia automática (monitorización de tráfico o de entornos susceptibles de ser analizados por temas de seguridad), minería *inteligente* de datos en secuencias de vídeo (indexación semántica y recuperación de información), interacción hombre - máquina basada en visión o mejoras tecnológicas en eventos deportivos (ojo del halcón en tenis o estadísticas de jugadores de fútbol obtenidas en tiempo real). Todas ellas requieren información sobre los objetos que están en movimiento para, a partir de ellos, descubrir información relevante.

Las cantidades masivas de vídeo generado por las cámaras de seguridad instaladas en toda clase de ubicaciones hacen que su revisión visual exhaustiva sea inviable; sólo en

FIGURA 1: A la izquierda, imagen de una cámara de videovigilancia que se ha convertido en parte del paisaje habitual en lugares públicos. A la derecha, monitorización de un conjunto de cámaras dentro del mismo sistema.

los Estados Unidos existen más de 30 millones de cámaras de vigilancia, que producen cerca de 4.000 millones de horas de grabación a la semana [1]. El objetivo es reconocer automáticamente las actividades que ocurren dentro del campo de visión de una cámara y detectar sucesos anormales o actividades anómalas. Sin embargo, el incremento de la investigación en videovigilancia no ha significado un gran auge en la utilización operativa de este tipo de sistemas, ni la consecución de una solución sistemática al problema.

Las propuestas clásicas en este ámbito suelen reducirse al análisis de secuencias de vídeo obtenidas mediante cámaras estáticas, dado que es mas fácil crear un modelo de la escena si la cámara no se mueve. No obstante, existe un interés cada vez mayor en aprovechar las capacidades de las cámaras de vigilancia tipo **PTZ** (*pan-tilt-zoom*), ver Figura 2, que pueden moverse horizontal (*pan*) y verticalmente (*tilt*) así como ampliar o reducir (*zoom*). Esta tarea es bastante compleja debido a varias dificultades que no concurren con cámaras estacionarias:

- **Movimiento de la cámara**: este tipo de cámaras a veces crean cambios en la imagen de fondo debido su propio movimiento, especialmente cuando la cámara rota, aumenta el zoom o cambia su punto de vista.
- **Movimiento del primer plano**: los objetos de primer plano en movimiento pueden desplazarse muy rápido, muy lento o a la misma velocidad del movimiento de la cámara. Adicionalmente, estos objetos pueden tener apariencia no rígida (e.g. personas) y pueden ser seguidos por sus propias sombras. La aparición de oclusiones entre un objeto de primer plano y el fondo o entre dos (o más) objetos en primer plano es bastante frecuente y es, en ocasiones, dificil de corregir. Por último, los objetos de primer plano relativamente pequeños (como una pelota de tenis) no debe ser mal clasificados como valores espurios en secuencias con presencia de ruido.
- **Dinámica del fondo**: existen objetos que pueden aparecer o desaparecer del fondo de la escena (e.g. la entrada o salida de vehículos cuando están aparcados),

FIGURA 2: Imagen de varios modelos de cámaras PTZ.

con lo que la solución propuesta debe reaccionar rápidamente para considerar los cambios y adaptar el modelo de fondo. Por otra parte, los objetos estacionarios de fondo en constante movimiento, tales como la ondulación de las ramas de los árboles debido al viento, la nieve, la lluvia o las olas del mar en una escena de costa, no deben ser modelados como objetos que se mueven.

- **Fluctuaciones causadas por las condiciones de adquisición**: Si el contexto en el que se realiza la adquisición es dinámico (e.g. ruido, iluminación, color, contraste, resolución,...), existe la posibilidad de que objetos *sin sentido* aparezcan en la escena debido a cambios globales (e.g. cuando el sol está cubierto / descubierto por las nubes) o locales (e.g. sombras, reflejos y reflexiones rroyectadas por los objetos en movimiento)

La principal ventaja de utilizar cámaras con movimiento es la **reducción de costes**, pues no es necesario colocar un cámara en cada esquina de la escena a observar, si no que con una sola cámara es posible barrer a intervalos constantes y frecuentes la escena completa.

Los sistemas *inteligentes* de análisis de vídeo están generalmente diseñados utilizando una estructura dividida en tres fases:

- La primera etapa consiste en realizar un procesado a bajo nivel usando como entrada una secuencia de vídeo, cuyo objetivo es detectar **objetos en movimiento** que aparezcan en la escena. Para ello, se utiliza información espacial y temporal de los fotogramas que componen el vídeo, y cuyo análisis permitirá diferenciar, de forma robusta, entre regiones del fondo (estáticas) y de primer plano (dinámicas).
- La segunda etapa de un sistema de vídeovigilancia consiste en realizar un **seguimiento** de los objetos de interés en el tiempo y obtener propiedades temporales de los mismos (actividades).

- Después de recopilar información sobre las características de los objetos, tales como tipo, trayectoria, tamaño y velocidad, el último paso tratará de detectar el **comportamiento** de cada uno de estos objetos para entender y analizar lo que está ocurriendo en la escena.

En general, la primera etapa es considerada como la parte crucial del sistema debido a que el resto de módulos dependen en gran medida de ella. Este trabajo estará enfocado expresamente en la **etapa de detección de objetos en movimiento**, siendo su dificultad especialmente elevada debido a que el fondo de la escena también estará en movimiento. Para esta tarea, los pasos a seguir serán:

- Generar un **fondo panorámico** utilizando técnicas de emparejamiento de puntos entre fotogramas consecutivos. Básicamente, consistirá en una compensación de la cámara, que tratará de determinar de manera robusta una transformación lineal que exprese el sistema de coordenadas de un fotograma en términos de un sistema de coordenadas global (panorámico). Se realizará en dos fases: una segmentación inicial para ver cuales son las regiones más fiables para hacer la correspondencia de puntos, y la utilización de algoritmos específicos para obtener los puntos de interés.
- Identificar la **posición exacta** en el fondo panorámico a la que se corresponde el fotograma analizado.
- Aplicación de técnicas **probabilísticas multivariantes** utilizadas para la detección de objetos con cámara estacionarias de forma (casi) directa.

Por tanto, los **objetivos** a cumplir en este trabajo fin de máster pueden resumirse principalmente en dos:

- **Detectar objetos de primer plano** utilizando cámaras en movimiento.
- **Construir** un modelo de **fondo panorámico** que cubra la zona de interés a analizar.

Para ello, se ha desarrollado un **prototipo** que, dada un secuencia de imágenes (formato .avi o .jpg) de una escena analizada con una cámara en movimiento, determine los objetos de primer plano con la mayor exactitud posible. Nótese que analizar la información observada por una cámara IP en tiempo real, no es más que capturar los fotogramas observados y utilizarlos como entrada al prototipo desarrollado.

Si bien inicialmente propusimos centrarnos en analizar secuencias de vídeo de eventos deportivos, e.g. fútbol, baloncesto, balonmano, voleibol, ... (que tienen la particularidad de que una sola cámara no puede cubrir el campo de juego en su totalidad), en las

primeras fases del desarrollo de este trabajo detectamos una dificultad intrínseca a este tipo de escenas. El hecho de que el fondo de las mismas sea prácticamente homogéneo, hace que la correspondencia de puntos de interés no se realice correctamente, lo que conlleva a que la detección de movimiento no redunde en la calidad esperada.

Así, nos centraremos en el análisis de las secuencias obtenidas por cámaras **PTZ**, pero restringiendo su movimiento a desplazamientos horizontales y verticales, con objeto de que el sistema a desarrollar sea viable y no tratemos de alcanzar objetivos demasiado ambiciosos.

Como objetivos secundarios a alcanzar podemos citar los siguientes:

- Estudio, adaptación y desarrollo de modelos matemáticos probabilisticos que *aprendan* la información relativa al fondo en movimiento de la escena.
- Estudiar los fundamentos de las técnicas de detección de puntos de interés y correspondencia, y aplicarlas para la corrección del movimiento de la cámara.
- Robustez y fiabilidad en la salida de los algoritmos.
- Disponibilidad para una ejecución continuada, pues la entrada de los algoritmos consistirá en secuencias de fotogramas sin una duración definida, principalmente en el caso de la captura utilizando cámaras IP.

2. Modelo básico

En esta sección presentaremos un modelo de detección de objetos en movimiento basado en distribuciones probabilísticas, con la restricción de que el tipo de cámaras a utilizar serán de naturaleza estacionaria, es decir, **cámaras fijas** que capturarán la escena siempre desde la misma posición. Este modelo servirá de punto de partida para desarrollar un modelo que determine los objetos en movimiento sobre escenas que, a su vez, se encuentran en movimiento.

El objetivo principal de la detección de movimiento consiste en separar los píxeles asociados a los objetos del primer plano (o movimiento) de aquellos correspondientes a los objetos estacionarios del fondo. Existen diferentes técnicas en la literatura que tratan de abordar este problema desde diversos enfoques, desde los más simples que sólo están basados en una imagen de fondo, pasando por modelos estadísticos paramétricos que representan la tonalidad del color de cada píxel [2, 3], hasta modelos más ambiciosos que analizan y mantienen información sobre un conjunto de características de la escena [4]. La complejidad de estos modelos de fondo influye directamente en cómo de complejo o eficiente sea su proceso de actualización. Este tipo de algoritmos deberá lidiar con las dificultades intrínsicas del propio problema a resolver, que incluye la eliminación de

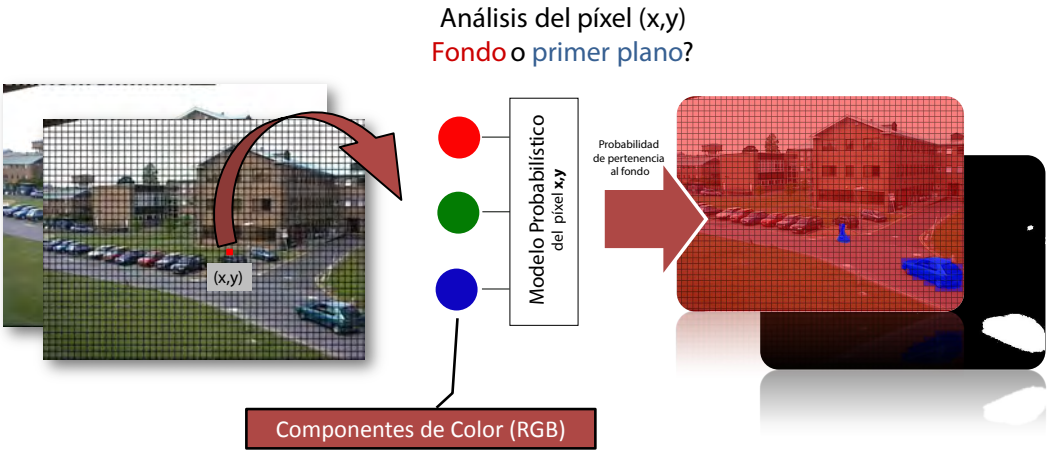


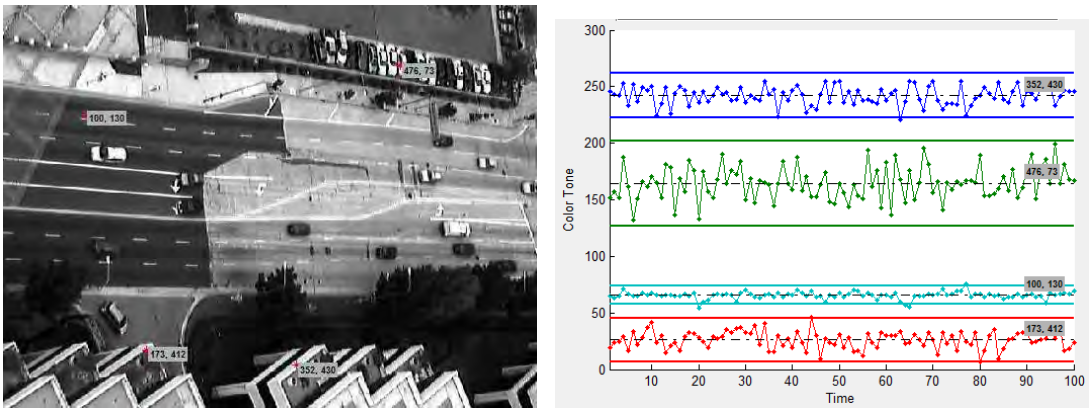
FIGURA 3: Esquema general del modelado de la escena mediante distribuciones probabilísticas a nivel de píxel. La salida del modelo determina la probabilidad de pertenencia al fondo o al primer plano.

sombras, cambios de iluminación tanto graduales como repentinos, o la existencia de objetos del fondo que no permanecen inmóviles durante toda la secuencia, entre otros ([5]).

La entrada que requerirán estas técnicas será una **secuencia de imágenes** obtenida directamente de una cámara de vídeo que está capturando continuamente imágenes del entorno. Nótese que tanto para modelos con cámaras estacionarias o cámaras en movimiento la entrada será la misma: el fotograma actual capturado por el sensor multidimensional.

La propuesta que presentamos en esta sección se enmarca dentro de los algoritmos basados a nivel de píxel donde, en este caso, cada píxel de la escena está modelado por una distribución probabilística. En la Figura 3 puede observarse una representación de cómo actúa el sistema para el análisis de cada fotograma.

Es importante reseñar que la utilización de un modelo por cada píxel se debe a que la variabilidad de la señal de color en cada posición de la escena difiere significativamente incluso entre píxeles anexos. Teóricamente, si un píxel es de fondo, su tonalidad no debería variar de un fotograma a otro. Sin embargo, debido a, principalmente, la compresión utilizada para almacenar la secuencia, la variabilidad es totalmente diferente entre píxeles, como puede obsevarse en la Figura 4. En ella, se muestra un análisis de la variabilidad de algunos píxeles señalados en la Figura 4(a) (sobre imágenes monocromas para mayor comprensión), observándose en la imagen de la derecha (Figura 4(b)) como la varianza o rango de la señal de algunos píxeles es bastante mayor que en otros. Nótese que estos píxeles en el intervalo analizado siempre pertenecen al fondo, con lo que la señal debería ser una línea recta sobre la tonalidad media. Estos valores altos de variabilidad en la señal de color se producen especialmente en los píxeles de bordes por motivos de compresión.



(a) (b)

FIGURA 4: Variabilidad de la tonalidad en escala de grises para diferentes píxeles de la escena. En (a) se han seleccionado varios píxeles para analizar su tonalidad a lo largo del tiempo. La Figura (b) muestra esta variabilidad de color para los píxeles seleccionados.

