Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Seguimiento de personas mediante visión por computador para aplicaciones de control de personal

Autor: Miguel Ángel Maestre Trueba
Tutor: José Ramiro Martínez de Dios

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Seguimiento de personas mediante visión por computador para aplicaciones de control de personal

Autor:
Miguel Ángel Maestre Trueba

Tutor:
José Ramiro Martínez de Dios
Profesor Titular de Universidad

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Trabajo Fin de Grado: Seguimiento de personas mediante visión por computador para aplicaciones de control de personal

Autor: Miguel Ángel Maestre Trueba

Tutor: José Ramiro Martínez de Dios

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El secretario del Tribunal:

Fecha:

Resumen

El presente proyecto expone el diseño y desarrollo de un algoritmo cuyo objetivo es la realización del seguimiento a personas obteniendo datos de estos a partir del uso de una cámara de vídeo-vigilancia situada en un lugar fijo por encima de una zona de paso por donde transitan personas. La meta de este trabajo se alcanza mediante el uso técnicas pertenecientes a la visión por computador y al control. Una vez alcanzada dicha meta, el proyecto servirá como punto inicial para nuevos trabajos también basados en el seguimiento de personas.

Abstract

This project describes the design and development of an algorithm which aims to track people, obtaining data from them by using a video-surveillance camera located at a fixed location above a passageway along which people walk. The goal of this work is achieved by using computer vision and control techniques. After reaching this goal, the project will serve as a starting point for new tracking based works.

Índice

Notación

track	Unidad de seguimiento a un objeto.
R, G, B	Valores de colores rojo, verde y azul en el píxel.
MAX, MIN	Máximo y mínimo de los valores R, G y B del píxel.
H, S, V	Valores de matiz, saturación e intensidad luminosa en el píxel.
$B(x, y, t)$	Imagen de fondo en función del valor horizontal y vertical de píxeles y del instante de tiempo actual.
$I(x, y, t)$	Imagen actual analizada del vídeo.
n	Número de imágenes tomadas para el cálculo del fondo
$\bar{x}, \bar{y}$	Componentes X e Y del centroide de una región.
m_{pq}	Momentos espaciales de una región.
f_{ij}	Valor de intensidad de un píxel.
v_x, v_y	Velocidades de un objeto en movimiento en la imagen.
$x_{región}, y_{región}$	Posición del centroide de la región detectada.
x_{track}, y_{track}	Posición del centroide de la región asociada al track.
$x_{predicho}, y_{predicho}$	Posición predicha del objeto del track.
d	Coste de relacionar las posiciones de regiones con posiciones predichas de los tracks.
$x_{desaparecido}, y_{desaparecido}$	Posición del centroide de la región asociada a un track que ha sido clasificado como no visible en la imagen.
α, β	Parámetros usados en el cálculo del coste.

1 INTRODUCCIÓN

El seguimiento de objetos en entornos del mundo real se refiere al problema de usar las medidas de un sensor para determinar la localización, trayectoria o distintas características del objeto analizado. El sensor utilizado para el estudio puede ser cualquier dispositivo de medida tal como un radar, sonar, ultrasonido, una cámara u otros y es por eso por lo que esta técnica tiene múltiples aplicaciones.

Esta tarea ha ido ganando atención debido a su potencial tanto comercial como académico y sus aplicaciones van desde el seguimiento de aeronaves mediante radares así como vigilancia mediante el uso de cámaras. En este proyecto, el objetivo que se quiere alcanzar es realizar el seguimiento a personas mediante el uso de técnicas de visión por computador para el control de personal, es decir, mediante el uso de cámaras de vídeo-vigilancia. El uso de vigilancia basada en vídeos digitales está creciendo significativamente y cada vez se usa más en empresas, comercios, aeropuertos, etc. Debido a la gran cantidad de información que recopilan las cámaras de seguridad, a parte del componente humano, es necesario instalar un sistema inteligente que interprete dicha información en función de la aplicación que se le quiera dar a este.

Este proyecto no va a llegar a tales puntos de complejidad, sino que simplemente va a consistir en realizar el seguimiento de personas mediante el uso de una única cámara situada en un punto fijo. A estas personas, en cada imagen del vídeo, se les va a determinar su posición y sus características como por ejemplo puede ser su velocidad así como también se les va a calcular su trayectoria completa en la zona dentro de los límites del vídeo. La resolución de este problema no es trivial y por tanto las distintas partes que lo componen si tendrán un cierto de grado de complejidad que habrá que estudiar con detenimiento.

Aunque existen múltiples métodos para la resolución de la tarea de seguimiento, todavía existen problemas no resueltos que dificultarán su correcta realización, aunque esto no significa que no se pueda realizar dicha tarea. Existen métodos que aunque no resuelvan todos los problemas, obtienen resultados suficientemente aproximados que sirven y cumplen adecuadamente. En este documento se expone el diseño de un algoritmo para el seguimiento de personas mediante un método aproximado y se estudiarán sus resultados, tantos positivos como negativos. Dicho algoritmo va a estar formado por dos partes principalmente: la primera de ellas dedicada al tratamiento de las imágenes entregadas por la cámara para adaptarlas a la siguiente parte, la de seguimiento, en la que se obtendrá la información de las personas analizadas. De estas dos partes, la que más problemas va a ocasionar es la primera, la de procesamiento, y al depender el resto de esta, existe el riesgo de que se arrastren errores y se generen nuevos errores más complejos. Cada parte del algoritmo se estudiará en detalle y se buscarán soluciones a los distintos problemas que puedan ocasionarse.

Este trabajo se organiza en 6 capítulos. El primero de ellos se corresponde con esta introducción al tema estudiado. En el siguiente capítulo se realiza una descripción general y abreviada de cada parte del algoritmo diseñada y se muestra la estructura de todo el conjunto de partes que forman el sistema. En los capítulos 3 y 4 se describen en detalle los métodos llevados a cabo para la resolución de los dos grandes bloques que definen el sistema, cada uno en un capítulo. A continuación, en el capítulo 5, se realizan los distintos experimentos de cada uno de los bloques así como los experimentos de validación para la comprobación del correcto funcionamiento de cada bloque y la observación de posibles fallos que haya que corregir en desarrollos futuros. Y ya por último, en el capítulo 6 se exponen las conclusiones obtenidas a lo largo de la realización de este proyecto, así como la descripción de

desarrollos futuros a partir de este.

Para la programación del algoritmo del proyecto se usa el lenguaje C++ y la librería de visión por computador OpenCV[1]. El sistema operativo usado para todo el desarrollo ha sido Ubuntu 14.04.

[1] OpenCV: Open Source Computer Vision. Librería de libre acceso con funciones en C++ para aplicaciones relacionadas con el procesamiento de imágenes.

2 DESCRIPCIÓN GENERAL

Si tu intención es describir la verdad, hazlo con sencillez y la elegancia déjasela al sastre.

ALBERT EINSTEIN

En este capítulo se va a realizar una breve descripción de qué se hace para solucionar el problema planteado y cumplir los objetivos propuestos. Para ello, se va a explicar resumidamente cada sección del algoritmo diseñado y cuál es su función.

Más adelante en el capítulo se va a mostrar el diagrama de bloques del sistema completo, enseñando las secciones y las subsecciones dentro de estas.

2.1 Esquema general

En este proyecto, el algoritmo general está compuesto por dos grandes bloques. El primero es el de procesamiento de imágenes y el segundo es el de seguimiento. A continuación se explica la función de cada uno resumidamente, así como su estructura y sus diferentes partes.

2.1.1 Procesamiento de imágenes

La única fuente de información que se tiene para la realización de este proyecto es el metraje de una cámara o vídeos ya grabados. Para el correcto funcionamiento del seguimiento de personas es necesario tratar estas imágenes dadas de una manera concreta para que puedan extraerse de ellas los datos necesarios. Por ello, es necesario la creación de un algoritmo que a partir de las imágenes dadas se generen unas nuevas imágenes aptas para la obtención de los datos necesarios para el seguimiento.

Lo que este algoritmo hace concretamente es convertir imágenes a color en imágenes binarias en las que se ha sustraído previamente el fondo, es decir, imágenes en blanco y negro en las que lo único que se representa en blanco son las personas en movimiento.

Para llegar a esta imagen objetivo, antes han de hacerse distintas operaciones con la imagen dada. Primero se convierte a un modelo de color más adecuado como el HSV, el cual se explicará más adelante. Después es necesario la generación de un fondo dinámico a partir de las imágenes del vídeo. Debido a que en este proyecto se quiere realizar el seguimiento de personas desde un punto fijo, para poder sustraer los objetos en movimiento de la imagen, se genera un fondo representado por una imagen en la que no existe movimiento. Su realización se explica detalladamente en el próximo capítulo.