La técnica propuesta se aleja de los algoritmos clásicos en los que la actualización del modelo de fondo está basada en el algoritmo *EM* de maximización de la esperanza (*expectation-maximization*). De esta forma, se propone el uso de los métodos de aproximación estocástica [6–13], sustentados en el algoritmo de Robbins-Monro [14]. Estas técnicas han sido utilizadas para reemplazar o modificar las típicas estrategias *EM* en una amplia gama de problemas [15–19], además de que han sido convenientemente aplicadas en el ámbito del procesamiento de la imagen y del vídeo [9, 20–23]. Una ventaja clave en el uso de la aproximación estocástica es que está concebida para procesar patrones de entrada de forma individual. Por otro lado, *EM* suele asociarse con el procesamiento por lotes de un conjunto completo de datos de entrada. En segmentación de vídeo, el hecho de que la naturaleza del procesamiento de datos mediante aproximación estocástica sea individual o en línea es de crucial importancia, ya que los datos se generan en tiempo real. Además, esta propuesta tiene una tendencia intrínseca a asignar más importancia a los datos más recientes con respecto a los más antiguos, lo que refuerza su idoneidad para esta tarea. Como veremos, también es una estrategia sólida que necesita un número mínimo de parámetros a ajustar.

2.1. Definición del modelo

Utilizaremos la aproximación estocástica para entrenar una mixtura que modele la distribución de los valores de los píxeles $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}), t_2(\mathbf{x}), t_3(\mathbf{x}))$ en la posición $\mathbf{x} = (x_1, x_2)$. Contaremos con una componente gaussiana para el fondo y una componente uniforme para el primer plano. Esto nos lleva al siguiente modelo probabilístico para la distribución de los valores de un píxel en cualquier posición dada, donde, sin pérdida de generalidad, esta posición viene definida por $\mathbf{x}$:

$$p(\mathbf{t}) = \pi_{Back} p(\mathbf{t}|Back) + \pi_{Fore} p(\mathbf{t}|Fore) =$$

$$\pi_{Back} G\left(\mathbf{t}|\boldsymbol{\mu}_{Back}, \mathbf{C}_{Back} + \boldsymbol{\Psi}\right) + \pi_{Fore} U\left(\mathbf{t}\right) \tag{1}$$

Tenemos que:

$$G\left(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = (2\pi)^{-D/2} \det\left(\boldsymbol{\Sigma}\right)^{-1/2} \exp\left(\left(\mathbf{t} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1} \left(\mathbf{t} - \boldsymbol{\mu}\right)\right) \tag{2}$$

$$U\left(\mathbf{t}\right) = \begin{cases} 1/Vol\left(H\right) & \text{iff } \mathbf{t} \in H \\ 0 & \text{iff } \mathbf{t} \notin H \end{cases} \tag{3}$$

$$\boldsymbol{\mu}_{Back} = E\left[\mathbf{t}|Back\right] \tag{4}$$

$$\boldsymbol{\mu}_{Fore} = E\left[\mathbf{t}|Fore\right] \tag{5}$$

$$\mathbf{C}_{Back} = E\left[\left(\mathbf{t}-\boldsymbol{\mu}_{Back}\right)\left(\mathbf{t}-\boldsymbol{\mu}_{Back}\right)^T |Back\right] \tag{6}$$

$$R_{ni} = P\left(i|\mathbf{t}_n\right) = \frac{\pi_i p\left(\mathbf{t}_n|i\right)}{\pi_{Back} p\left(\mathbf{t}_n|Back\right) + \pi_{Fore} p\left(\mathbf{t}_n|Fore\right)} \tag{7}$$

donde $i \in \{Back, Fore\}$, H es el soporte para la función de densidad uniforme, $Vol\left(H\right)$ es el volumen D-dimensional de H, y $\boldsymbol{\Psi}$ es una matriz diagonal constante que representa el ruido de cuantificación debido a la compresión con pérdida. La diagonal de esta matriz puede estimarse en función del grado de compresión que presenten los fotogramas de la escena analizada. Esta compresión dependerá del algoritmo de captura de la imagen que la cámara tenga incorporada. En esta propuesta consideraremos un valor predeterminado de 5 como valores de la diagonal de esta matriz $\boldsymbol{\Psi}$. En el modelado de fondo, típicamente asignaremos D=3 para píxeles con valores triéstimulo, donde H se extenderá por todo el rango de color:

$$H = \left\{\left(t_1, t_2, t_3\right) | t_1, t_2, t_3 \in [0, v]\right\} \tag{8}$$

con $Vol\left(H\right) = v^3$. En la mayoría de los casos, las imágenes de la secuencia vendrán dadas en el espacio de color RGB, pero $\mathbf{t}$ puede también ser expresado en cualquier otro espacio, tales como Lab, Luv ó HSV. Por otro lado, si tenemos valores de color con una precisión de 8 bits entonces v=255. Hay que destacar que $\boldsymbol{\mu}_{Fore}$ no se utiliza para calcular la densidad de probabilidad en ningún caso. Esta variable sólo es necesaria cuando se detecte un cambio repentino de iluminación, tal y como se expondrá en la posterior Sección 2.3.

La componente gaussiana $G\left(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ es capaz de capturar un amplio rango de valores de fondo que varían dinámicamente, ya que $\mathbf{C}_{Back}$ (y en consecuencia $\mathbf{C}_{Back} + \boldsymbol{\Psi}$) no están limitados a ser matrices diagonales. Por otro lado, la componente uniforme modela la aparición de objetos de primer plano con independencia del color de cada píxel.

2.2. Aprendizaje

Tras la aplicación del modelo probabilístico anterior, el algoritmo de aproximación estocástica de Robbins - Monro define las siguientes ecuaciones de actualización en el tiempo n para el valor observado de un píxel $\mathbf{t}_n$:

$$\forall i \in \{Back, Fore\}\,,\ \pi_i\,(n) = (1-\epsilon_i)\,\pi_i\,(n-1) + \epsilon_i R_{ni} \tag{9}$$

$$\forall i \in \{Back, Fore\}\,,\ \mathbf{m}_i\,(n) = (1-\epsilon_i)\,\mathbf{m}_i\,(n-1) + \epsilon_i R_{ni}\mathbf{t}_n \tag{10}$$

$$\forall i \in \{Back, Fore\}\,,\ \boldsymbol{\mu}_i\,(n) = \frac{\mathbf{m}_i\,(n)}{\pi_i\,(n)} \tag{11}$$

$$\mathbf{M}_{Back}\,(n) = (1-\epsilon)\,\mathbf{M}_{Back}\,(n-1) +$$

$$\epsilon R_{n,Back}\,(\mathbf{t}_n - \boldsymbol{\mu}_{Back}\,(n))\,(\mathbf{t}_n - \boldsymbol{\mu}_{Back}\,(n))^T \tag{12}$$

$$\mathbf{C}_{Back}\,(n) = \frac{\mathbf{M}_{Back}\,(n)}{\pi_{Back}\,(n)} \tag{13}$$

donde ϵ es una tasa de valor constante, $\epsilon \approx 0{,}01$. Esta tasa es constante sin ir decayendo con el tiempo, debido a que el sistema es variable a lo largo del mismo, como se considera en la Sección 3.2 de [6]. Nótese que $\mathbf{m}_i$ y $\mathbf{M}_{Back}$ son variables auxiliares requeridas para actualizar los parámetros del modelo π_i, $\boldsymbol{\mu}_i$ y $\mathbf{C}_{Back}$.

2.3. Detección de cambios repentinos en el fondo

En algunos casos, un objeto requiere integrarse en el fondo, como en el ejemplo de un coche cuando aparca. Esto produce un cambio repentino en las distribuciones de fondo de muchos píxeles, i.e. aquellos que están dentro del objeto a integrar. Estos cambios deben ser tratados separadamente del modelo de fondo desarrollado [24]. Como *Kushner* ha señalado en la Sección 3.2 de [6], la aproximación estocástica en sistemas variantes con el tiempo es sólo aconsejable cuando la tasa de cambio de los parámetros estimados es lenta. De esta forma, estos cambios repentinos deben de ser manejados con algún procedimiento diferente que detecte la situaciones que la propuesta estocástica no es capaz de abarcar.

El método de detección propuesto requiere un estudio *offline* previo. Sea χ la salida real para un píxel particular:

$$\chi = \begin{cases} 1 & \text{si el píxel pertenece al primer plano} \\ 0 & \text{si el píxel pertenece al fondo} \end{cases} \tag{14}$$

Además, sea $\widetilde{\chi}$ la estimación de χ llevada a cabo durante una ejecución *básica* del algoritmo, i.e. sin detectar cambios de fondo repentinos:

$$\widetilde{\chi} = \begin{cases} 1 & \text{si el píxel es clasificado como primer plano por el algoritmo básico} \\ 0 & \text{si el píxel es clasificado como fondo por el algoritmo básico} \end{cases} \tag{15}$$

El algoritmo básico funciona correctamente si y sólo si $\widetilde{\chi} = \chi$. Definimos tres eventos aleatorios disjuntos: cambios en el fondo no detectados (*Change*), detección adecuada del primer plano (*Good*) y otras situaciones en las que no estamos interesados (*Other*):

$$Change \equiv (\widetilde{\chi} = 1) \wedge (\chi = 0) \tag{16}$$

$$Good \equiv (\widetilde{\chi} = 1) \wedge (\chi = 1) \tag{17}$$

$$Other \equiv \widetilde{\chi} = 0 \tag{18}$$

En el fotograma n, el número de fotogramas $z(n)$ en los que el píxel ha sido continuamente clasificado como primer plano es definido como sigue:

$$z(n) = \text{máx}\{N \mid \widetilde{\chi}(n) = \widetilde{\chi}(n-1) = ... = \widetilde{\chi}(n-N) = 1\} \tag{19}$$

Nótese que $z(n)$ es fácilmente calculado mediante un contador que se incrementa en cada fotograma que cumpla $\widetilde{\chi} = 1$ y se reinicializa a cero cada vez que $\widetilde{\chi} = 0$. La tarea *offline* consiste en estimar las probabilidades $P(Good \mid z, \widetilde{\chi} = 1)$ y $P(Change \mid z, \widetilde{\chi} = 1)$. Esto se logra contando el número de veces que *Change* y *Good* se verifican para un cierto valor de z, donde tomaremos como entrada los valores de $z(n)$ para todos los fotogramas n y todos los píxeles tales que $\widetilde{\chi} = 1$. Estos recuentos por cada píxel son funciones de ruido de z que se suavizan mediante la regresión del núcleo, antes de la estimación de las probabilidades.

Sea el umbral Z definido como sigue:

$$Z = \text{mín}\left\{z \mid P(Change|z, \widetilde{\chi} = 1) > \frac{1}{2}\right\} \tag{20}$$

El cálculo de Z finaliza el estudio *offline*. Cuando el sistema de detección está en operación, en cada píxel se definirá un contador del número de fotogramas z. Si en el fotograma n un píxel es clasificado como primer plano y $z(n) \geq Z$, entonces el píxel debe de reinicializarse. Este procedimiento de reinicialización involucra asignar $\mathbf{C}_{Back} = \mathbf{0}$. Además, intercambiaremos $\boldsymbol{\mu}_{Back}$ con $\boldsymbol{\mu}_{Fore}$, y $\mathbf{m}_{Back}$ con $\mathbf{m}_{Fore}$.

2.4. Inicialización

En este punto se esbozará el procedimiento de inicialización que será llevado a cabo al comienzo de la ejecución del método. Los primeros K fotogramas de la secuencia analizada serán utilizados para este propósito, utilizando un valor de $K = 200$ cuando analizamos escenas con cámaras estacionarias. Posteriormente, veremos que este valor variará en el caso de las cámaras en movimiento. Asi, el modelo de fondo por cada píxel se ajustará en función de los K patrones del píxel obtenidos en estos primeros fotogramas. Tendremos que:

$$\forall i \in \{Back, Fore\}\,,\ \pi_i\,(0) = \frac{1}{2} \tag{21}$$

$$\mathbf{m}_{Back}\,(0) = \frac{\pi_{Back}\,(0)}{K} \sum_{j=1}^{K} \mathbf{t}_j \tag{22}$$

$$\boldsymbol{\mu}_{Back}\,(0) = \frac{\mathbf{m}_{Back}\,(0)}{\pi_{Back}\,(0)} \tag{23}$$

$$\mathbf{M}_{Back}\,(0) = \frac{\pi_{Back}\,(0)}{K} \sum_{j=1}^{K} (\mathbf{t}_j - \boldsymbol{\mu}_{Back}\,(0))\,(\mathbf{t}_j - \boldsymbol{\mu}_{Back}\,(0))^T \tag{24}$$

$$\mathbf{C}_{Back}\,(0) = \frac{\mathbf{M}_{Back}\,(0)}{\pi_{Back}\,(0)} \tag{25}$$

Por otro lado, la inicialización de $\boldsymbol{\mu}_{Fore}$ no es crítica, ya que su valor no es utilizado al menos en los primeros Z fotogramas. De esta forma, comenzaremos con un punto en medio del espacio de color:

$$\boldsymbol{\mu}_{Fore}\,(0) = \begin{pmatrix} v/2 \\ v/2 \\ v/2 \end{pmatrix} \tag{26}$$

$$\mathbf{m}_{Fore}\,(0) = \pi_{Fore}\,(0)\,\boldsymbol{\mu}_{Fore}\,(0) \tag{27}$$

Finalmente, el contador por cada píxel será inicializado con $z\,(0) = 0$.

2.5. Implementación

Debido a que nuestro sistema está diseñado para operar en tiempo real, es de gran importancia reducir el número de cálculos requeridos. En particular, la obtención del determinante y la inversa de $\boldsymbol{\Sigma}$, operaciones necesarias para obtener las responsabilidades R_{ni}, son los cálculos más costosos, debido a que involucran un gran número de instrucciones en punto flotante. A continuación describimos cómo optimizar la implementación de estas costosas operaciones.

En primer lugar, no conviene utilizar rutinas de librerías estándar de cálculo numérico (*SNL*), optimizadas para la computación para altas dimensiones D. Los mejores resultados se obtienen tras derivar las ecuaciones específicas para el caso de $D = 3$, que es el único valor a considerar (no tendremos imágenes o fotogramas de más de tres dimensiones). Adicionalmente, aprovecharemos el hecho de que $\mathbf{\Sigma}$ es simétrica, debido a que se corresponde con una matriz de covarianza.

El cálculo del determinante se determina imponiendo las restricciones de simetría a la conocida regla de *Sarrus*:

$$\det \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} = adf - c^2 d + 2bce - e^2 a - b^2 f \tag{28}$$

Una vez que el determinante es conocido, calcularemos la inversa a partir de los cofactores. De nuevo, tendremos en cuenta la simetría de $\mathbf{\Sigma}$, que implica que $\mathbf{\Sigma}^{-1}$ es también simétrica:

$$\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}^{-1} = \left(\det \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \right)^{-1} \begin{pmatrix} a' & b' & c' \\ b' & d' & e' \\ c' & e' & f' \end{pmatrix} \tag{29}$$

$$a' = df - e^2 \tag{30}$$

$$b' = ce - bf \tag{31}$$

$$c' = be - cd \tag{32}$$

$$d' = af - c^2 \tag{33}$$

$$e' = bc - ae \tag{34}$$

$$f' = ad - b^2 \tag{35}$$

Estas ecuaciones proporcionan un forma rápida de cumplir los requisitos de ejecución en tiempo real.