Una vez que se ha convertido a HSV y se ha generado un fondo para la imagen analizada, hay que restarle el fondo a dicha imagen para sustraer a las personas en movimiento de la imagen. Y posteriormente, realizar una binarización a partir de un umbral designado para segmentar los objetos en movimiento.

Como en muchas otras aplicaciones de procesamiento de imágenes, en la fase de la sustracción del fondo y segmentación también aparece ruido no deseado que hay que intentar corregir. Para ello, una vez se tenga la segmentación realizada, se aplican sobre la imagen dos operaciones morfológicas, erosión y dilatación. La primera se encarga de la eliminación de ruido de pequeño tamaño y la segunda de expandir los objetos de interés por si existieran errores parciales de segmentación.

Una vez realizadas todas estas operaciones, se obtiene la imagen objetivo de este bloque, una imagen binaria en la que los únicos objetos representados son las personas en movimiento. A partir de esta imagen se puede empezar a hacer el seguimiento de dichas personas.

En la figura 2.1 puede verse una imagen de un vídeo acompañada de la imagen ya tratada por el bloque de procesamiento (figura 2.2).

Figura 2-1. Imagen de vídeo real.

Figura 2-2. Resultado del procesamiento de imágenes.

2.1.2 Seguimiento

Contando ya con la imagen procesada, puede realizarse el seguimiento de los objetos representados en blanco. En este bloque, es necesario convertir la información dada por la imagen en datos numéricos.

El problema del seguimiento se divide principalmente en dos partes. La primera de ellas se corresponde con la detección de las regiones segmentadas en la imagen y la segunda con el algoritmo de seguimiento como tal.

En la detección de regiones, mediante el uso de distintas funciones, se almacenan datos de interés de estas como son las posiciones o los tamaños. Con estos datos de las regiones, se podrá realizar el seguimiento. Por ejemplo, para la imagen mostrada en 2.2, se detectará una región y por tanto, se obtendrá una posición para esta así como un tamaño.

El problema de mayor importancia que resuelve el seguimiento es la capacidad de asociar distintas regiones detectadas en imágenes independientes como un mismo objeto que se encuentra en movimiento. A partir de los datos de las regiones detectadas, creará un *"track"* o unidad de seguimiento para cada objeto en movimiento e irá actualizando su información en cada imagen con los datos de nuevas regiones detectadas.

Al tratarse de un sistema de seguimiento de múltiples objetos de forma simultánea, es necesario contar con un algoritmo de tres fases. Esto ha de ser así debido a que al contar con más de una región en una misma imagen, cada *"track"* o unidad de seguimiento tiene que asegurarse de que la región de la que toma los datos para actualizarse, es la más adecuada. Las tres fases del algoritmo son predicción,

asociación y actualización.

La fase de predicción se usa para estimar dónde va a estar situado el objeto en la imagen actual a partir de la información conocida en el *"track"* de dicho objeto en la imagen anterior. Esta posición predicha será utilizada en la siguiente fase, la asociación.

La fase de asociación se encarga de relacionar todas las posiciones predichas de los objetos con todas las posiciones de las regiones detectadas en cada imagen calculando las distancias de unas con otras. Y una vez calculadas, mediante el criterio de la mínima distancia, se asocian región y objeto. Esta fase, aparte de esto, también se encarga de clasificar el tipo de *"track"*, ya que no todos tienen las mismas características y no se actualizan de la misma forma. Esto se explicará más en detalle en el capítulo de seguimiento.

Por último, la fase de actualización modificará la información del track con la nueva información dada por las regiones a las que los *"tracks"* se asocian. Aparte de la información dada por la región, el *"track"* cuenta con más datos que también se actualizan. Como se dice en el anterior párrafo, al existir distintos tipos de *"tracks"*, la actualización de estos será diferente para cada uno.

Como ya se ha mencionado, existen distintos tipos de *"tracks"* y cada uno de estos tipos se actualiza de una manera concreta. Y ¿por qué existen distintos tipos? Esto se debe a que hay que tener en cuenta cuando estos se crean o cuando no se asocian con ninguna región, así como que también hay que tener en cuenta distintos tipos de errores de segmentación que comprometan al seguimiento y actuar de la manera correcta para corregir cualquier equivocación que pueda darse.

Por último, cabe mencionar que todos los puntos de la trayectoria de los objetos en movimiento se almacenan en un fichero. Este fichero es leído posteriormente por otro programa para realizar distintas acciones con él, como por ejemplo hacer la cuenta del número de personas que entran y salen de la zona de paso o en qué imagen del vídeo entran y en cuál salen.

2.2 Diagrama de bloques

El diagrama de bloques general viene representado en la figura 2.3. Se observan los dos grandes bloques: procesamiento de imágenes y seguimiento y dentro de ellos todos los sub-bloques pertenecientes a estos, cada uno con una función específica en el algoritmo.

Respecto al primer bloque, cabe mencionar que al realizar la conversión de la imagen a HSV, se separan los tres canales H, S y V, descartando el canal H. Por tanto, el resto de operaciones del bloque se van a realizar tanto para el canal S como para el canal V. Una vez aplicada la dilatación sobre ambas imágenes, se solapan en una única y se consigue la imagen objetivo que va a servir como referencia en el bloque de seguimiento.

En el segundo bloque, se puede observar el bucle formado por las tres fases, predicción, asociación y actualización y como a su vez estas dependen de la detección de regiones. Es en la fase de asociación donde se relacionan los datos de las regiones con las predicciones de los *"tracks"* y una vez que se han asociado región y *"track"*, se pasa a la fase de actualización. Una vez actualizados, tanto la predicción como el almacenamiento de la trayectoria necesitarán los nuevos datos de esta fase. También cabe decir que para algunos de los casos de clasificación de los *"tracks"*, las predicciones serán necesarias para la actualización.

Este algoritmo general se ejecuta para todas y cada una de las imágenes del vídeo que se esté analizando.

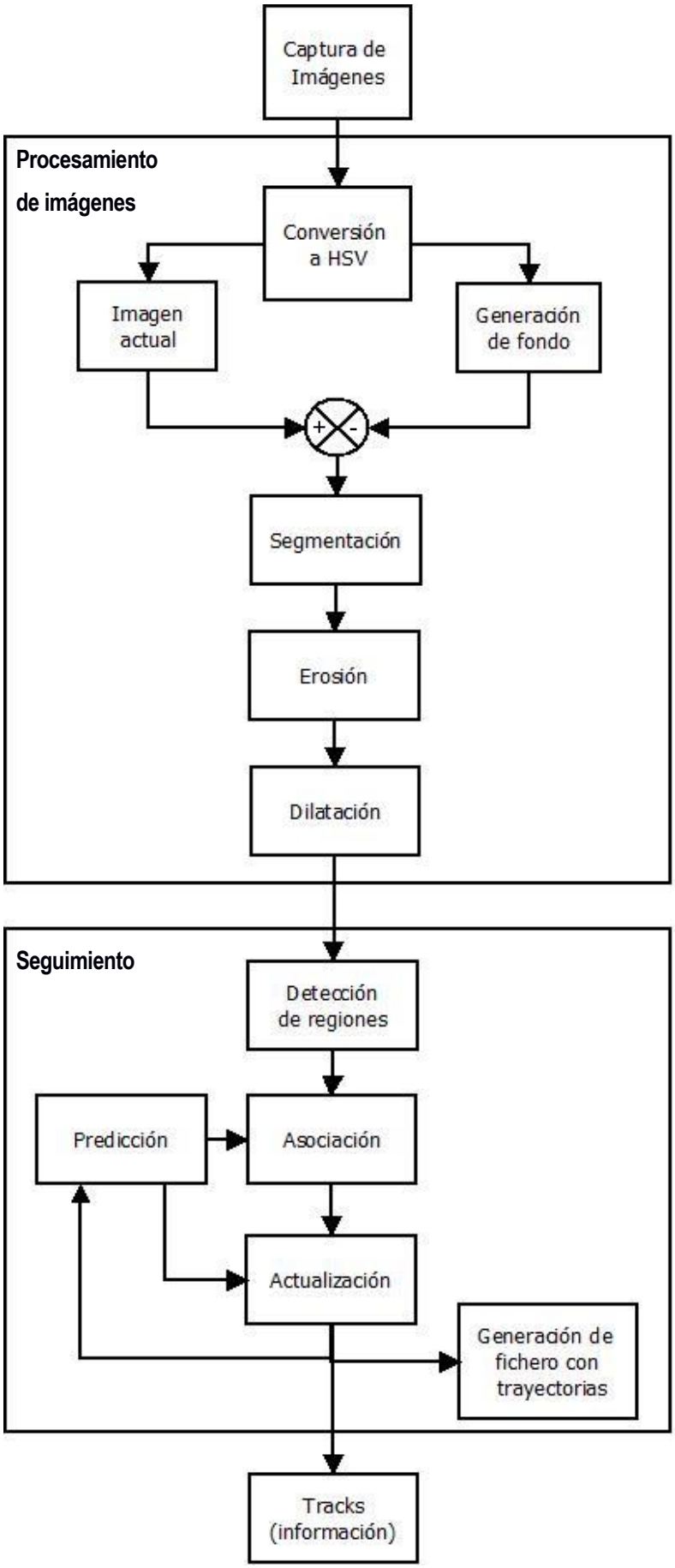

Figura 2-3. Diagrama de bloques general.

2.3 Conclusiones

Cada bloque descrito a lo largo de este capítulo será estudiado en detalle a continuación en esta memoria. También cabe mencionar que el método elegido para resolver el problema del seguimiento no es la única forma de resolverlo, existen múltiples formas de diseñar el algoritmo.

Al no haber ahondado aún en ninguna de las partes del sistema, no se pueden obtener muchas conclusiones pero si alguna observación. Examinando el diagrama se puede apreciar que todas las partes del sistema, tanto bloques como sub-bloques, son modulares y muy independientes el uno del otro, por tanto, fácilmente diferenciables.

3 PROCESAMIENTO DE IMÁGENES

La ciencia se compone de errores, que a su vez son los pasos hacia la verdad.

JULIO VERNE

Este capítulo describe el primer gran bloque de este proyecto, el procesado de imágenes. La función de dicho bloque consiste en tomar las imágenes capturadas por la fuente de vídeo. Esta puede ser tanto un archivo de película como una cámara capturando en tiempo real y realizar sobre cada una de ellas distintas operaciones para así obtener unas nuevas imágenes que serán necesarias en el siguiente bloque de seguimiento.

El bloque de procesamiento va a estar compuesto por tres algoritmos internos: el primero de ellos será la generación de un fondo con el que puedan compararse las imágenes dadas por el vídeo. Después se realizará la segmentación en la que la imagen será binarizada y por último, se aplicarán dos operaciones morfológicas, erosión y dilatación. A continuación, se explicará de manera detallada la estructura de este bloque así como cada algoritmo interno.

3.1 Esquema General

Como se ha mencionado previamente, el funcionamiento de este bloque se basa en capturar cada imagen dada por la fuente de vídeo y realizar sobre ellas distintas operaciones. Con esto se genera una nueva imagen apta para posteriormente realizar el seguimiento a partir de ciertos datos extraídos de esta. El diagrama mostrado en la figura 3.1, describe a grandes rasgos todo el bloque.

El diagrama representa de una manera ordenada toda la secuencia del procesado:

- Primero se captura la imagen del video, esta será el único dato para el algoritmo.
- Esta imagen está representada en RGB[2], por tanto el siguiente bloque se encarga de convertirla al modelo HSV[3] y también de separar cada canal para poder operar sobre ellos por separado. Para los siguientes bloques sólo se usarán los canales S y V.
- Para cada canal se genera un fondo que posteriormente pasará a ser utilizado en la segmentación.
- Para esta segmentación, se toman la imagen de cada canal y el correspondiente fondo generado y con ello se sustraen las personas en movimiento de la imagen.
- Con la nueva imagen binaria obtenida de la segmentación, se aplican las operaciones morfológicas, erosión y dilatación para mejorar el resultado.
- Una vez que se tiene la imagen dilatada de cada canal, se solapan formando una única imagen, también binaria. Con esto, ya se tiene la imagen objetivo para realizar el seguimiento.
- Este algoritmo se realiza para todas las imágenes del video.

[2] RGB: Red, Green, Blue. Rojo, Verde, Azul.
[3] HSV: Hue, Saturation, Value. Matiz, Saturación, Intensidad luminosa.

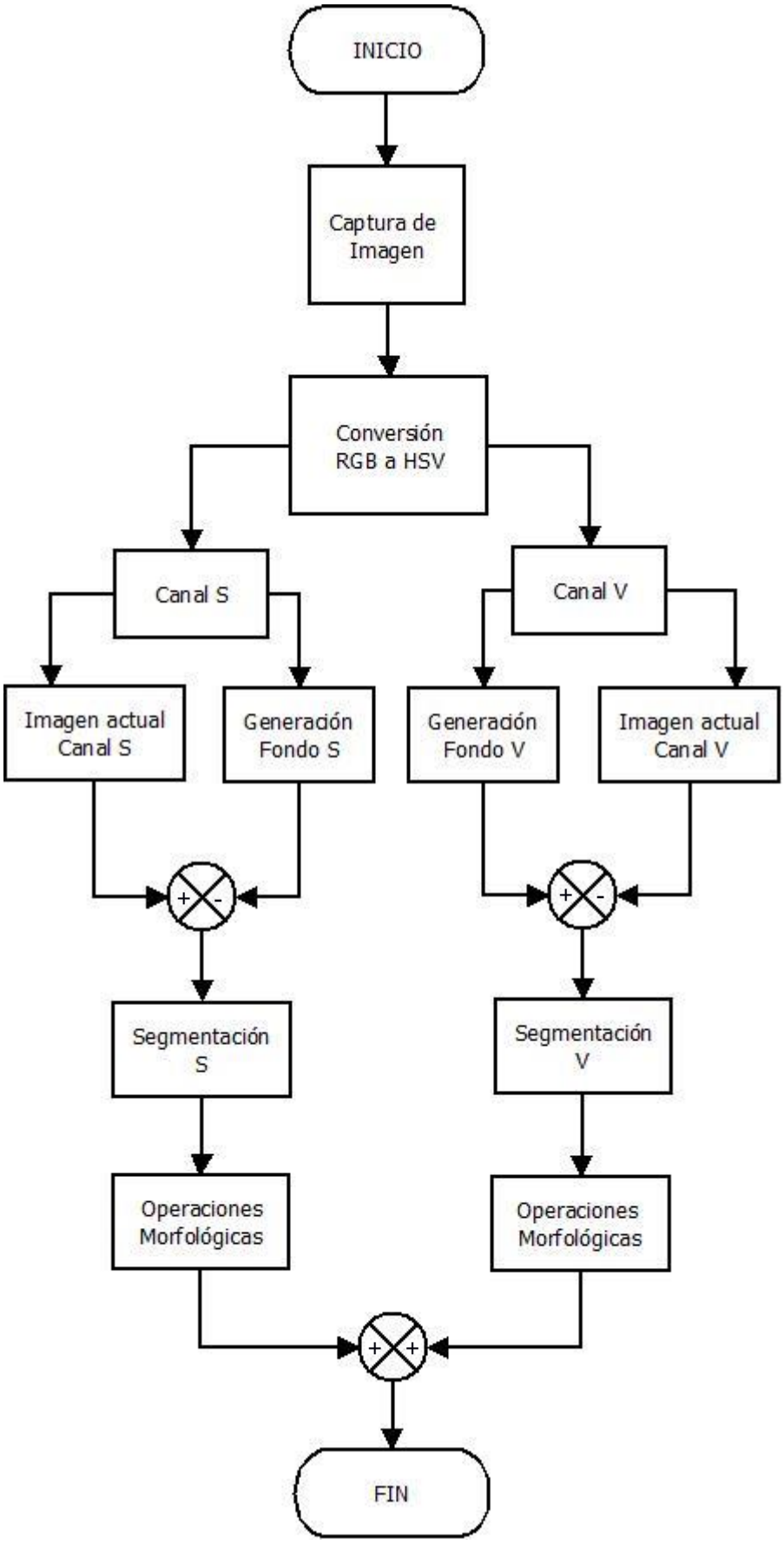

Figura 3-1. Algoritmo general del bloque.

3.2 Conversión de color

Las imágenes de entrada que se reciben generalmente vienen en formato de color RGB, es decir, en tres canales solapados de colores rojo, verde y azul. En esta aplicación, el objetivo es reconocer personas, independientemente del color de ropa o atuendo que lleven puestas y por ello, el formato RGB puede no ser lo suficientemente útil. Para obtener mejores resultados, se hace uso de otro modelo de color, HSV. Siendo H el matiz (*"Hue"*), S el nivel de saturación (*"Saturation"*) y V el nivel de brillo (*"Value"*). En la figura 3.2 puede verse el espacio de color HSV representado en forma cónica.

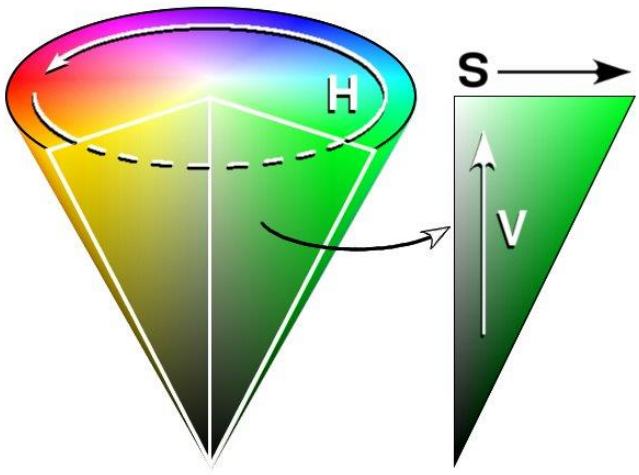

Figura 3-2. Cono de colores del espacio HSV.