3. Metodología

En esta sección desarrollaremos todo el proceso requerido para la detección de primer plano sobre secuencias en movimiento. Las dos claves más importantes para ello son:

- **Localizar** el fotograma de entrada en el modelo panorámico probabilístico utilizando técnicas de correspondencia de imágenes.

FIGURA 5: Tamaño del panorama con respecto al tamaño de la imagen. La posición del primer fotograma dentro del panorama también está definida por el usuario.

- Obtener un región reducida de este modelo panorámico para **detectar** el movimiento y **actualizarlo** con la información de cada fotograma.

Cabe destacar que utilizamos un modelo probabilístico en lugar de una media recurrente sin tanta complejidad, para manejar señales de color de píxeles que tenga una variación constate y variada, como pueden ser las hojas de los árboles, olas del mar, fuente de agua o cualquier elemento cuya variación en la tonalidad sea repetitiva a lo largo del tiempo. En las siguientes secciones se detallará en mayor medida la metodología propuesta.

3.1. Construcción del panorama

El análisis de una escena comienza con la construcción del panorama, que englobará a todo el espacio visual observado por la cámara en movimiento. Este **panorama** no es más que un modelo estadístico definido por una matriz de un tamaño mayor que la imagen, donde se almacenan los parámetros de la distribución gaussiana que modela cada píxel, tal y como se ha descrito en el modelo básico de la sección anterior. A diferencia de este modelo donde la inicialización del mismo se realizaba analizando los K primeros fotogramas, en este caso no podremos inicializar todo el panorama desde el inicio, porque el tamaño de la imagen es menor que el tamaño del fotograma (ver Figura 5). Por tanto, la inicialización de cada píxel se realizará en el momento en el que observamos al menos K muestras de un píxel del panorama (ver Sección 3.3).

(a)

(b)

FIGURA 6: Máscaras de segmentación y emparejamiento para dos secuencias de entrada. En (a) solamente seleccionamos una región central de la imagen, mientras que en (b) se observa como no seleccionamos los marcadores que estarán fijos en esa posición durante toda la secuencia.

En este proceso de construcción, se define también el lugar de comienzo dentro del panorama del primer fotograma. Además, para todas las escenas, inicialmente, tendremos la opción de definir dos tipos de **máscaras** sobre la imagen:

- Máscara de **segmentación**. Matriz binaria que indica qué píxeles de la imagen se tendrán en cuenta para la detección de movimiento. Por defecto se considera toda la imagen, pero podemos marcar aquellas regiones sobre las que estemos interesados

en detectar movimiento, como por ejemplo la región central del fotograma (ver Figura 6(a)).

- Máscara de **emparejamiento**. Matriz binaria que indica qué regiones de la imagen se utilizarán en la detección de puntos de interés para la correspondencia de imágenes. Normalmente, esta máscara no englobará a toda la imagen de entrada en el caso de que estemos analizando secuencias con elementos superfluos añadidos encima de la propia secuencia de movimiento. Ejemplos varios pueden observarse en secuencias de videojuegos como la mostrada en la Figura 6(b). En ella, los marcadores de los jugadores, tiempo restante de la partida o elementos varios se mantienen durante toda la secuencia en la misma posición, confundiendo a los métodos de emparejamiento (ver Sección 3.2). Con esta máscara evitamos que se obtengan puntos de interés en regiones no requeridas.

Normalmente, las dos máscaras serán la misma ya que es habitual que lo que no se marque como región para emparejar tampoco se utilice para detectar movimiento.

3.2. Correspondencia

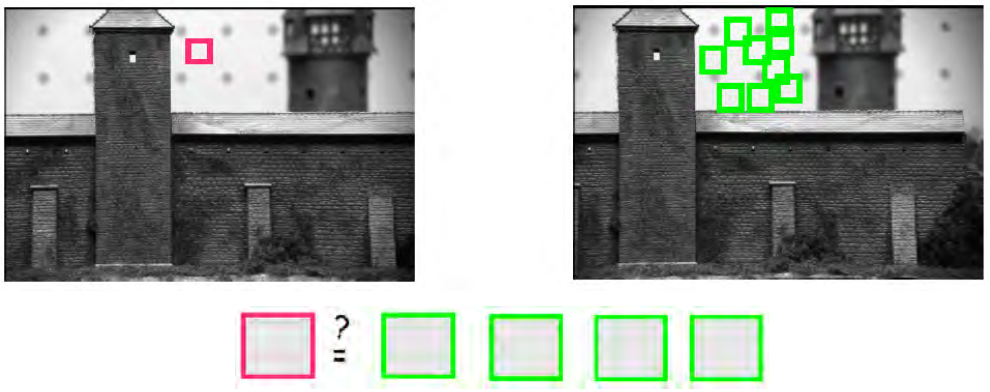
El proceso de correspondencia consiste en la **localización** del fotograma actual que estemos analizando dentro del panorama creado. El objetivo consiste en determinar esta ubicación exacta para obtener un submodelo del modelo estadístico del panorama, con el que se evaluará el fotograma de entrada para detectar los objetos de primer plano en movimiento.

Con la correspondencia de imágenes realmente lo que conseguimos es **corregir** el movimiento de la cámara, para luego aplicar la técnica propuesta en la Sección 2 adaptada a este tipo de escenas. Este proceso se dividirá en diferentes etapas:

- Detección de los **puntos de interés** tanto del fotograma actual como del panorama.
- **Emparejamiento** de las imágenes a partir de los puntos de interés obtenidos.
- **Transformación** de la imagen de entrada a partir de los vectores clave en el emparejamiento.

El resultado final del proceso será la obtención del submodelo de fondo con el que comparar el fotograma. A continuación se detallan en mayor profundidad cada una de las fases anteriores.

(a)



(b)

FIGURA 7: Ejemplo de selección de regiones para la correspondencia de imágenes. La elección de los puntos de interés influye directamente en elección de estas regiones y, en consecuencia, en la calidad del emparejamiento. Aquellos puntos sobre regiones homogéneas no son adecuados para esta tarea, tal y como puede observarse en la imagen (b).

3.2.1. Detección de puntos de interés

Nos referiremos por *punto de interés* a aquellos puntos de la imagen en los que la intensidad de la señal o tono de gris del píxel varía a lo largo de las dos dimensiones. Las *esquinas* convencionales como L-esquinas, uniones en forma de **T** ó **Y** satisfacen la condición anterior, aunque también la cumplen cualquier punto negro sobre fondo blanco o cualquier ubicación con una textura que sea significativa.

Esta elección de puntos de interés tiene mucho que ver con el problema del emparejamiento o correspondencia de dos imágenes, ya que este tipo puntos hace que las regiones que los incluyan sean **distinguibles** entre dos fotogramas o imágenes consecutivas. En la Figura 7 se observa que la selección de regiones significativas, que incluyen estos puntos de interés, mejora la correspondencia de imágenes.

La repetitividad de la obtención de los puntos en una imagen es crítica. Si los algoritmos de extracción de puntos o píxeles de interés no son capaces de obtener los mismos puntos *físicos* para dos imágenes de la misma escena que tengan cierta variación, la alineación

de las imágenes se verá comprometida. De hecho, la variación de la perspectiva de varias imágenes de la misma escena, hace que la coincidencia en los puntos extraídos se vea reducida. Por tanto, mientras mayor sea el número de puntos extraídos y distinguibles, más opciones tendremos de realizar la correspondencia en caso de dificultades. Propiedades como invarianza a escala, orientación, perspectiva e iluminación son consideradas idóneas para este tipo de algoritmos. El hecho de trabajar con secuencias de vídeo que incluirán cientos o miles de fotogramas sobre los que aplicar la correspondencia de imágenes, hace que la eficiencia y el coste computacional de estas técnicas también sea tenida en cuenta a la hora de la elección entre los diferentes algoritmos existentes.

Existen una gran cantidad de detectores de puntos de interés en la bibliografía, divididos en tres categorías [25]: basados en el contorno, en la intensidad o tono de gris y en modelos paramétricos. En esta sección vamos a centrarnos en los métodos basados en la intensidad de la señal, debido a que son los más utilizados desde un punto de vista práctico.

El detector de Harris [26] ha sido una referencia durante los últimos años, siendo aplicado en un gran número de modelos de visión artificial. Está basado en la función de auto correlación de la señal, midiendo las diferencias de tonalidad entre una ventana y la misma ventana desplazada en diferentes direcciones:

$$f(x,y) = \sum_{W} W(x_k, y_k) \left[I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y) \right]^2 \tag{36}$$

donde (x_k, y_k) son los píxeles dentro de la ventana gaussiana W (valores reales entre cero y uno) centrada en (x, y), I es la imagen en tonos de gris y $(\Delta x, \Delta y)$ indican el desplazamiento de la ventana.

Para regiones prácticamente planas u homogéneas tendremos valores cercanos a cero en la Ecuación (36), mientras que aquellas regiones muy texturizadas o con variaciones significativas en su tonalidad tendrás valores más altos.

La utilización práctica de esta función implica integrar sobre todas las direcciones de movimiento de la ventana posibles. Para ello, utilizaremos la matriz de auto correlación que se deriva de la aproximación de la primera derivada de la imagen utilizando las series de Taylor:

$$\begin{aligned} &\textstyle\sum_W \quad [I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y)]^2 \approx \\ &\textstyle\sum_W \quad [I(x_k, y_k) + \Delta x I_x(x_k, y_k) + \Delta y I_y(x_k, y_k) - I(x_k, y_k)]^2 \end{aligned} \tag{37}$$

En forma matricial, la Ecuación (36) combinada con la aproximación de la ecuación anterior quedará:

$$f(x,y) = \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M(x,y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{38}$$

Esta matriz M, que realmente representa la matriz *Hessiana*, captura la estructura de la vecindad local del píxel (x, y). Es interesante reseñar que, a nivel práctico, el cálculo de la derivada de la imagen en base a cada eje (I_x e I_y) se realiza aplicando un filtro de Sobel horizontal (función convolución con la máscara [1 2 1; 0 0 0; -1 -2 -1]) y un filtro de Sobel vertical (convolución con la máscara [1 0 -1; 2 0 -2; 1 0 -1]) sobre la imagen de tonos de gris.

Finalmente, el índice que determina qué píxeles se consideran puntos de interés, es decir, qué píxeles de la imagen son esquinas, se obtiene mediante:

$$R = det(M) - k(trace(M))^2 \tag{39}$$

donde k es constante y empíricamente asignada al rango $k = 0{,}04 \ldots 0{,}06$. Tanto el determinante como la traza pueden ser determinados mediante los autovalores λ_1 y λ_2 de la matriz M:

$$\begin{aligned} det(M) &= \lambda_1 \lambda_2 \\ trace(M) &= \lambda_1 + \lambda_2 \end{aligned} \tag{40}$$

Los autovalores de la matriz $M(x, y)$ nos dan información sobre el tipo de vecindad del punto (x, y) analizado. Así, si λ_1 y λ_2 son valores pequeños la región es plana u homogénea, mientras que si uno de los autovalores es grande y el otro pequeño, diremos que la región está centrada en un borde. En el caso de que ambos autovalores proporcionen valores elevados, indicaremos que la región es una esquina, siendo éste el tipo de regiones más interesante a detectar debido a su robustez en la correspondencia desde diferentes perspectivas. La Figura 8 resume visualmente los diferentes tipos de regiones en función de λ_1 y λ_2.

La Figura 9 muestra un ejemplo de la aplicación del detector de Harris sobre una imagen.

Este tipo de detectores de puntos de interés son invariantes a la rotación, lo que significa que, incluso si la imagen se gira, podemos encontrar las mismas esquinas. Es obvio porque las esquinas permanecen siendo esquinas aunque la imagen esté girada. Sin embargo, esta invarianza no se mantiene si hablamos de la propiedad del escalado. Una esquina

[1] Imagen obtenida de http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

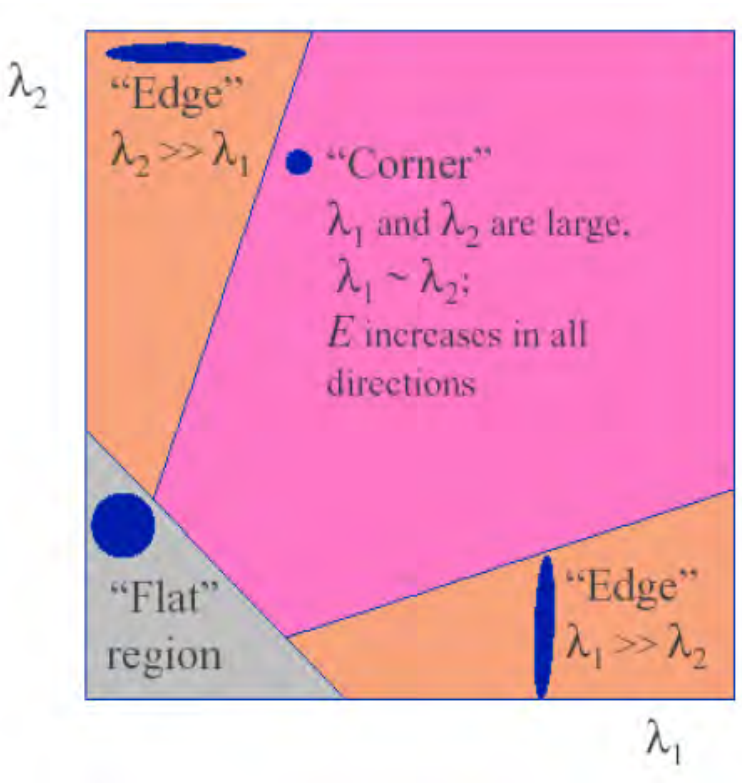


FIGURA 8: Clasificación de los píxeles de la imagen en función de los autovalores de la matriz M[1].