Siendo *MAX* el máximo de los componentes *(R, G, B)* y *MIN* como el mínimo de esos mismos valores, los componentes HSV se pueden calcular a partir de las siguientes expresiones:

$$H = \begin{cases} \text{no definido}, & \text{si } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \text{ y } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{si } MAX = R \text{ y } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{si } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 240°, & \text{si } MAX = B \end{cases} \tag{3.1}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases} \tag{3.2}$$

$$V = MAX \tag{3.3}$$

El uso de este modelo de color tiene la ventaja de que todos los colores están agrupados en un único canal por lo que facilitará el reconocimiento de las personas en el espacio de visión de la cámara basándose en la cantidad de brillo y de saturación de los colores detectados.

Tal y como se ha mostrado en el diagrama de bloques general del procesamiento de imágenes en el anterior apartado, la conversión RGB a HSV es la primera operación que se realiza en el código. En las figuras 3.3, 3.4 y 3.5 pueden observarse los canales H, S y V respectivamente representado en escala de grises:

Figura 3-3. Canal *Hue* de una imagen.

Figura 3-4. Canal *Saturation* de una imagen.

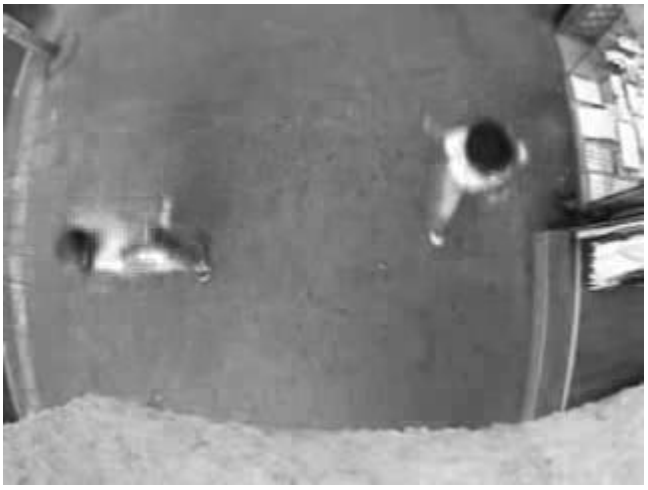

Figura 3-5. Canal *Value* de una imagen.

Estas imágenes representan el mismo fotograma de un vídeo usado en el desarrollo del proyecto pero en sus tres canales H, S y V separados. Viendo el canal *Hue* se puede apreciar que difícilmente puede reconocerse a las personas y además genera ruido que complicaría dicho reconocimiento, por lo que no será muy útil. Sin embargo, en los canales de *Value* y *Saturation*

sí que se obtienen resultados positivos. En el de saturación se ven las personas en un tono más claro de grises en comparación con el fondo y mientras más oscura sea la ropa o las características de los viandantes, más claro se ven. Sin embargo, en el canal de brillo la representación de la imagen es bastante similar al resultado de convertir la imagen RGB a gris, resaltando más las prendas de colores más claras y cercanas al blanco.

Visto esto, se concluye que solo es necesario aplicar los algoritmos de procesamiento sobre los canales de brillo y saturación de los colores de la imagen. Desacoplando así el canal de matiz, ya que este no posee información relevante para el reconocimiento de las personas y ahorrándose así, coste computacional innecesario. Ya que el canal de saturación destaca los colores oscuros y el canal de brillo los colores más claros, éstos se complementarán perfectamente.

3.3 Generación de fondos

Para poder analizar el movimiento de personas en una zona de paso es necesario tener en cuenta que la cámara que captura la información se encuentra en una posición fija, por lo que el fondo de todas las imágenes recogidas va a ser el mismo. Dicha imagen de fondo sólo se diferenciará del resto de imágenes en que en ésta no existe movimiento alguno, es decir, no hay personas pasando. A continuación se muestra el diagrama de bloques del algoritmo en la figura 3.6:

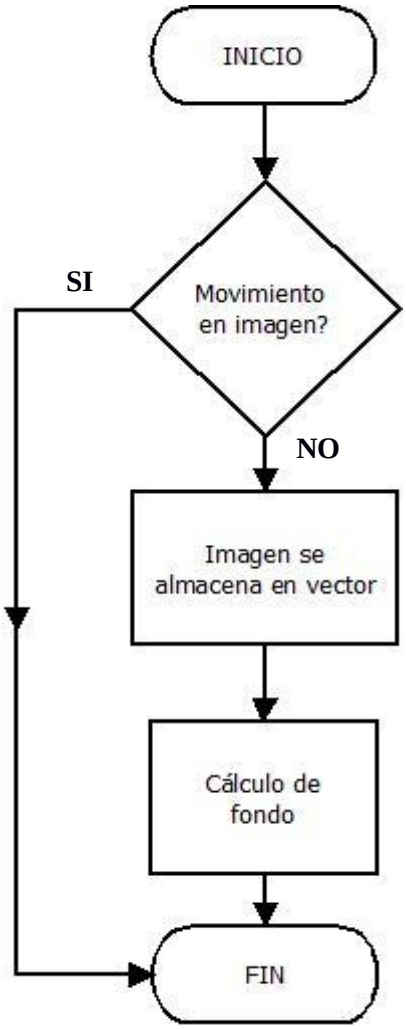

Figura 3-6. Diagrama de generación de fondo

3.3.1 Detección de Movimiento

Para generar un fondo correctamente, lo primero que se hace es detectar si existe movimiento o no en el video. Esto se consigue comparando cada imagen del video con una única captura del fondo, tomada al inicio del algoritmo (ambos convertidos a escala de grises). El resultado de la comparación será una nueva imagen, también en grises, la cual es binarizada tomando como umbral un valor situado entre 0 y 255. El valor de éste se toma a base de prueba y error. Finalmente se elige un umbral de valor generalmente bajo, como por ejemplo 20. Una vez que se ha umbralizado el resultado de comparar la imagen actual con la captura del fondo, el siguiente paso consiste en contar el número de píxeles blancos o con valor lógico 1. Si éstos superan una cierta cantidad, se considera que hay movimiento y si no la superan pues se supone lo contrario, que no existe movimiento en la imagen. Esta cifra de píxeles utilizada en la condición se obtiene contando inicialmente el número de píxeles blancos que hay en la imagen binarizada sin movimiento y ajustando aproximadamente a partir de la realización de pruebas.

A continuación se muestran las imágenes descritas anteriormente, la figura 3.7 se corresponde con la captura del fondo realizada al inicio del algoritmo, la figura 3.8 es la captura de una imagen cualquiera del vídeo y la figura 3.9 es el resultado de restar la imagen cualquiera con la captura del fondo. En el algoritmo 3.1 se muestra la estructura básica de la detección de movimiento.

Algoritmo 3.1 Algoritmo para detección de movimiento

```
Capturar de imagen de fondo al inicio de video
for bucle infinito do
    Capturar imagen de video
    Restar: imagen de video – imagen de fondo
    Binarizar imagen resultado
    Contar número de píxeles blancos en imagen resultado
    if número de píxeles blancos > cantidad elegida
        Hay movimiento
    else
        No hay movimiento
    end if
end for
```

Figura 3-7. Fondo capturado al inicio del algoritmo.

Figura 3-8. Imagen del vídeo.

Figura 3-9. Resultado de la resta de la imagen de la figura 3.8 con la imagen de la figura 3.7.

Puede verse que la imagen resultado de la resta reconoce los objetos no pertenecientes al fondo. En la figura 3.10 se muestra el resultado de la binarización. El único objetivo de esta imagen es demostrar que la densidad de píxeles blancos es mayor cuando hay personas en la escena que cuando no.

Figura 3-10. Resultado de la binarización de la imagen de la figura 3.9.

3.3.2 Vector de imágenes

Una vez se tiene conocimiento de cuándo no hay movimiento, es necesario almacenar en un vector un cierto número de imágenes para las operaciones que se realizarán para generar el fondo. El algoritmo que se aplica para esto es muy simple y permite que cada imagen en el que no haya movimiento, tenga asociado un vector que contiene un cierto número de imágenes de las mismas características. Este vector va actualizando sus componentes con nuevos resultados, también sin movimiento, hasta que se llena. En el siguiente apartado se explicará el porqué de la importancia del número de imágenes que contenga el vector. Por último, cabe decir que este algoritmo se realiza tanto para el canal *Saturation* como para el canal *Value*.

3.3.3 Cálculo de Fondo

Una vez que se tienen los vectores de imágenes sin movimiento para cada canal, hay que realizar la media aritmética de todos los píxeles(i,j) de cada imagen componente del vector.

$$B(x, y, t) = \frac{1}{n}\sum_{i=0}^{n-1} I(x, y, t-1) \tag{3.4}$$

siendo B el fondo generado en el instante *t* en función de los píxeles horizontales (*x*) y verticales (*y*). I se corresponde con todas las imágenes pertenecientes al vector de "no movimiento" en los instantes de tiempo anteriores al actual hasta *n* instantes anteriores.

Al completar esta operación, se consigue como resultado una imagen que va a servir como fondo para posteriormente sustraer los objetos en movimiento del vídeo. Este algoritmo se realiza para cada imagen del vídeo, por lo que el fondo generado es dinámico y va cambiado. Que el resultado sea positivo depende del número de imágenes almacenadas en el vector. Si se almacenan pocas imágenes, el cálculo de la media puede generar un fondo poco preciso. En principio, se toman cinco imágenes para el cálculo de la media y los resultados son buenos para el video de prueba. Aunque cinco fotogramas son pocos y pueden darse fallos, por tanto se modifica el algoritmo para realizar la media de muchas más imágenes sin movimiento ya pasados hasta el actual, dando así resultados muchos más robustos. Aunque esto también tiene un problema y es que el computo puede enlentecerse si el vector es excesivamente grande.

A continuación se muestran los resultados de fondos generados en ambos canales de saturación y de brillo en las figuras 3.11 y 3.12 respectivamente. El vector usado para estos resultados es de tamaño 30.

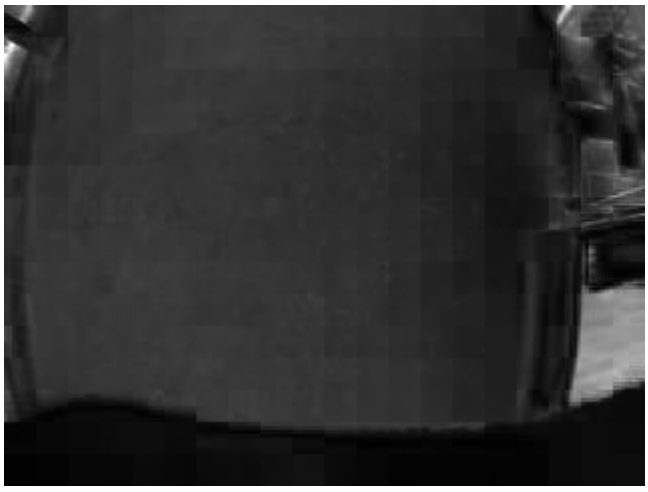

Figura 3-11. Fondo generado en el canal *Saturation*.

Figura 3-12. Fondo generado en el canal *Value*.

3.4 Segmentación

Una vez que se tiene un fondo generado, el siguiente paso del procesamiento de imágenes consiste en sustraer dicho fondo de la imagen para poder trabajar únicamente con los objetos en movimiento, que es lo que interesa del vídeo. La sustracción se divide en dos simples pasos que van a ser restar la imagen actual con el fondo generado para esa imagen y binarizar el resultado. Después de sustraer y segmentar se aplican las operaciones morfológicas, erosión y dilatación, sobre las imágenes resultantes. La figura 3.13 representa el diagrama de bloques de la segmentación completa.

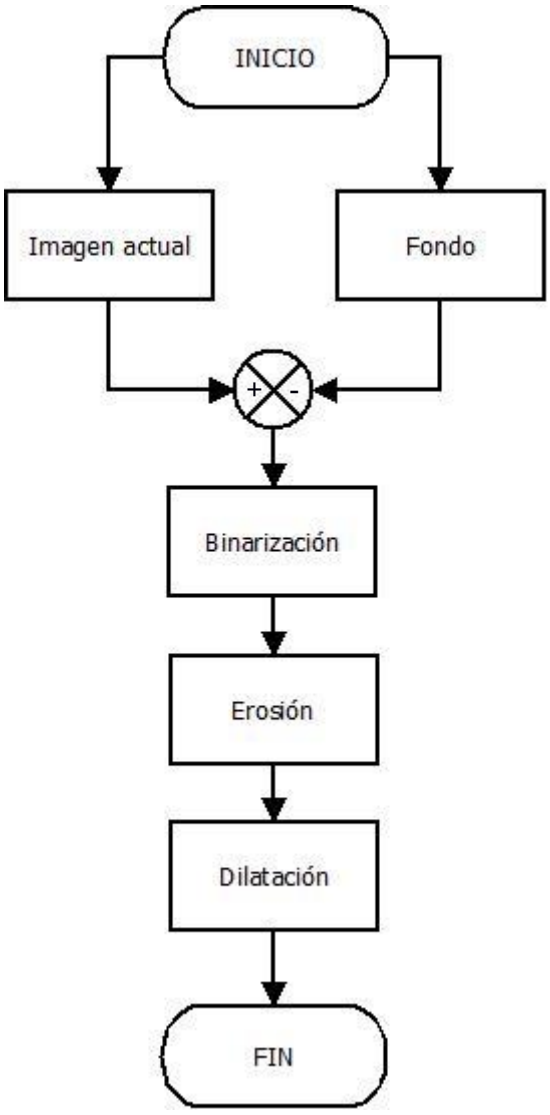

Figura 3-13. Diagrama de la segmentación.

3.4.1 Resta

Tal y como se realizó en el apartado anterior en el bloque de detección de movimiento, se vuelve a coger la imagen actual (previamente convertida en HSV y separada por canales) y se le resta la imagen resultante del algoritmo de generación de fondo. Esta operación, como todas las posteriores a la conversión de color, se aplica en ambos canales *Saturation* y *Value*. Las figuras 3.14 y 3.15 muestran en escala de grises ambos canales de interés de una imagen cualquiera del

vídeo de prueba.

Figura 3-14. Resultado de restar la imagen actual al fondo generado en el canal *Saturation*.

Figura 3-15. Resultado de restar la imagen actual al fondo generado en el canal *Value*.

Aunque sea difícil apreciarlo, se trata del mismo instante del vídeo, para verificarlo, la figura 3.16 muestra dicha imagen. En él hay dos personas, una vestida de colores principalmente blancos y la otra de colores oscuros. Como se explicó previamente, los canales de saturación y brillo tienen estas características y ayudarán a la obtención de mejores resultados.

Figura 3-16.Imagen del video estudiada.

3.4.2 Binarización

El siguiente paso de este bloque de segmentación consiste en binarizar las imágenes mostradas anteriormente en las figuras 3.14 y 3.15. Para ello, se aplica un simple *threshold* que a partir de un umbral dado, todos los píxeles con valor menor a este umbral sean 0 y todos los que estén por encima, tengan valor 1. Las figuras 3.17 y 3.18 muestran la binarización resultante de aplicar un umbral de valor 40 sobre 255 (máximo):

Figura 3-17. Resultado de la binarización de la imagen de la figura 3.14 con umbral 40.

Figura 3-18. Resultado de la binarización de la imagen de la figura 3.15 con umbral 40.

Puede verse que el resultado no es malo, aunque existe ruido que habría que eliminar. Si juntamos las dos imágenes en una sola, puede ver como se complementan muy bien los canales de brillo y saturación. La figura 3.19 representa lo último.

Figura 3-19. Resultado de sumar las imágenes de las figuras 3.17 y 3.18 en una única imagen.

La imagen mostrada en la figura 3.19, es sólo para enseñar el resultado de la segmentación. Ésta no será utilizada en las próximas operaciones debido a que se seguirán utilizando los dos canales por separado.

Por último, ¿qué umbral tomar para realizar la binarización? Para este proyecto, el umbral se ha elegido en base a los resultados de realizar distintas pruebas. Por ejemplo, si elegimos un umbral de valor 20 para ambos canales, se obtiene lo siguiente, mostrado en las figuras 3.20 y 3.21:

Figura 3-20. Resultado de la binarización de la imagen de la figura 3.14 con umbral 20.