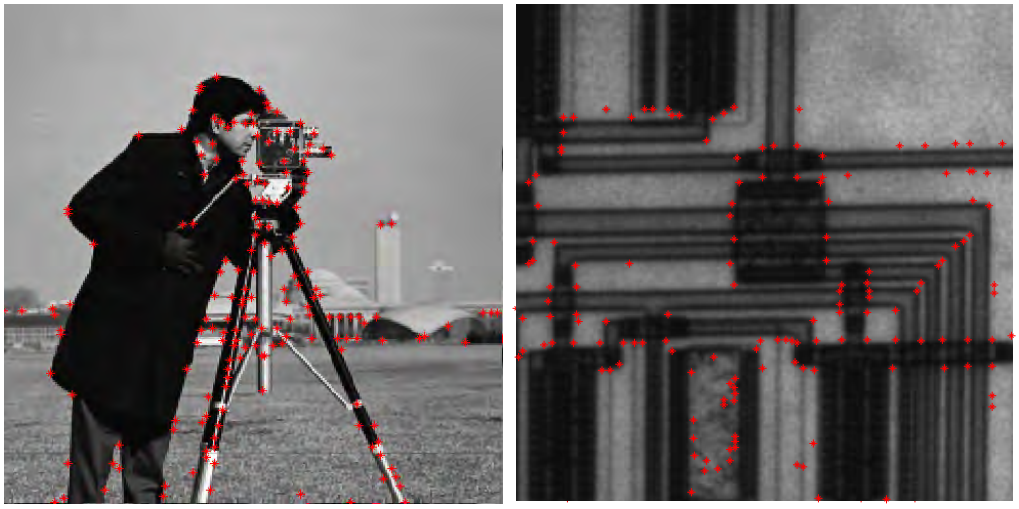


FIGURA 9: Puntos de interés sobre las imágenes predefinidas de Matlab *cameraman* (izquierda) y *circuit* (derecha) obtenidas mediante el detector de Harris.

puede no ser esquina si la imagen se escala, tal y como se observa en la Figura 10. Así, lo que sería una esquina en una pequeña imagen dentro de una pequeña ventana, se convierte en un borde o región plana cuando se realiza una ampliación o *zoom* en la misma ventana. Por tanto, el detector de Harris no es invariante a escala, con lo que si tratamos de hacer correspondencia en secuencias donde se realice un *zoom* de una zona concreta (e.g. en secuencias de fútbol donde la cámara amplia o aleja algunas zonas del

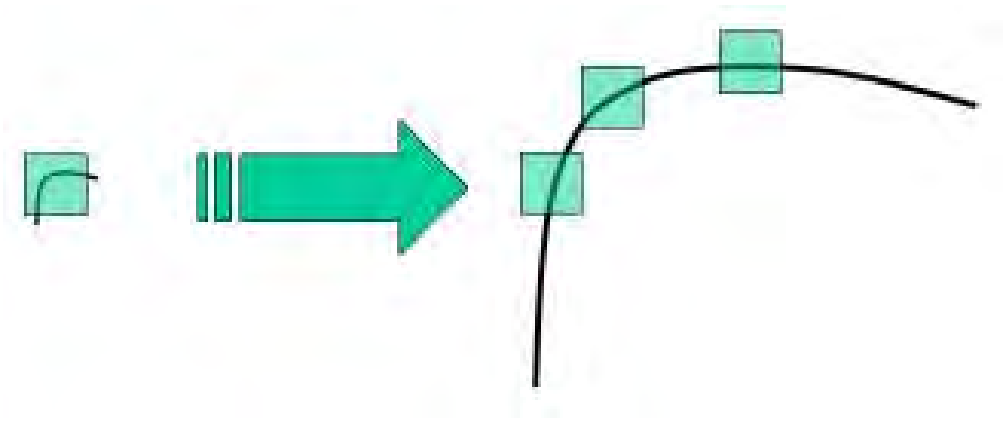


FIGURA 10: Ejemplo de la problemática del escalado en el detector de Harris[2]. La ampliación de una imagen hace que lo que eran esquinas originariamente, en la imagen escalada ya no lo sean.

campo durante el partido), el emparejamiento entre fotogramas consecutivos podría no realizarse correctamente.

El detector de puntos de interés SIFT (*Scale Invariant Feature Transform*) propuesto por Lowe et al. [27] trata de corregir la debilidad anterior utilizando imágenes con diferentes escalas [28]. Para mejorar la robustez de la detección de esquinas sobre imágenes de diferentes tamaños, y con el objetivo de que los puntos obtenidos se refieran a los mismos puntos físicos, esta técnica utiliza la metodología *scale-space* [29]. En ella se aplica un filtro gaussiano lineal sobre las dos dimensiones de la imagen para generar una serie de imágenes *derivadas*:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{41}$$

donde $I(x, y)$ es la imagen de entrada, $L(x, y, \sigma)$ es el conjunto de imágenes de salida en función de σ, $*$ es el operador de convolución y $G(x, y, \sigma)$ es la función de densidad gaussiana definida en dos dimensiones como:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{42}$$

El cálculo de todo el espacio $L(x, y, \sigma)$ implica la construcción de una matriz de imágenes (a la que refieren como *pirámide gaussiana*) donde en uno de los ejes tendremos imágenes del espacio L con el mismo tamaño y que difieren en el valor gaussiano σ aplicado a cada una de ellas (en intervalos múltiplos de k). A este conjunto se le denominará *octava*. El segundo eje de la matriz representa al conjunto de imágenes denominado *escala* y que engloba aquellas imágenes con el mismo parámetro σ pero con diferente tamaño. En la Figura 11 puede observarse un ejemplo de está pirámide de imágenes.

[2]Imagen obtenida de http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

FIGURA 11: Ejemplo de pirámide de imágenes utilizada para el cómputo de los *keypoints* utilizando el método SIFT [30].

Para la detección inicial de puntos de interés estables utilizaremos la diferencia de gaussianas, que puede obtenerse aplicando una resta sobre el espacio L utilizando valores de σ cercanos:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{43}$$

Esta operación es utilizada porque se aproxima en gran medida al operador *Laplaciano* de la Gaussiana, computacionalmente más costoso, pero cuyos máximos y mínimos proporcionan las características más estables comparadas con otras funciones similares (gradiente, Harris, etc) [31]. Todos los puntos seleccionados (máximos y mínimos) se compararán con sus ocho vecinos y con los nueve vecinos de la escala anterior y posterior, marcándose como puntos de interés o *keypoint* si dicho punto es mayor o menor que los 26 píxeles comparados.

Posteriormente, se realizará un filtrado eliminando aquellos puntos con bajo contraste o que estén situados a lo largo de los bordes, debido a que pueden considerarse relativamente sensibles al ruido. Para este último caso, recurrimos al cálculo de la matriz *Hessiana* H sobre la matriz D anteriormente obtenida y, utilizando la traza y el determinante al igual que en el operador de Harris, definimos el siguiente criterio:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{44}$$

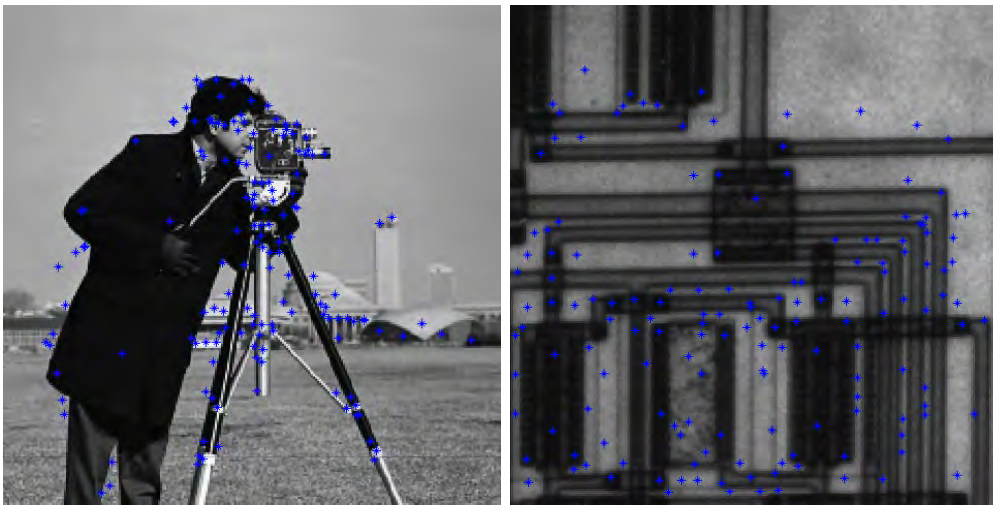


FIGURA 12: Puntos de interés sobre las imágenes predefinidas de Matlab *cameraman* (izquierda) y *circuit* (derecha) obtenidas mediante el método SIFT. Debido a que Matlab en su versión oficial no incorpora este algoritmo de detección de puntos de interés, hemos incorporado una librería (*framework* mexopencv [1]) que permite llamar a los algoritmos desarrollados en OpenCV [2] para su utilización desde Matlab.

donde r define la relación entre los autovalores de la matriz *Hessiana* λ_1 y λ_2:

$$\lambda_1 = r\lambda_2 \tag{45}$$

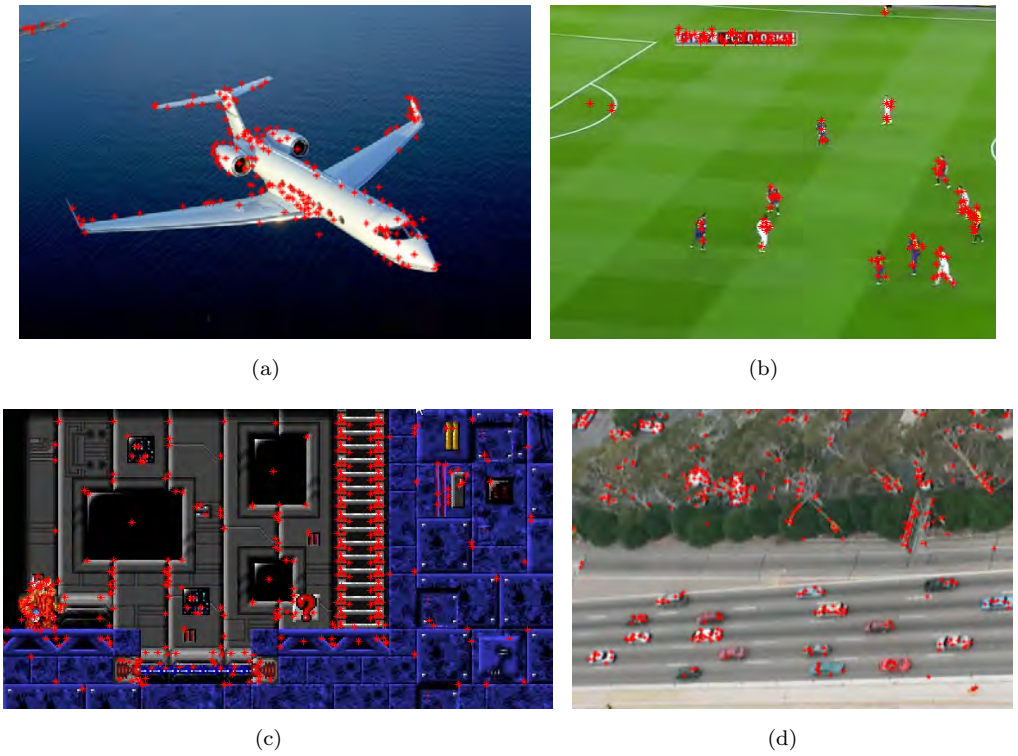
Aquellos *keypoints* que tengan una respuesta reducida en la dirección del borde y muy elevada en dirección perpendicular (están situados a lo largo del borde) tendrán valores altos de r. Estableciendo un valor de $r = 10$, se eliminarán todos aquellos puntos de interés en los bordes pero se mantendrán los localizados en las esquinas.

El siguiente paso consiste en determinar la orientación de cada punto de interés. Para ello, se define una región de 16x16 alrededor de cada píxel y se calcula el módulo y orientación del gradiente para cada punto de esta ventana, con objeto de obtener invarianza a la rotación de la imagen:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{46}$$

$$\theta(x,y) = tan^{-1}((L(x+1,y) - L(x-1,y))/(L(x,y+1) - L(x,y-1))) \tag{47}$$

Esta información relativa a la orientación de cada punto se agrupará en un histograma que represente los 360^o grados posibles de rotación para determinar las direcciones dominantes de las región alrededor del *keypoint* y, en consecuencia, la orientación final del mismo. El paso final del método SIFT consiste en determinar los descriptores por cada punto de interés, para aportar la mayor invarianza posible a cambios de iluminación y cambios de punto de vista. Para más información de esta fase remítase el lector al artículo original [27].



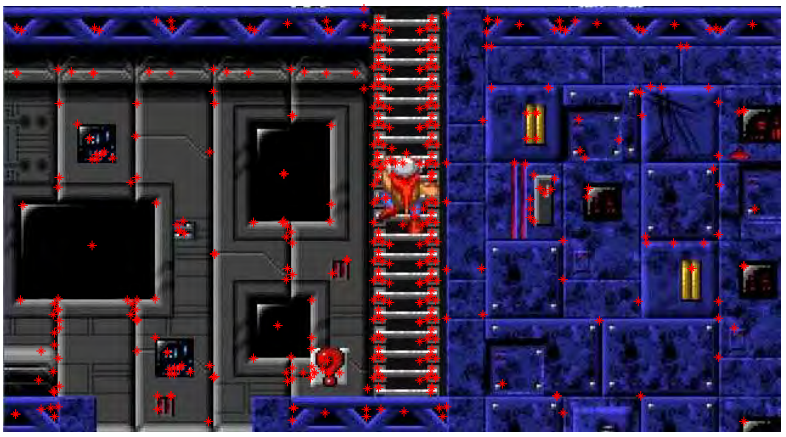
(a) (b)

(c) (d)

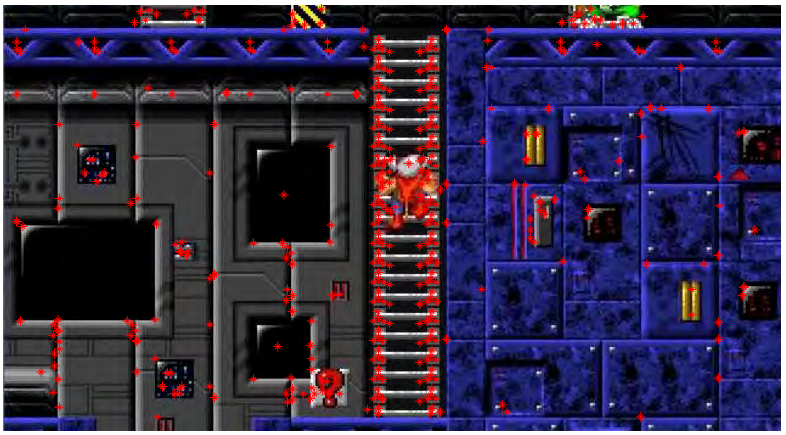
FIGURA 13: Cálculo de puntos de interés con el método SIFT sobre diferentes escenas en movimiento. Se observa que los puntos de interés obtenidos sobre las imágenes de las escenas de la fila superior se centran principalmente en los objetos de primer plano. A diferencia de ello, en las imágenes de las escenas de la fila inferior se detectan *keypoints* tanto del primer plano como del fondo, con lo que con estos últimos se realizará la correspondencia.