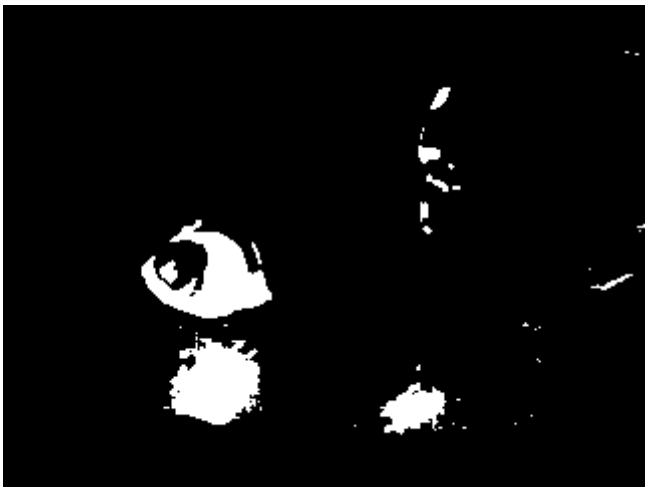

Figura 3-21. Resultado de la binarización de la imagen de la figura 3.15 con umbral 20.

Puede verse que al ser un umbral más relajado, existe más ruido. Por tanto, no conviene tener un umbral tan bajo. Si ahora se sube el umbral por encima del inicial (40) a 60, por ejemplo en las figuras 3.22 y 3.23:

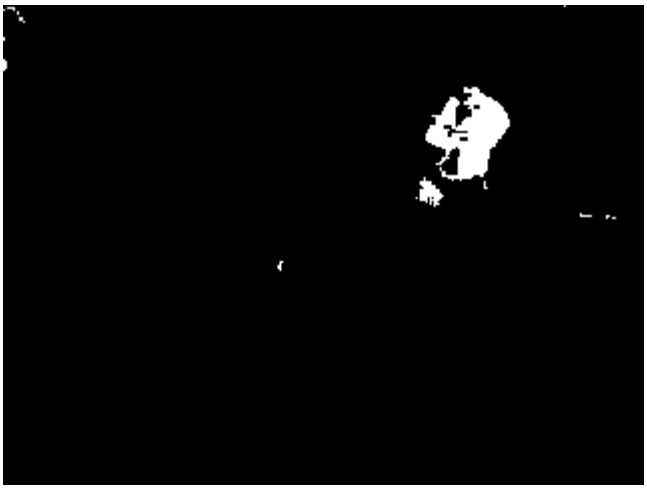

Figura 3-22. Resultado de la binarización de la imagen de la figura 3.14 con umbral 60.

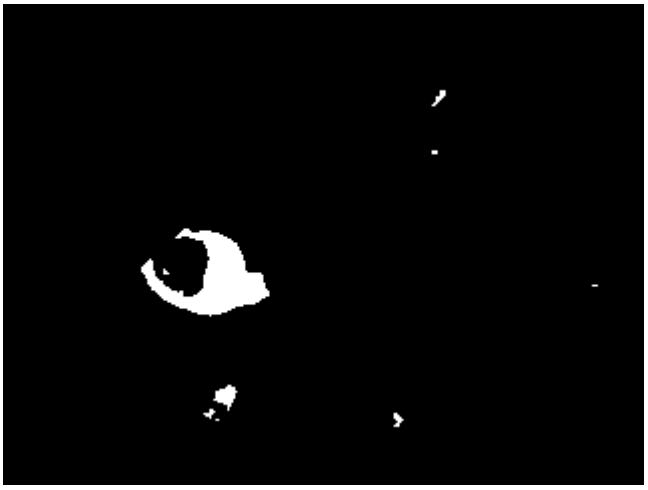

Figura 3-23. Resultado de la binarización de la imagen de la figura 3.14 con umbral 60.

El ruido ha disminuido y por tanto, el umbral de 40 va a sustituirse por este de valor 60. Aunque el resultado sea positivo, también tiene un efecto negativo y es que al subir el umbral y hacer más estricta la binarización, parte de los objetos de interés pueden eliminarse o reducir su tamaño. Visto esto, se concluye que el umbral se elige mediante la prueba y el error sin comprometer en exceso a los objetos que se van a analizar, incluso si hace falta dejar ruido en la imagen ya que éste podrá eliminarse parcialmente mediante las siguientes operaciones.

3.5 Operaciones morfológicas

Si se mira con detalle las imágenes resultantes de la binarización, puede verse que existe ruido y que los objetos de interés, en este caso personas, pueden no estar formados por una única área blanca, sino por varias que pueden clasificarse de manera incorrecta. Por ejemplo en la figura 3.22 la pierna de la persona está separada del resto mientras que lo que se quiere es que esta persona esté formada por un único objeto. Para intentar mejorar los resultados y solucionar tanto el ruido como la separación en más de un área de los objetos de interés, se aplicarán dos operaciones morfológicas: erosión y dilatación.

3.5.1 Erosión

La erosión consiste en examinar cada píxel de la imagen y si alguno de los píxeles de la vecindad de éste tiene valor 0, cambiarlo de blanco a negro, si lo fuera. Las vecindades utilizadas normalmente incluyen los ocho píxeles que rodean al examinado, aunque también pueden usarse más o menos píxeles. En este caso, para obtener un buen resultado, se va a aplicar la erosión con una vecindad de ocho píxeles. El algoritmo 3.2 muestra lo anteriormente explicado en forma de pseudocódigo.

Algoritmo 3.2 Algoritmo básico de erosión morfológica

```
for j← filas de la vecindad .. (número de filas de la imagen - filas de la vecindad), +1
  for k← columnas de la vecindad .. (número de columnas de la imagen - columnas de la vecindad), +1
    0 ← sum
    for m← - filas de la vecindad .. filas de la vecindad, +1
      for n← - columnas de la vecindad .. columnas de la vecindad, +1
        if pixel de la imagen perteneciente la vecindad==0
          sum+1 ← sum
        end if
      end for
    end for
    if sum < (filas de la vecindad)*(columnas de la vecindad)
      Poner pixel blanco
    else
      Poner pixel negro
    end if
  end for
end for
```

La erosión se va a encargar de limpiar la imagen de ruido pequeño que no está conectado a ningún objeto, así como reducir el tamaño de ruido mayor. Al igual que con el umbral de la binarización, hay que tener cuidado con el tamaño de la vecindad a escoger. Si es muy grande, puede hacer que los objetos de interés se vean perjudicados.

Para este proyecto, inicialmente se diseñó un algoritmo propio pero requería de mucho computo por lo que finalmente se decidió optar por la función "*erode*" incluida en OpenCV. Esta consigue los mismos resultados pero con mucho menos coste computacional.

En las figuras 3.24 y 3.25 puede verse el resultado de aplicar erosión sobre las imágenes binarias resultantes del anterior bloque y para poder compararlas se muestran las imágenes antes de erosionar en las figuras 3.26 y 3.27:

Figura 3-24. Resultado de la erosión de la imagen de la figura 3.22.

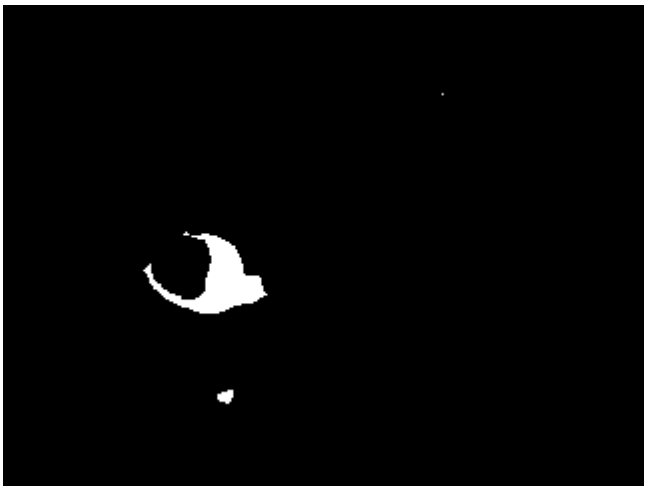

Figura 3-25. Resultado de la erosión de la imagen de la figura 3.23.

Figura 3-26. Resultado de la binarización de la imagen de la figura 3.14 con umbral 60.

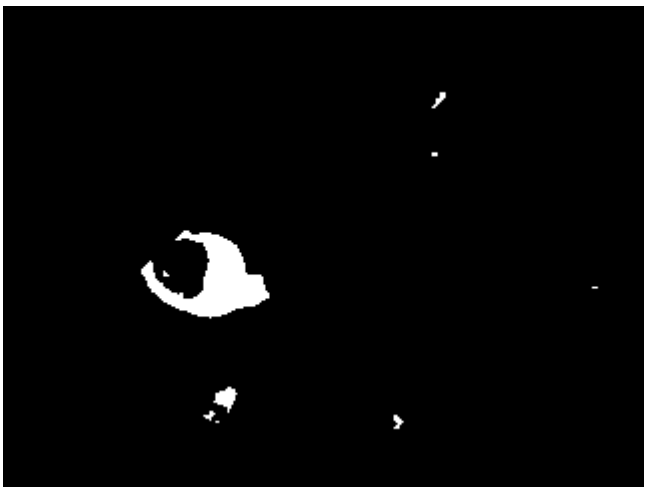

Figura 3-27. Resultado de la binarización de la imagen de la figura 3.15 con umbral 60.

Al comparar las imágenes, se observa claramente la mejoría. En los fotogramas erosionados no existe casi nada de ruido en comparación con las imágenes previas. También puede verse que los objetos de interés han reducido su tamaño. Esto va a mejorarse con la siguiente operación, la dilatación.

3.5.2 Dilatación

La dilatación es la operación contraria a la erosión. En este caso, se examina cada pixel de la imagen y si todos los de su vecindad fueran 1, se cambiaría el pixel examinado de 0 a 1 si era 0 inicialmente. El algoritmo 3.3 describe el funcionamiento de la dilatación.

Algoritmo 3.3 Algoritmo básico de dilatación morfológica

```
for j← filas de la vecindad .. (número de filas de la imagen - filas de la vecindad), +1
  for k← columnas de la vecindad .. (número de columnas de la imagen - columnas de la vecindad), +1
     0 ← sum
     for m← - filas de la vecindad .. filas de la vecindad, +1
        for n← - columnas de la vecindad .. columnas de la vecindad, +1
           if pixel de la imagen perteneciente la vecindad==0
              sum+1 ← sum
           end if
        end for
     end for
     if sum > 0
        Poner pixel blanco
     else
        Poner pixel negro
     end if
  end for
end for
```

De nuevo, la vecindad que se va a elegir inicialmente va a ser la misma que en la erosión, los ocho píxeles que rodean al examinado. En las figuras 3.28 y 3.29, se observan los resultados:

Figura 3-28. Resultado de la dilatación de la imagen de la figura 3.24.

Figura 3-29. Resultado de la dilatación de la imagen de la figura 3.25.

Examinando los resultados, se ve que las imágenes tampoco han cambiado en exceso. Si se aplicara una vecindad mayor, un cuadrado de 8x8, se obtiene lo mostrado en las figuras 3.30 y 3.31:

Figura 3-30. Resultado de la dilatación de la imagen de la figura 3.24 con una vecindad mayor.

Figura 3-31. Resultado de la dilatación de la imagen de la figura 3.25 con una vecindad mayor.

Las imágenes ahora si han cambiado mucho respecto de las iniciales. Se ve de forma clara que las figuras se ven más homogéneas. Si el resultado no fuera bueno, se podría dilatar una segunda vez consecutiva para hacer a las personas más homogéneas aún. Esto sería recomendable debido a que ciertas personas son de difícil reconocimiento y son representadas como varias regiones juntas en vez de una y lo que interesa es que se vea como una única región.

En las figuras 3.32 y 3.33, mostradas abajo, pueden verse los resultados finales de aplicar dos veces dilatación:

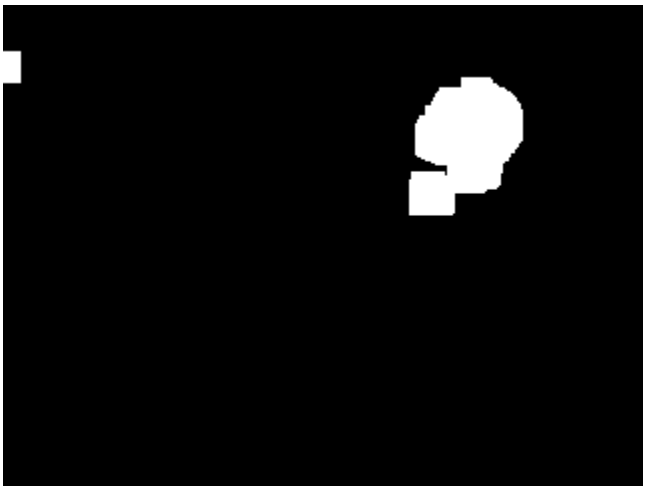

Figura 3-32. Resultado de dilatar dos veces la imagen de la figura 3.24.

Figura 3-33. Resultado de dilatar dos veces la imagen de la figura 3.25.

Los objetos se ven mucho más gruesos, pero a la vez más unidos. Si ahora se suman ambas imágenes de ambos canales S y V, se obtiene la imagen objetivo que se utilizará en el seguimiento (Figura 3.34).

Figura 3-34. Resultado de juntar las imágenes de las figuras 3.32 y 3.33 en una sola.

El resultado es bueno aunque exista ruido todavía y al dilatar dos veces, provoca que este ruido se amplifique. A partir de esta imagen, se realizará el seguimiento de los objetos en movimientos.

3.6 Conclusiones

Una vez terminado este apartado se concluye que los resultados son buenos pero existe un error que va arrastrándose de una operación a otra. Esto implica que si en la sustracción del fondo existe ruido, es muy posible que éste se reduzca pero que parte de él no se elimine en la binarización o en las operaciones siguientes.

Otro punto crítico es el umbral a elegir para los canales en la binarización. Si este umbral es demasiado relajado, esto provocará la no eliminación de ruido, mientras que si es demasiado estricto, puede comprometer a los objetos a analizar dividiéndolos en varios objetos de menor tamaño, provocando que al erosionar, partes del objeto corran el riesgo de ser eliminados.

Por tanto, se concluye que un procesado perfecto es difícilmente posible de realizar, aunque mediante la realización de bastantes experimentos se puede encontrar buenos resultados cercanos a los óptimos. Por último, como el ruido es difícil de eliminar por completo, puede llevar a errores en el seguimiento, por tanto hay que tenerlo bastante en cuenta a la hora de realizar dichos algoritmos.

4 SEGUIMIENTO

El experimentador que no sabe lo que está buscando
no comprenderá lo que encuentra.

CLAUDE BERNARD

En este capítulo se explica el funcionamiento del otro gran bloque del proyecto, el seguimiento de objetos. Tal y como se explica en el tema anterior, el bloque de procesamiento toma la imagen del vídeo. A partir de esta, se obtiene una imagen binaria con los objetos en movimiento representados en blanco y el resto en negro. Con esta imagen será suficiente para realizar el algoritmo completo de seguimiento.

El problema del seguimiento de múltiples objetos puede ser dividido en dos partes principales:

- Detección de regiones en movimiento en cada imagen perteneciente al vídeo.
- La asociación de estas regiones pertenecientes a distintas imágenes a un mismo objeto del cual se está realizando el seguimiento.

La primera parte se resuelve de manera simple ya que se está detectando en cada imagen todos los objetos que han sido previamente sustraídos mediante el uso del fondo. En cada imagen se detectarán nuevas regiones.

Sin embargo, la dificultad de este problema radica en la asociación. El programa tiene que ser capaz de entender que la región detectada en la imagen correspondiente al instante de tiempo anterior y la región detectada en la imagen actual son un mismo objeto que se encuentra en movimiento.

En general, el funcionamiento del programa se basa en la creación de unos bloques de información llamados *"tracks"*. Estos tracks van a contener toda la información correspondiente a los objetos de los que se está realizando el seguimiento. En cada iteración del bucle general, o sea, en cada imagen del vídeo analizada, dichos tracks se actualizarán con nueva información debido a nuevas asociaciones con regiones detectadas en las nuevas imágenes.

En los siguientes apartados se explican los pasos que sigue el algoritmo para el seguimiento de los objetos en movimiento, tanto para uno como para múltiples objetos simultáneamente. También se explican otros casos como la creación de nuevos tracks o la eliminación de tracks que han dejado de asociarse. Por último, se explican problemas que pueden llevar a errores de seguimiento y cómo se tratan de solucionar.

4.1 Esquema general

Como ya se ha comentado en la introducción de este capítulo, en el problema de seguimiento existen dos partes, la detección de las regiones en las imágenes y la asociación de estas regiones con los llamados tracks, encargados del seguimiento. A continuación se detallan estas partes, recurriendo a algún ejemplo si fuera necesario.

4.1.1 Detección de regiones

Desde el bloque de procesamiento se entrega el único dato necesario para este bloque, una imagen binaria resultante de aplicar sobre ella una sustracción del fondo, segmentación y las operaciones morfológicas, erosión y dilatación. Con esto, se tiene una imagen en la que, sin contar el ruido que aparece, sólo están representados los objetos en movimiento a los cuales se les va a realizar el seguimiento.

Para la realización del seguimiento, hace falta convertir de alguna manera la información visual dada por las imágenes en datos numéricos. Conseguir esto puede hacerse de distintas maneras. En este proyecto se consigue dicho objetivo mediante la detección de contornos en las imágenes usando la función *"findContours"* disponible en la librería OpenCV. Esta función, a partir de una imagen binaria, detecta todas las regiones en dicha imagen. Entrega para cada contorno un vector que contiene todos los puntos (en coordenadas cartesianas X e Y) que forman el contorno de la región detectada. Cabe mencionar que las coordenadas en las imágenes tienen su origen en la esquina superior izquierda de la imagen.