Una vez que se han descrito y estudiado dos de los métodos más utilizados para la detección de esquinas y puntos de interés, cabe indicar que la elección de uno u otro método no es tan importante como el tipo de escena a analizar. Esto es debido a que los puntos de interés **válidos** para nuestro problema son aquéllos que se corresponden con elementos del fondo de la escena y no con objetos de primer plano que están en movimiento. Si el fondo de la escena es prácticamente homogéneo y no presenta esquinas o puntos de interés característicos y replicables sobre los que hacer la posterior correspondencia de imágenes, la construcción del panorama pierde robustez y fiabilidad, haciendo que el modelo de fondo se corrompa y en consecuencia la detección de movimiento no se realice correctamente. En la Figura 13 se observa una comparativa de varias escenas con diferentes fondos. Así, en la Figura 13(a) sólo se detectan puntos del avión (objeto de primer plano) y prácticamente ningún punto del océano. Igualmente ocurre en la escena de fútbol representada en la Figura 13(b), donde las líneas blancas del campo no reciben ningún punto de interés que pueda utilizarse posteriormente en el emparejamiento. En las Figuras 13(c) y 13(c), debido a la heterogeneidad del fondo de la escena, sí es posible obtener puntos de los objetos de fondo.

[1]http://vision.is.tohoku.ac.jp/ kyamagu/software/mexopencv/
[2]http://opencv.org/



(a)



(b)

FIGURA 14: Puntos de interés detectados en fotogramas consecutivos. Se observa que a pesar del movimiento, la mayoría de los puntos obtenidos son estables y robustos para indicar la misma localización.

Por tanto, escenas como las mostradas en las Figuras 13(a) y 13(b) deben evitarse si utilizamos la metodología propuesta en este trabajo. No obstante, si se diera este caso, tendríamos que utilizar una transformación de estas imágenes para que el espacio de entrada tenga cierta heterogeneidad, por ejemplo, cambiando de espacio de color o aplicando un filtro que realce los bordes tenues en la escena. También existen otras alternativas que tratan de regionalizar la escena y hacen el emparejamiento en base a la posición de la regiones de fondo obtenidas en diferentes fotogramas [32]. Por otro lado, aunque las secuencias de entrada se representan normalmente mediante el espacio de color *RGB*, podemos utilizar cualquier otro espacio conocido (*Lab*, *HSV*, *YCC*, ...) para detectar puntos de interés y determinar su robustez y viabilidad. Combinaciones entre puntos obtenidos de diferentes espacios podrían mejorar el emparejamiento en secuencias complejas y con pocos puntos de interés en el fondo de la escena.

Para realizar el emparejamiento de puntos de la sección siguiente, es necesario determinar puntos de interés en dos imágenes. Por ejemplo, en la Figura 14 pueden observarse dos

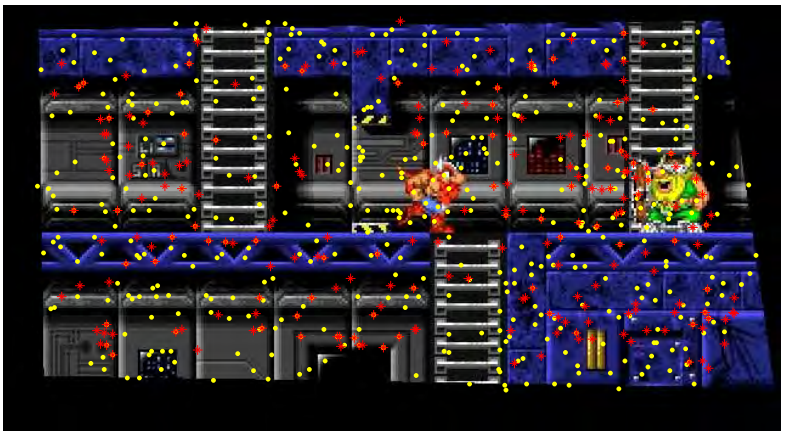
fotogramas consecutivos en los que se produce movimiento de fondo. Los puntos de interés detectados son robustos a este movimiento, es decir, estos puntos se localizan en la misma zona en las dos imágenes pese a existir un movimiento vertical de la cámara.

Además, el primer plano en movimiento (vikingo moviéndose por la escalera) no agrupa la mayoría de los *keypoints* a diferencia de lo ocurrido en la Figura 13(a), con lo que los puntos en común en ambas imágenes son del fondo y mayoritarios, posibilitando un emparejamiento correcto. Sin embargo, esta idea general hay que extrapolarla al modelo que estamos desarrollando, ya que en este caso no tendremos dos fotogramas de la escena para hacer la correspondencia, sino el fotograma actual y el **panorama** que estamos creando. Como el panorama es un modelo probabilístico que contiene información estadística por cada píxel, nos quedaremos con la tonalidad media de cada píxel (μ_{Back}) definida en la Ecuación (4) de la Sección 2.1. En la Figura 15 se muestran las imágenes sobre las que realizar la correspondencia, así como algunos de los puntos de interés detectados. Se observan en rojo los *keypoints* detectados y que han sido válidos para el emparejamiento (ver sección siguiente) y en amarillo el resto de puntos de interés. Nótese que los puntos válidos para realizar el emparejamiento en el panorama (Figura 15(b)) se encuentran situados en la parte superior izquierda, zona en la que se encuentra el fotograma actual analizado (Figura 15(a)). Para ser un poco más eficientes en la ejecución del algoritmo, la detección de *keypoints* sobre el panorama no se realizará en cada fotograma, si no que será cada 5 ó 10 fotogramas, pues la actualización del panorama no será tan rápida como para que tengamos nuevas zonas significativas en ese tiempo. Aún así, este parámetro podría modificarse en caso de escenas con movimientos de cámara más rápidos.

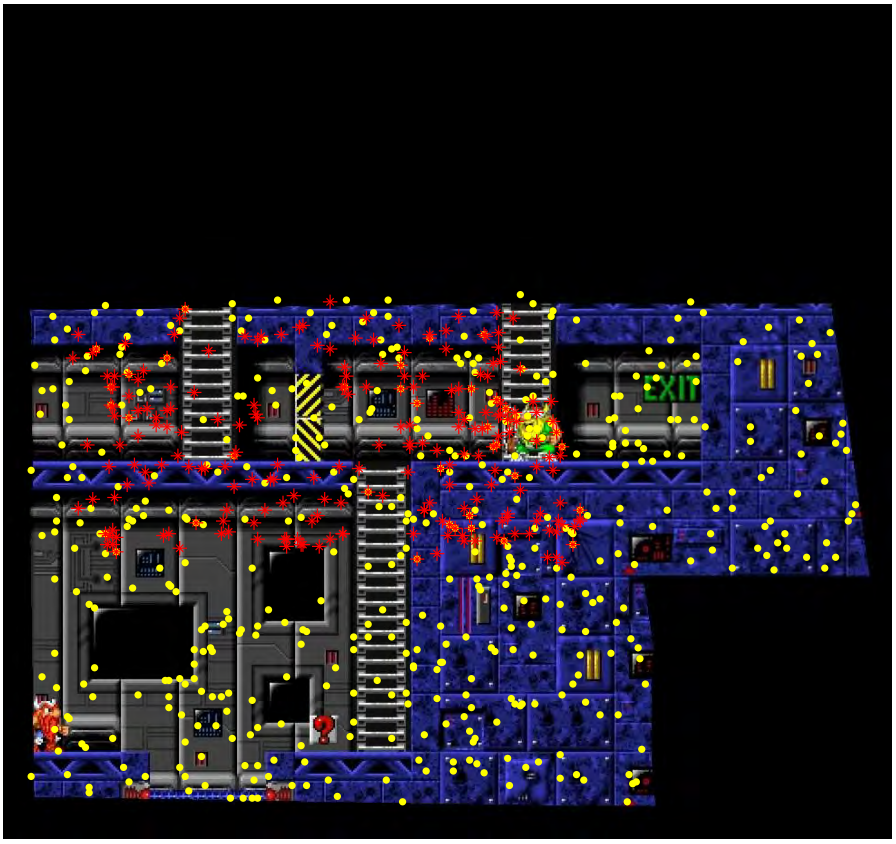
Tras todo lo comentado, podemos concluir que tener imágenes con resolución adecuada, definidas y con calidad aceptable será crucial y crítico en el buen funcionamiento de la detección de puntos y, en consecuencia, del emparejamiento. Si las imágenes no tienen calidad, sea por la propia secuencia o por una incorrecta actualización del panorama, la correspondencia no se realizará correctamente. De esta fase depende en gran medida la detección de movimiento.

3.2.2. Emparejamiento de puntos

Tras la detección de los puntos de interés en el fotograma actual y el panorama será necesario determinar la correspondencia entre los *keypoints* de ambas imágenes. Para ello, utilizaremos el algoritmo de la búsqueda del vecino más cercano, del cuál existen multitud de variantes y que básicamente consiste en, dado un conjunto de puntos S en un espacio M, y un punto de búsqueda $q \in M$, encontrar el punto más cercano en S a q [33].



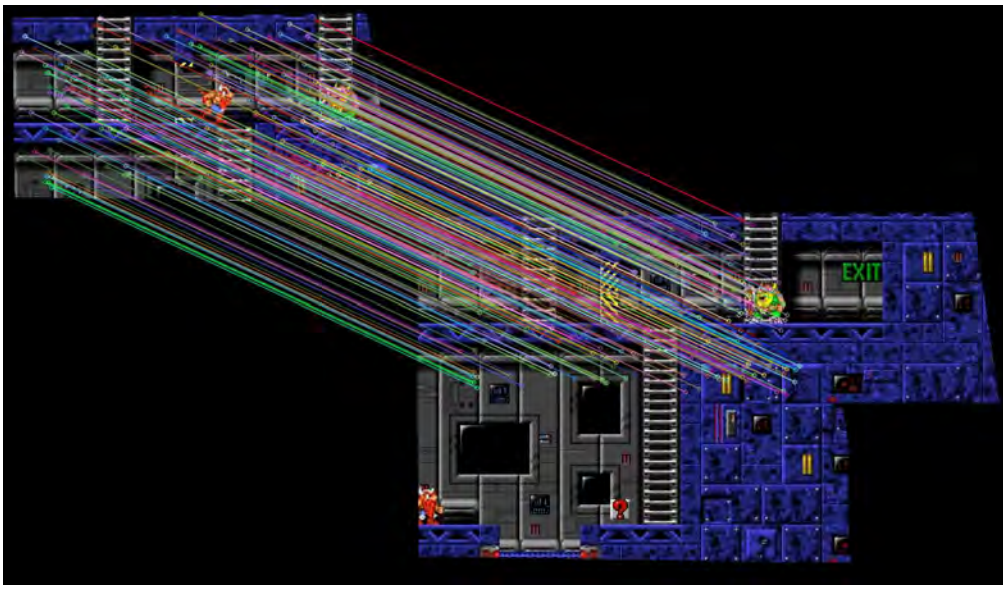
(a)



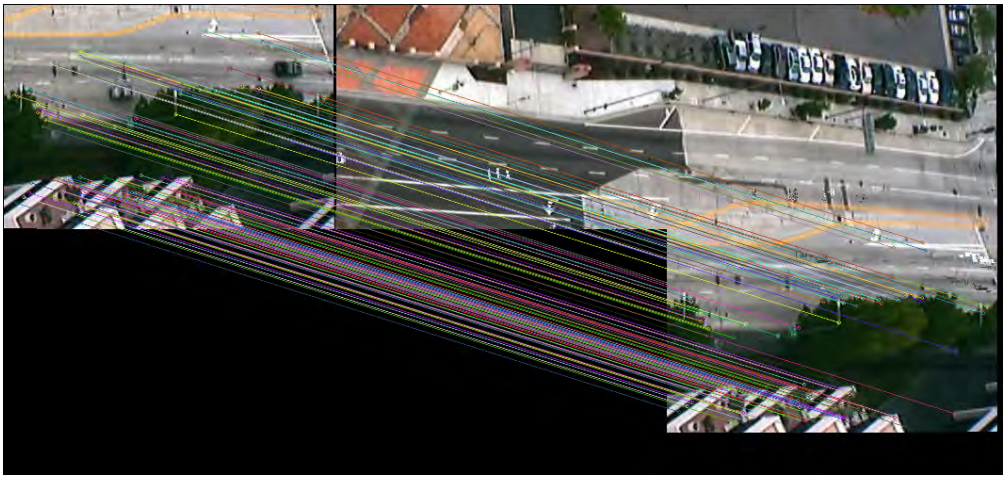
(b)

FIGURA 15: Detección de puntos de interés en una de las fases del algoritmo. En (a) observamos el fotograma actual, en (b) el panorama construido hasta ese momento. Los puntos de color rojo indican que han sido utilizados para el emparejamiento posterior, en amarillo algunos de los puntos de interés restantes.

Este problema de optimización ha sido ampliamente estudiado en la bibliografía de diferentes ámbitos, aplicándose en el caso de la correspondencia de imágenes a la búsqueda de los descriptores más afines (asociados a los *keypoints* y comentados en la sección anterior) entre ambas imágenes de entrada. El método define una medida de similitud donde los *keypoints* más cercanos entre ellos tienen valores cercanos a 0, mientras que



(a)



(b)

FIGURA 16: Emparejamiento de puntos entre el fotograma actual y el panorama para dos escenas analizadas. Las líneas rectas muestran la correspondencia entre los píxeles de ambas imágenes.

puntos más lejanos o menos parecidos en base a los descriptores tendrán valores altos. Nos quedaremos con aquellas correspondencias que denominaremos *válidas*, siempre que la medida de similitud δ_m sea inferior a k veces la mínima medida de similitud obtenida:

$$\delta_m < k \min \{\delta_i, \quad i \in (1 \dots N)\} \tag{48}$$

Debido a que este problema de optimización es *NP* completo (no resoluble en tiempo polinómico) utilizaremos aproximaciones que operan en tiempo reducido y que están englobadas en una librería bastante utilizada denominada FLANN (*Fast Library for Approximate Nearest Neighbors*) [34, 35]. Esta librería está incluida en el *framework*

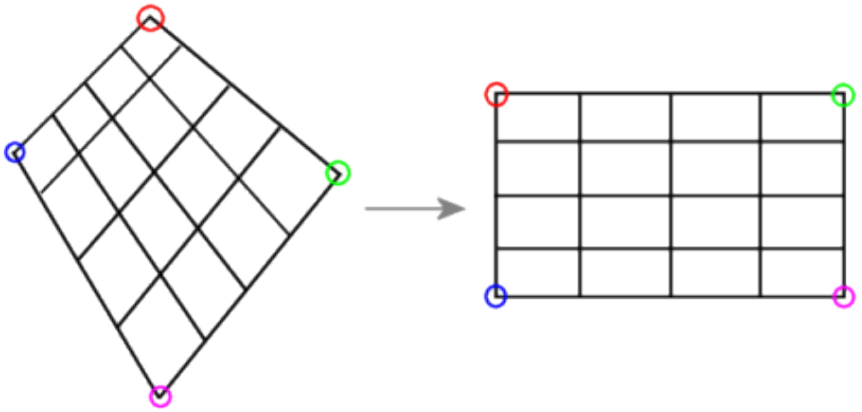


FIGURA 17: Ejemplo de homografía sobre una cuadrícula. Para una región 2D la matriz de homografía puede obtenerse a partir de cuatro puntos.

OpenCV que hemos utilizado junto con Matlab para el desarrollo del prototipo de este trabajo. En la Figura 16 observamos dos ejemplos de aplicación del método de emparejamiento de puntos. Las líneas rectas enlazan aquellos *keypoints* que han sido asociados. Como puede observarse, el emparejamiento no requiere que el panorama esté construido totalmente, si no que utilizará el que tengamos en ese momento.

3.2.3. Transformación de la imagen de entrada

El objetivo de la correspondencia en nuestra propuesta es determinar **dónde** ubicar el fotograma actual analizado dentro del modelo del panorama que estamos desarrollando. Esta necesidad se justifica en que posteriormente tendremos que actualizar los píxeles del panorama con la nueva información obtenida del fotograma actual y detectar el movimiento del primer plano, con lo que la coincidencia entre los puntos de una imagen y los de la otra (panorama) debe de ser lo más exacta posible.

Una vez hemos determinado los píxeles emparejados utilizando la librería FLANN, será necesario encontrar la transformación geométrica de la imagen de entrada más adecuada. Esta transformación proyectiva se denomina **homografía** y trata de corresponder dos figuras geométricas planas definidas por puntos y rectas de emparejamiento (ver Figura 17). A partir de la matriz obtenida para la homografía en coordenadas homogéneas (3x3), podemos obtener la rotación, traslación e incluso transformación perspectiva del fotograma de entrada con respecto a la imagen del panorama [36]. Este tipo de transformaciones se aplican también en el campo de la realidad aumentada, con objeto de añadir información virtual a la escena real observada [37].

Si hacemos previamente un estudio de la escena podríamos determinar el tipo de movimiento que realiza la cámara, restringiendo la matriz de homografía a un tipo concreto.

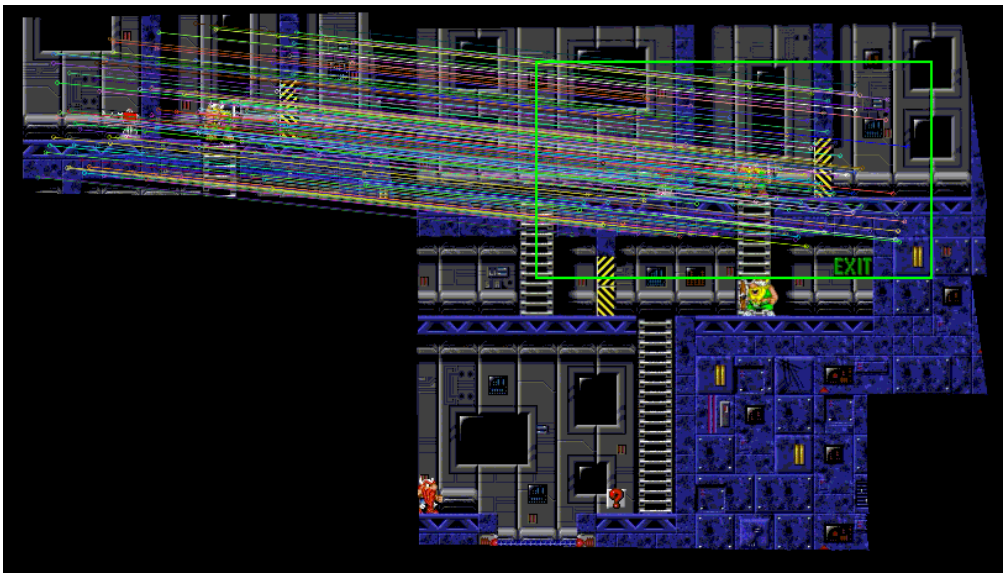
FIGURA 18: Ejemplo del problema de la correspondencia. ¿Cuál es el vector válido para la transformación?

Con ello prevenimos posibles errores en la obtención de esta matriz que pueden provocar que la correspondencia no se haga del todo exacta, y redunde en una incorrecta actualización del panorama.

Normalmente, el movimiento de la cámara suele ser traslacional, consistiendo en el desplazamiento de la misma de un lado para otro. Este tipo de cámaras, utilizadas en videovigilancia para observar amplias vistas de un escenario, se denominan cámaras **PTZ** (*pan-tilt-zoom*). En este trabajo consideraremos como restricción que la cámara sólo pueda realizar movimientos verticales y horizontales, por ser los más frecuentes para el tipo de escenas analizadas. En posteriores modificaciones trataremos de incorporar al modelo el análisis de escenas con *zoom* en la cámara, pero son bastante más complejas porque la correspondencia con el fondo no es exacta, y la actualización del panorama suele desenfocar y *emborronar* el modelo. Por tanto, la matriz de homografía tendrá su diagonal a uno.

Sin embargo, el cálculo de esta matriz no es tan trivial, ya que no todas las correspondencias detectadas son válidas para generarla (existencia de *outlayers*). En el ejemplo de la Figura 18, no todos los vectores de emparejamiento son correctos y lo ideal sería quedarse con aquel vector *medio* de entre todos los que no son *outlayers*. Esto es precisamente lo que realiza el método iterativo **RANSAC** (*Random Sample Consensus*) [38], consistente en la estimación de parámetros a partir de los datos observados en presencia de *outlayers*. En nuestro caso, estos parámetros serán los valores de la matriz de homografía. El funcionamiento en bucle del método es el siguiente:

- Selecciona cuatro pares de puntos de forma aleatoria (mínimo necesario para una transformación 2D).
- Calcular la matriz de homografía H.



(a)

(b)

FIGURA 19: Localización del fotograma de entrada en el panorama utilizando RANSAC. Se observa que en las dos imágenes el fotograma de entrada está a la izquierda.

- Obtener los llamados *inliers*, que son puntos de correspondencia (p_i en la primera imagen, p'_i en la segunda) que cumplen la siguiente ecuación:

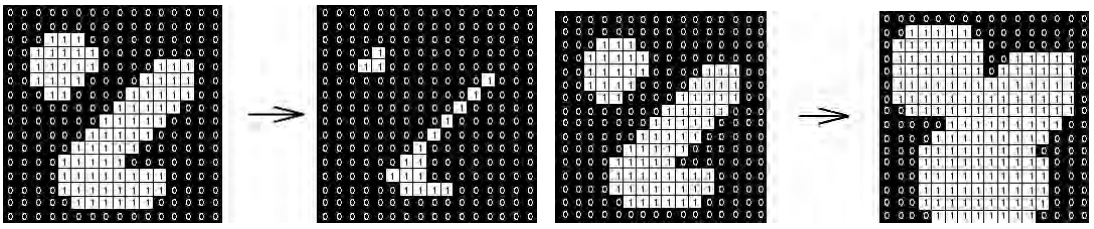
$$\|p'_i - Hp_i\| < \epsilon \tag{49}$$

  donde ϵ es un valor muy pequeño. Serán puntos válidos aquellos p_i de la primera imagen que, multiplicados por la matriz de transformación, coincidan o estén muy próximos a los puntos de interés p'_i de la segunda imagen.

- Almacenamos el conjunto mayor de *inliers*.

- Recalculamos mediante mínimos cuadrados la matriz H utilizando todos los *inliers* anteriores.

Realizaremos el bucle anterior un número N de veces ($N = 100, 1000$) y nos quedaremos con los pares de puntos que generen un número mayor de *inliers* para, finalmente, obtener la matriz de homografía.



(a) (b)

FIGURA 20: Aplicación de operadores morfológicos de erosión y dilatación (plantilla de 3x3) sobre una más binaria. En (a) se observa como se reducen los bordes del objeto, mientras que en (b) se amplían. Combinaciones de estos operadores generan una imagen más limpia para la salida de la máscara de movimiento.

En la Figura 19 mostramos dos ejemplos donde se observa el rectángulo del panorama donde se localiza el fotograma actual analizado. Para dibujar este recuadro es necesario conocer la matriz de homografía exacta para aplicar la transformación.

3.3. Inicialización de regiones no vistas

En los primeros fotogramas de una secuencia nos encontraremos con píxeles del panorama que se observan por primera vez. La inicialización de estos píxeles conlleva tener información de cada uno de ellos al menos durante K fotogramas, para así poder actualizar los parámetros de su distribución gaussiana (media y matriz de covarianzas). Por tanto, dispondremos de un contador por cada píxel del panorama inicializado al valor K. Normalmente este valor será de $K = 10$ para las secuencias con cámaras en movimiento, mientras que es habitual que para secuencias estáticas tenga valores más altos. Cada vez que se observe el píxel i, j del panorama en el fotograma, restaremos uno al contador y almacenaremos en un *buffer*, independiente para cada píxel, la tonalidad RGB (o la tonalidad en el espacio de color en el que se encuentre la secuencia) de ese píxel. Cuando este contador esté a cero para el píxel i, j, utilizaremos la información almacenada en el *buffer* para actualizarlo, tal y como se describe en la Sección 2.4.

Una vez se haya inicializado un píxel, se marcará para que no vuelva a inicializarse, con lo que cada valor nuevo RGB asociado al píxel i, j actualizará su distribución gaussiana en el modelo panorámico. Cabe indicar que un píxel que se observe en el fotograma no se establecerá en el panorama hasta pasados K imágenes, debido a que necesita datos para modelar la señal.

3.4. Obtención del primer plano

La obtención del primer plano consiste en la construcción de una máscara binaria en la que establecemos a uno, aquellos píxeles que están en movimiento, y a cero, aquellos píxeles que pertenecen al fondo de la escena. No se tiene en cuenta el movimiento de

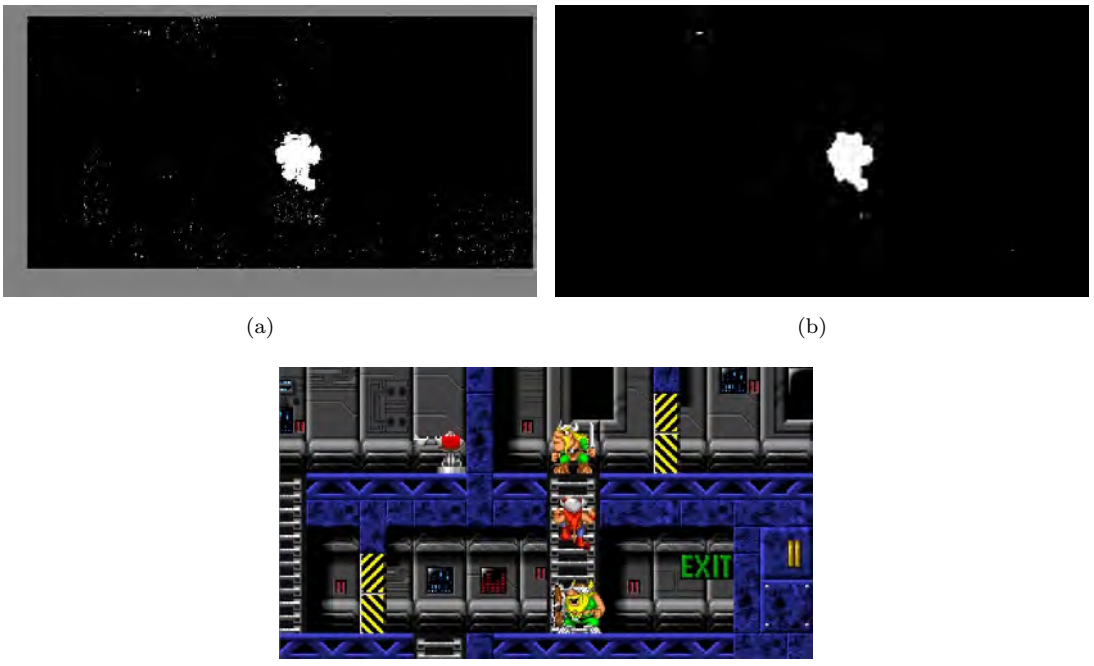
la cámara y sí el de los objetos en relación a ésta. Esta máscara tendrá el tamaño del fotograma de entrada, nunca el tamaño del panorama.

Esta máscara se calculará utilizando el fotograma de entrada y el submodelo probabilístico del panorama obtenido tras realizar el proceso descrito en la Sección 3.2. Para ello, se aplicarán las ecuaciones definidas en la Sección 2.1 para determinar si la tonalidad de entrada para cada píxel pertenece a la distribución gaussiana multidimensional asociada. Si pertenece, diremos que el píxel es de fondo y si no pertenece se asociará al primer plano.

Es habitual que la máscara de primer plano contenga más píxeles espurios aislados que las máscaras obtenidas para secuencias grabadas con cámara fija. Así, utilizaremos **operadores morfológicos**, consistentes en convoluciones de plantillas o *templates* de pequeño tamaño sobre la máscara binaria de detección de objetos final, para limpiar los posibles defectos encontrados en la salida. En este caso, aplicaremos operadores morfológicos de **erosión**, que elimina píxeles espurios, objetos de pequeño tamaño y reduce los bordes de los objetos en función del tamaño, y de **dilatación**, que extiende y amplia los bordes del los objetos según el número de píxeles indicado. En la Figura 20 puede verse un ejemplo del funcionamiento de estos operadores con una plantilla de 3x3 sobre una matriz binaria. Finalmente, en la Figura 21 se muestra una comparativa entre la máscara obtenida sin procesamiento posterior y con la aplicación de los operadores de erosión y dilatación.

Cabe destacar que, al ser una técnica realizada a nivel de píxel, el primer plano detectado es muy sensible a errores de correspondencia de incluso un par de píxeles, pues en este caso ya no coincidirían la posición de los píxeles del fotograma con los del submodelo. En este caso, tendremos que realizar una comprobación posterior para determinar si la máscara es considerada errónea, basándonos en el porcentaje de píxeles de la imagen que sean de primer plano, cuyo límite máximo está definido en el 75 %, sobre el total indicado en la máscara de segmentación inicial. A partir de este valor consideraremos que la máscara de movimiento no es correcta y puede existir algún problema con la correspondencia. También, un desplazamiento de un píxel provoca variaciones de tamaño un píxel en la máscara de segmentación. Si utilizamos la matriz limpia tras aplicar operadores morfológicos y la restamos a la máscara de primer plano detectado, es posible discernir que, si el número de píxeles activos es elevado, podrían existir errores en la detección del movimiento (ver Figura 22).

Tras una detección correcta, procedemos a actualizar los píxeles del submodelo de fondo asociados al fotograma de entrada. Dicha actualización se relata en mayor detalle en la sección siguiente.



(a) (b)

(c)

FIGURA 21: Ejemplo de detección de primer plano. En la Figura (a) se observa la máscara de movimiento, mientras que la misma se ha mejorado en la Figura 21(b) tras aplicar operadores morfológicos. Nótese que los otros dos vikingos que aparecen en la Figura 21(c) no se detectan como movimiento debido a que no han modificado su posición desde el inicio, considerándose, por tanto, parte del fondo.

3.5. Actualización del modelo de fondo

Tras detectar el primer plano, el último paso consiste en actualizar el panorama con los píxeles del fotograma de entrada, con objeto de modificar los parámetros de la distribución gaussiana multivariante de cada píxel. Nótese que el modelo se actualizará siempre que no hayamos detectado un error en la obtención del *foreground*. En caso contrario, el modelo panorámico no se modificará. Esto se debe a que si actualizamos con datos incorrectos o desplazados en el espacio varios píxeles, el modelo tiende a difuminarse y a perder los detalles, haciendo que la detección de puntos de interés sobre el panorama ya no localice las posiciones exactas, y provocando, en definitiva, que el modelo se corrompa.

Cuando hablamos de la actualización del modelo de fondo, realmente nos estamos refiriendo a la actualización del submodelo asociado al fotograma de entrada, con lo que sólo una región del panorama se actualizará. Para ello, se seguirán las directrices indicadas en la Sección 2.2. A diferencia de secuencias con cámaras estáticas, en las analizadas en este trabajo se requerirán valores altos para la tasa de aprendizaje ϵ (0.1, 0.05), pues la utilización de valores pequeños puede provocar el *emborronamiento* del fondo panorámico y que la correspondencia de puntos no se realice correctamente.

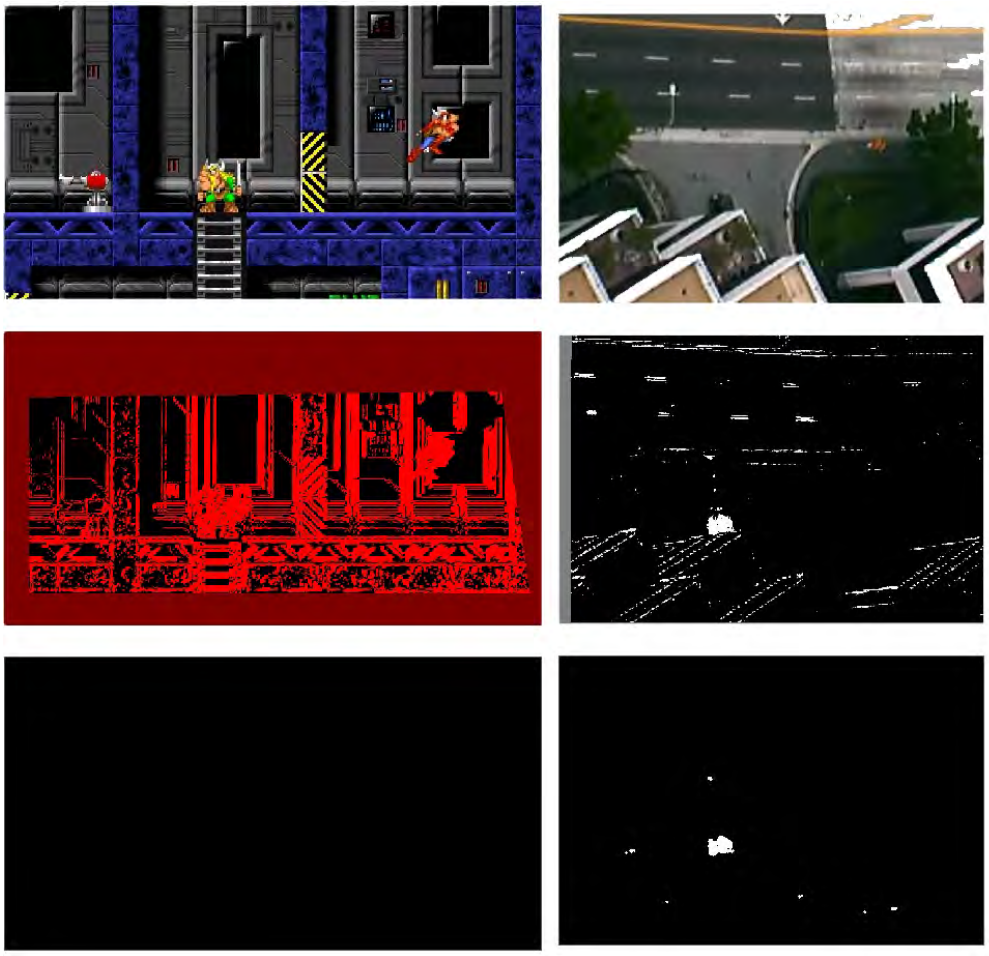


FIGURA 22: Ejemplos de errores obtenidos en el proceso de detección. En el caso de la columna de la izquierda el fotograma se ha considerado inválido debido al gran número de primer plano detectado. En el caso de la derecha, la aplicación de los operadores morfológicos ha mejorado la máscara de salida.

Cabe destacar que la utilización de un modelo probabilístico de fondo no es un capricho, si no que se requiere para poder manejar la **variabilidad** de cada uno de los píxeles del panorama. La utilización de sólo una matriz de valores medios del panorama provocaría un mayor número de puntos espurios en la detección del primer plano, además de los problemas clásicos en el ajuste y umbralización de este tipo de técnicas.

4. Resultados Experimentales

En esta sección presentaremos algunos de los resultados obtenidos tras aplicar la metodología anterior sobre secuencias con cámaras en movimiento. Orientaremos principalmente la aplicación de este trabajo al análisis de secuencias de vídeo obtenidas por cámaras de movimiento PTZ, que cubren un área visual mayor que las cámaras fijas, lo que conlleva a un ahorro en el costo de recursos hardware. Como desafortunadamente no disponemos

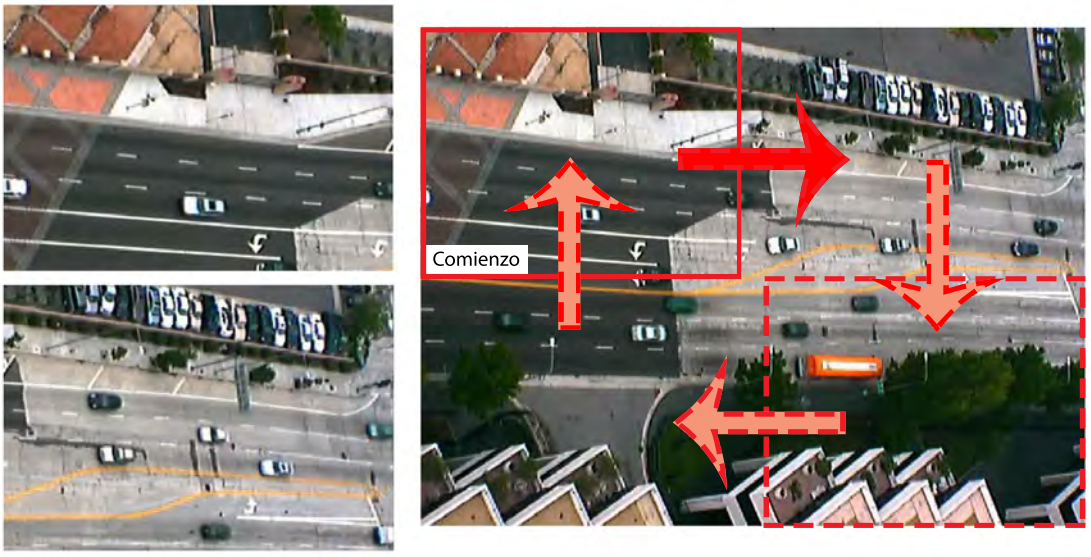


FIGURA 23: Aproximación del movimiento de una cámara PTZ sobre una secuencia fija utilizando un software de edición de vídeo. Se observa como la región ampliada se irá desplazando en bucle haciendo círculos sobre la escena.

de este tipo de cámaras tan especificas, hemos tratado de simularlas utilizando un software de edición de vídeo. Hemos utilizado varias secuencias de tráfico en autopistas y autovías proporcionadas por NGSIM[3], debido a que son escenas susceptibles a colocar cámaras con movimiento horizontal y vertical. Sobre dichas escenas hemos ampliado una región concreta y desplazado de forma progresiva y constante sobre la imagen, de izquierda a derecha y de arriba a abajo, tal y como se observa en la Figura 23.

Tendremos, por tanto, cuatro secuencias de vídeo en las que hemos simulado este movimiento, descritas como:

- *Lankershim*. Secuencia obtenida de NGSIM y que muestra una avenida principal en la que hay un cruce y varios carriles de conducción a izquierda y derecha. Está formada por varios miles de fotogramas con tamaño de 704x480 píxeles. La cámara se mueve en círculos por esta escena.

- *Sb-camera*. Secuencia obtenida también de NGSIM que transcurre en una autovía de una sola dirección hacia la izquierda. Nótese que es adecuada para nuestro sistema debido a que existe vegetación en la parte superior de la escena, con lo que la detección de puntos de interés es robusta y estable. Formada por casi 3000 fotogramas con un tamaño de 704x480 píxeles. El movimiento de la cámara es sólo lineal, de izquierda a derecha y viceversa.

- *AvdaAndalucía*. Secuencia grabada en la Avenida Andalucía de la ciudad de Málaga, ramal bastante concurrido de tráfico a cualquier hora del día. El tamaño de la misma será de 320x240 píxeles con movimiento circular de la cámara.

[3] http://ngsim-community.org/

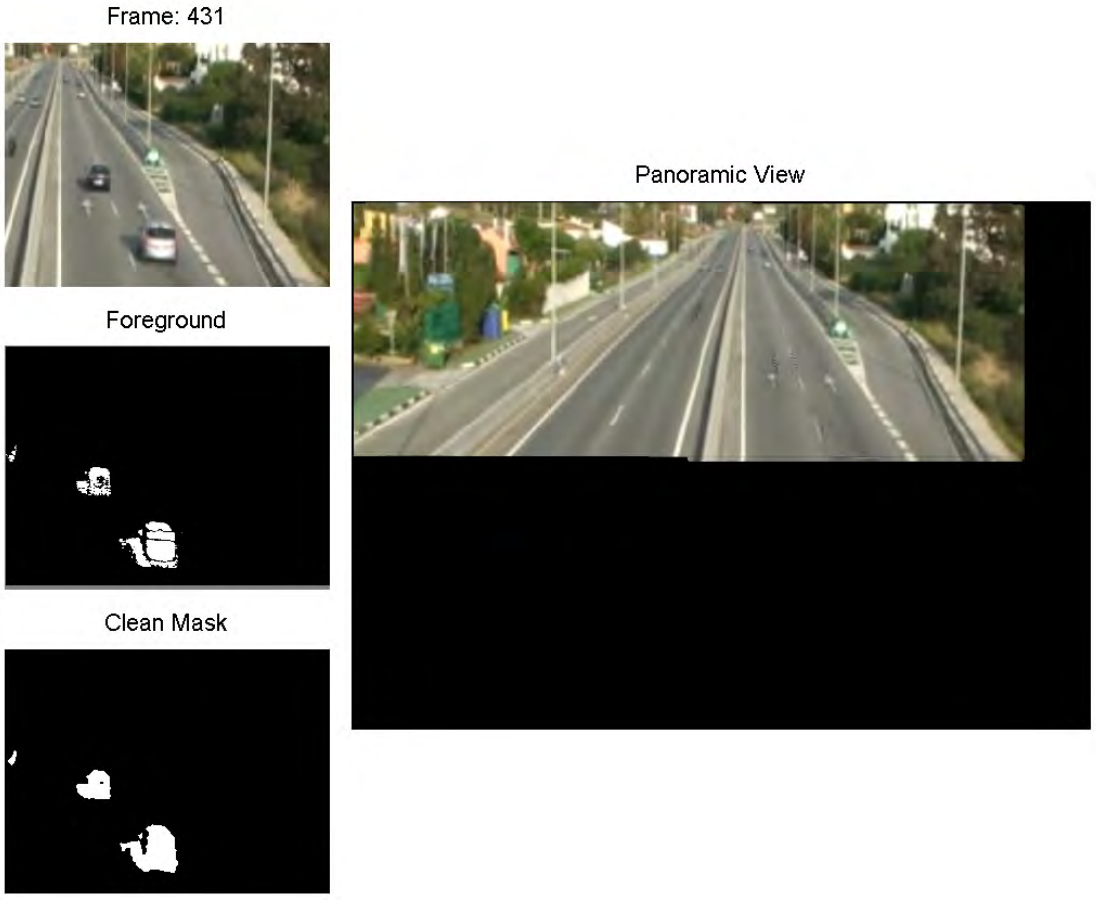


FIGURA 24: Detección de movimiento para la escena *AvdaAndalucia*. Se observa como el panorama está en proceso de construcción mientras se analiza la secuencia.

- *PETS01*. Secuencia obtenida del repositorio público PETS 2001[4] que muestra una escena de un campus universitario. Tiene un tamaño de 768x576 píxeles y, al igual que en la anterior, la cámara realiza un movimiento circular.

Otro de los objetivos consistía en analizar secuencias deportivas de eventos relacionados con el fútbol, baloncesto, voleibol, ... Sin embargo, los resultados obtenidos no han sido lo suficientemente adecuados para utilizarlos en este trabajo. El hecho de que el fondo de estas escenas sea bastante homogéneo (el campo verde en un partido de fútbol, el parqué de un partido de baloncesto o de balonmano, ...), hace que los puntos de interés se detecten principalmente sobre los jugadores, lo que implica que no podamos corregir el movimiento de la cámara con el panorama. Además, la utilización del *zoom* de la cámara en muchos de estos eventos hace que el proceso se complique. Como alternativa, hemos incluido alguna escena de juegos de ordenador (principalmente de juegos de plataformas) en los que tanto la cámara como los personajes están en movimiento. Es necesario que el fondo de estas escenas sea heterogéneo para una correcta correspondencia con el panorama. Se ha seleccionado el juego denominado *vikings*, consistente en la superación de diferentes niveles sobre escenas variopintas, donde el fondo está bastante detallado.

[4] ftp://ftp.pets.rdg.ac.uk/pub/PETS2000/

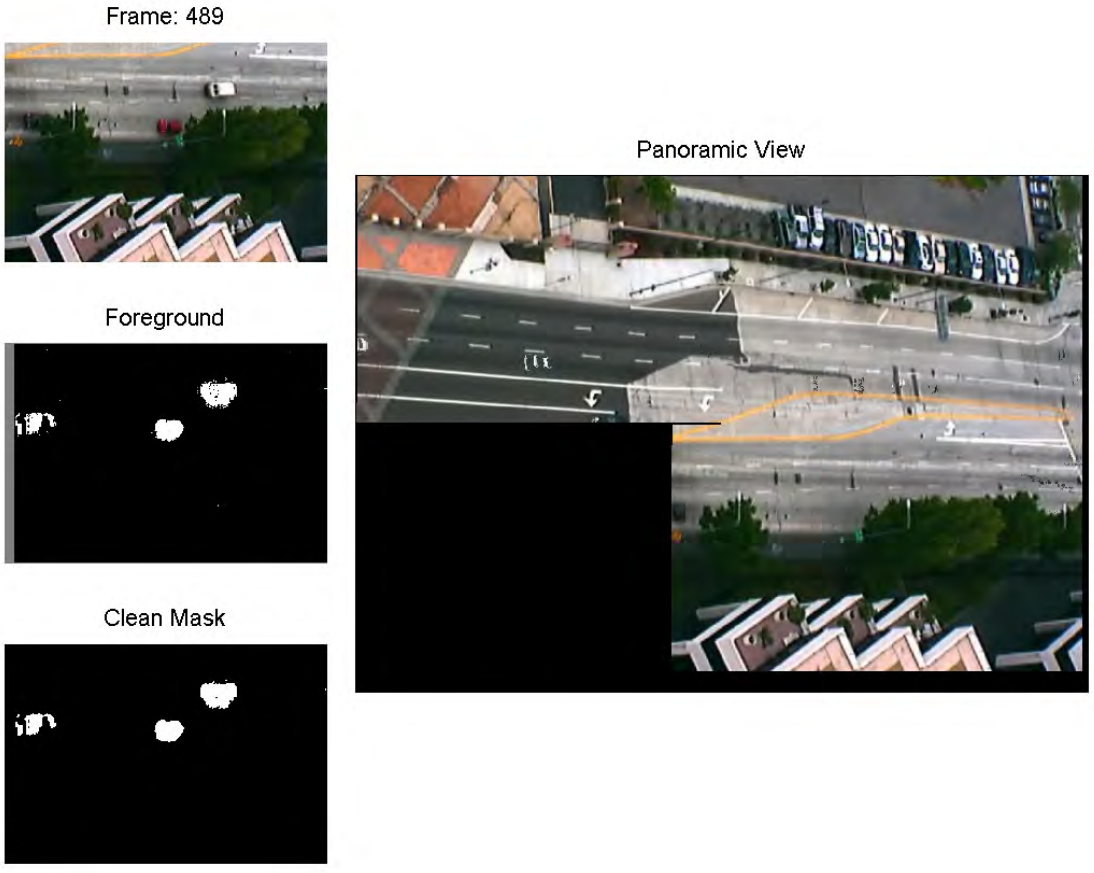


FIGURA 25: Análisis de la escena *lankershim*, donde se visualizan tres vehículos detectados de forma correcta.

Los imágenes que muestran resultados cualitativos sobre las secuencias anteriores se encuentran en las Figuras 24, 25, 26 y 27. Desafortunadamente, este tipo de secuencias no disponen de la *salida ideal* de la detección de movimiento (denominada *Ground Truth*, GT), normalmente utilizada para comparar la salida del algoritmo con el GT para determinar la eficiencia y eficacia. No obstante, para comprobar el buen funcionamiento de la metodología, se han publicado los resultados de algunos escenas en vídeos[5] donde se muestra su viabilidad.

5. Discusión y conclusiones

En el presente trabajo se ha desarrollado una metodología para el análisis de secuencias con cámaras en movimiento, basada en un modelo probabilístico del panorama completo de la escena y en métodos de correspondencia de imágenes. Se ha comprobado su buen funcionamiento en un conjunto que escenas que simulan y se aproximan al movimiento de una cámara PTZ en un entorno dado, con desplazamientos horizontales y verticales de la misma. Este tipo de cámaras cubren una mayor área visual que las cámaras estacionarias,

[5] https://drive.google.com/folderview?id=0Bwdn-fR124u5OUE4eFJ1dF9rY00&usp=sharing

FIGURA 26: Análisis de la escena *vikings*. Nótese que sólo uno de los vikingos se detecta como primer plano porque los otros dos permanecen inalterables durante el transcurso de la secuencia.

lo que implica un ahorro significativo en hardware al poder situar una cámara PTZ en lugar de varias fijas.

Además, se han utilizado otro tipo de secuencias relacionadas con juegos de ordenador para probar esta propuesta en otro tipo de entornos, resultando también satisfactoria la detección de movimiento. Nótese que todas las escenas requieren que el fondo de la misma sea los suficientemente heterogéneo para detectar puntos de interés adecuados en el proceso de correspondencia.

El principal objetivo de la metodología es la detección del primer plano, pero en este caso, además, se realiza una construcción del fondo completo de la escena o panorama de forma *online*, a la vez que se va realizando esta detección del movimiento. Los resultados obtenidos son satisfactorios, como ha podido comprobarse en la Sección 4.

No sólo se ha desarrollado el modelo propuesto de forma teórica, si no que se ha implementado un prototipo al que es posible pasarle como entrada una secuencia para su análisis. Este prototipo se ha desarrollado principalmente en Matlab, pero en combinación con la librería OpenCV para la ejecución de funciones o algoritmos que no se encontraban en la aplicación matemática. Así, se ha utilizado un *wrapper* o envoltorio

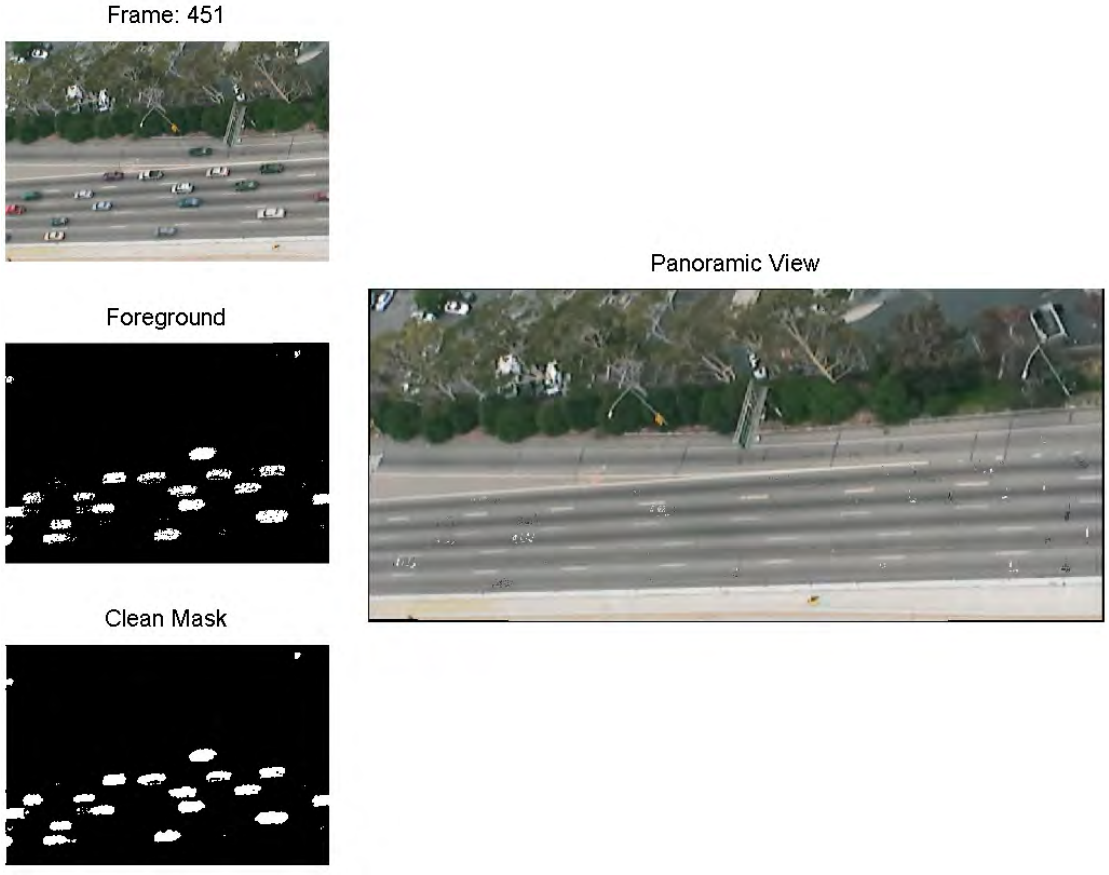


FIGURA 27: Análisis de la escena *Sb-camera*. Muestra el panorama completo.

que permite la ejecución de los métodos de OpenCV desde Matlab, lo cual aporta mucha potencia y funcionalidad. De forma adicional, ciertas partes *duras* del código se han desarrollado en C e integradas en Matlab a través de ficheros MEX. Con ello, conseguimos un aumento en la eficiencia del algoritmo, pues todo lo desarrollado en C siempre es más eficiente y rápido de ejecutar que lo implementado en Matlab, aunque también más costoso y complejo.

Uno de las principales dificultades encontradas es que esta metodología no funciona correctamente con escenas con una gran homogeneidad en el fondo. En estos casos, el algoritmo no detecta puntos de interés válidos en el fondo con los que realizar la correspondencia. Para solventar este problema, en trabajos futuros, deberíamos de utilizar técnicas de regionalización para segmentar inicialmente el fondo y, posteriormente, ubicar cada fotograma en el panorama regionalizado. Así, el emparejamiento se realizaría sobre dos fotogramas consecutivos en lugar de utilizar la media del panorama, evitando así suavizados que puedan provocar que no se detecten estos puntos de interés. No obstante habría que, de alguna forma, localizar el fotograma dentro del panorama, pero de una forma no tan exacta si no más difusa.

Para el objetivo de su publicación en congreso o en revistas de investigación de prestigio, se realizará una comparativa cuantitativa con otras propuestas existentes en la

bibliografía, lo que conllevará a la utilización o generación del *Ground Truth* para las secuencias analizadas.

En definitiva, aunque la metodología puede mejorarse para analizar un conjunto más amplio de secuencias, consigue el propósito de analizar secuencias con cámaras en movimiento, con lo que su utilidad para la video vigilancia de ciudades que instalen cámaras PTZ en lugar de cámaras tradicionales puede considerarse satisfactoria.

Bibliografía

[1] V. Saligrama, J. Konrad, and P. Jodoin. Video anomaly identification. *IEEE Signal Processing Magazine*, 27(5):18 – 33, 2010. doi: 10.1109/MSP.2010.937393.

[2] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[3] 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. ISSN 0162-8828. URL http://dx.doi.org/10.1109/34.598236.

[4] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[5] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003. doi: 10.1109/TPAMI.2003.1233909.

[6] H. J. Kushner and G. G. Yin. *Stochastic approximation and Recursive Algorithms and Applications*. Springer-Verlag, New York, NY, USA, 2003.

[7] T.L. Lai. Stochastic approximation. *Annals of Statistics*, 31(2):391–406, 2003.

[8] Z. Zivkovic and F. Van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651–656, 2004.

[9] Y. Pang, Y. Yuan, and X. Li. Iterative subspace analysis based on feature line distance. *IEEE Transactions on Image Processing*, 18(4):903–907, 2009.

[10] C.A. Tsao and Y.-C.I. Chang. A stochastic approximation view of boosting. *Computational Statistics and Data Analysis*, 52(1):325–334, 2007.

[11] S. Gadat and L. Younes. A stochastic algorithm for feature selection in pattern recognition. *Journal of Machine Learning Research*, 8:509–547, 2007.

[12] Y. Wei, S.B. Gelfand, and J.V. Krogmeier. Noise-constrained least mean squares algorithm. *IEEE Transactions on Signal Processing*, 49(9):1961–1970, 2001.

[13] E.K.P. Chong, I.-J. Wang, and S.R. Kulkarni. Noise conditions for prespecified convergence rates of stochastic approximation algorithms. *IEEE Transactions on Information Theory*, 45(2):810–814, 1999.

[14] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[15] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, 27(1):94–128, 1999.

[16] M.-A. Sato and S. Ishii. On-line em algorithm for the normalized Gaussian network. *Neural Computation*, 12(2):407–432, 2000.

[17] N. Bouguila and D. Ziou. A hybrid SEM algorithm for high-dimensional unsupervised learning using a finite generalized dirichlet mixture. *IEEE Transactions on Image Processing*, 15(9):2657–2668, 2006.

[18] E. López-Rubio, J. M. Ortiz-de-Lazcano-Lobato, and D. López-Rodríguez. Probabilistic PCA self-organizing maps. *IEEE Transactions on Neural Networks*, 20(9): 1474–1489, 2009. ISSN 1045-9227.

[19] E. López-Rubio. Multivariate Student-t self-organizing maps. *Neural Networks*, 22 (10):1432–1447, 2009. ISSN 0893-6080.

[20] S. Klein, M. Staring, and J.P.W. Pluim. Evaluation of optimization methods for nonrigid medical image registration using mutual information and b-splines. *IEEE Transactions on Image Processing*, 16(12):2879–2890, 2007.

[21] S. Klein, J.P.W. Pluim, M. Staring, and M.A. Viergever. Adaptive stochastic gradient descent optimisation for image registration. *International Journal of Computer Vision*, 81(3):227–239, 2009.

[22] Y.C. Trivedi and L. Kurz. A class of robust image processors. *Pattern Recognition*, 27(8):1111–1125, 1994.

[23] A.K.R. Chowdhury and R. Chellappa. Stochastic approximation and rate-distortion analysis for robust structure and motion estimation. *International Journal of Computer Vision*, 55(1):27–53, 2003.

[24] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, Nov. 2004.

[25] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000. ISSN 0920-5691.

[26] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147 – 151, 1988.

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

[28] My-Ha Le, Byung-Seok Woo, and Kang-Hyun Jo. A comparison of sift and harris conner features for correspondence points matching. In *Workshop on Frontiers of Computer Vision (FCV), 2011*, pages 1–4, 2011.

[29] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224 – 270, 1994.

[30] Milton Roberto Heinen and Paulo Martins Engel. NLOOK: a computational attention model for robot vision. *Journal of the Brazilian Computer Society*, 15:3 – 17, 09 2009.

[31] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Eighth IEEE International Conference on Computer Vision, ICCV*, volume 1, pages 525 – 531, 2001.

[32] Slim Amri, Walid Barhoumi, and Ezzeddine Zagrouba. A robust framework for joint background/foreground segmentation of complex video scenes filmed with freely moving camera. *Multimedia Tools Appl.*, 46(2-3):175 – 205, 2010. ISSN 1380-7501.

[33] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311 – 321, 1993.

[34] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331 – 340, 2009.

[35] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 2014.

[36] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[37] DWF Van Krevelen and R Poelman. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, 9(2):1, 2010.

[38] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381 – 395, 1981.