Una vez que se tiene el vector con los vectores para cada contorno, se puede calcular toda la información de las regiones que ha de saberse para realizar un buen seguimiento. Lo primero que hay que calcular son los centroides de las regiones, tanto su coordenada X como su coordenada Y. Las fórmulas de las componentes X e Y de los centroides de una región son las siguientes:

$$\bar{x} = \frac{m_{10}}{m_{00}},\ \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.1}$$

siendo m_{00}, m_{01} y m_{10} los momentos espaciales de la región, calculados todos ellos mediante el uso de la siguiente expresión:

$$m_{pq} = \sum_{i=1}^{M} \sum_{j=1}^{N} i^p j^q f_{ij} \tag{4.2}$$

en la que M es el número total de componentes X de los puntos que hay en un vector de contorno, N el número total de componentes Y de los puntos que hay en el vector y f_{ij} el valor de la intensidad del punto o pixel examinado.

Aparte de las coordenadas de los centroides, otro dato necesario que se extrae de las regiones es su dimensión. Mediante el uso de la función de OpenCV *"boundingRect"* se obtiene un rectángulo delimitante o recuadro para cada región. El ancho y alto de dicho rectángulo van a usarse también como información que va a incluirse en el track. El valor de estas dos características viene dado en número de píxeles.

A continuación se muestra un ejemplo de la detección de regiones en la figura 4.1. Puede verse el centroide de cada región mostrado mediante un punto azul mientras que el recuadro viene

representado en color rojo. La información que se obtiene de las regiones es la siguiente:

- Región de la izquierda:
  - Coordenada X del centroide: 156
  - Coordenada Y del centroide: 172
  - Ancho: 69
  - Alto: 55
- Región de la derecha:
  - Coordenada X del centroide: 441
  - Coordenada Y del centroide: 97
  - Ancho: 64
  - Alto:62

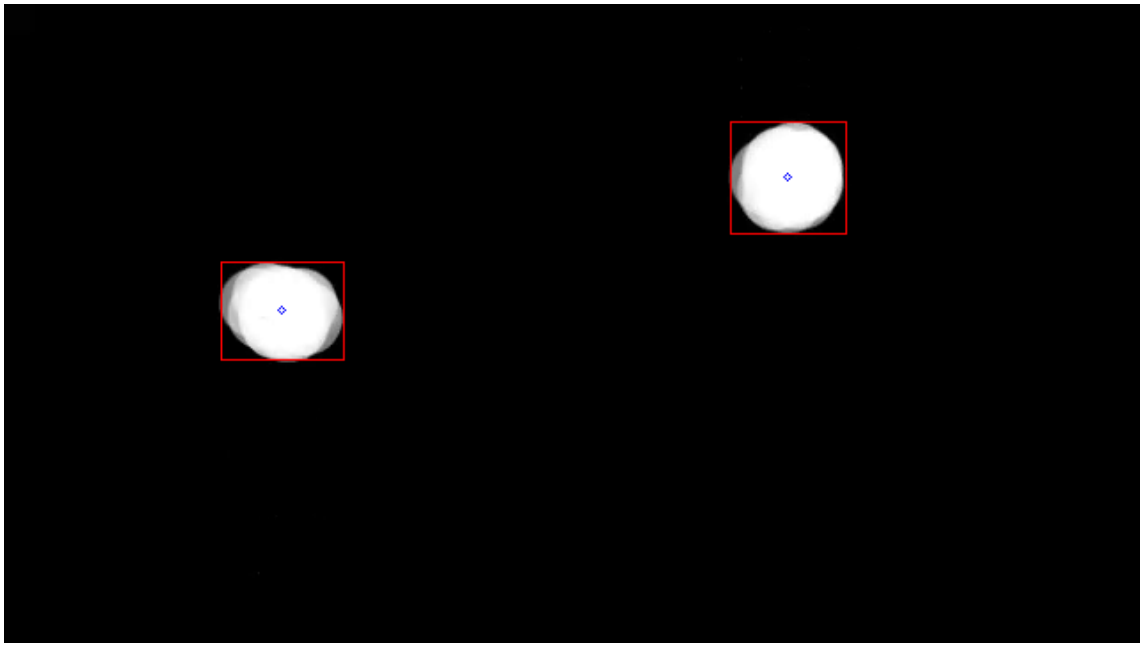

Figura 4-1. Detección de dos regiones en una imagen.

4.1.2 Seguimiento

La siguiente parte de este bloque consiste en el seguimiento en sí, que a partir de los datos de las regiones detectadas se van a crear tracks para cada objeto. Cada uno de estos se va a ir asociando en cada imagen a las regiones detectadas que mejor cumplan las condiciones de asociación.

Todo el algoritmo de seguimiento se basa en la creación de una matriz de tracks en la que en cada columna se aloja a un track y en las filas se establecen las distintas características de cada track. Dichas características serán las siguientes:

- Coordenada X del centroide de la región asociada al track
- Coordenada Y del centroide de la región asociada al track
- Ancho del recuadro que contiene la región asociada al track.
- Alto del recuadro que contiene la región asociada al track.
- Velocidad X: Mide la velocidad horizontal del objeto al que se le está realizando el seguimiento. Sus unidades son píxeles/imagen y su expresión es la siguiente:

$$v_x = x_{región} - x_{track} \tag{4.3}$$

- Velocidad Y: Igual que la anterior, pero en este caso, se trata de velocidad vertical.

$$v_y = y_{región} - y_{track} \tag{4.4}$$

- Visibilidad: Informa de si el track se puede ver o si ya no está en la zona de visión de la cámara.
- Edad: Informa sobre el número de imágenes en las que el track se actualiza desde que se creó.
- Contador de inactividad: Cuenta el número de imágenes en las que el track deja de estar visible.
- Señal de eliminación: Señal lógica que se activa cuando el contador de inactividad llega a 3. Cuando se activa se elimina el track, creando un hueco disponible en la matriz de tracks.

En la tabla 4.1 se representa un ejemplo de la matriz de tracks con todas sus características.

Tabla 4–1. Ejemplo de matriz de tracks

Info | Tracks	T_1	T_2	T_3	T_4	*Hueco*	*Hueco*	...
Posición X	223	154	422	349			
Posición Y	156	256	358	222			
Ancho	64	58	56	68			
Alto	69	67	70	63			
Velocidad X	2	−1	3	5			
Velocidad Y	−40	43	35	−37			
Visibilidad	1	1	0	1			

Edad	**5**	7	**10**	**3**			
Inactividad	**0**	**0**	**3**	**0**			
Eliminación	**0**	**0**	**1**	**0**			

Observando la matriz, cuando un track se aloja en una columna, nunca va a cambiar de posición salvo cuando se elimina, que permite que otro track nuevo acceda a esta posición. Esta matriz se actualiza con cada nueva imagen que pasa, cambiando la información de cada track, así como alojando a nuevos en los huecos disponibles o eliminando a antiguos.

La estructura general del algoritmo de seguimiento consta de tres fases bien definidas: predicción, asociación y actualización. A continuación se detallan cada una de estas.

4.1.2.1 Predicción

La fase de predicción en el seguimiento se usa para estimar dónde va a estar el track en la imagen actual, a partir de la información de dicho track en las imágenes anteriores. Esta estimación es necesaria para la siguiente fase: la asociación, en la que se va a relacionar las regiones detectadas con las predicciones de los tracks.

En este proyecto se analizan zonas de paso de personas mediante el uso de una cámara dispuesta en una posición fija. Esto quiere decir, que al tratarse de una misma zona de paso en todo el experimento, las personas entran y salen siempre por los mismos sitios y por tanto sus desplazamientos en la zona de visión de la cámara son lineales o aproximadamente lineales. Sabiendo esto, se puede suponer a la hora de realizar el algoritmo que las personas se van a desplazar de forma lineal siempre. Pudiéndose así calcular la predicción mediante una simple suma y no usando otros métodos más complejos como podría ser el filtro de Kalman.

Como se ha mencionado antes, las posiciones predichas se calculan a partir de la información del track aportada en imágenes anteriores. Concretamente, se calculan mediante las posiciones del centroide en la imagen anterior, tanto en X como en Y, y mediante las velocidades, también en X e Y, de estos tracks. Dichas posiciones estimadas tienen las siguientes expresiones:

$$x_{predicho} = x_{track} + v_x \tag{4.5}$$

$$y_{predicho} = y_{track} + v_y \tag{4.6}$$

A continuación, se muestra un ejemplo de cómo actúa la predicción en el seguimiento de un objeto a lo largo de varias imágenes. Si se observa la imagen representada en la figura 4.2, hay una única región detectada en el entorno (marcada en rojo) para la cual se crea un nuevo track. En el recuadro morado viene dada la información de interés del track para la predicción.

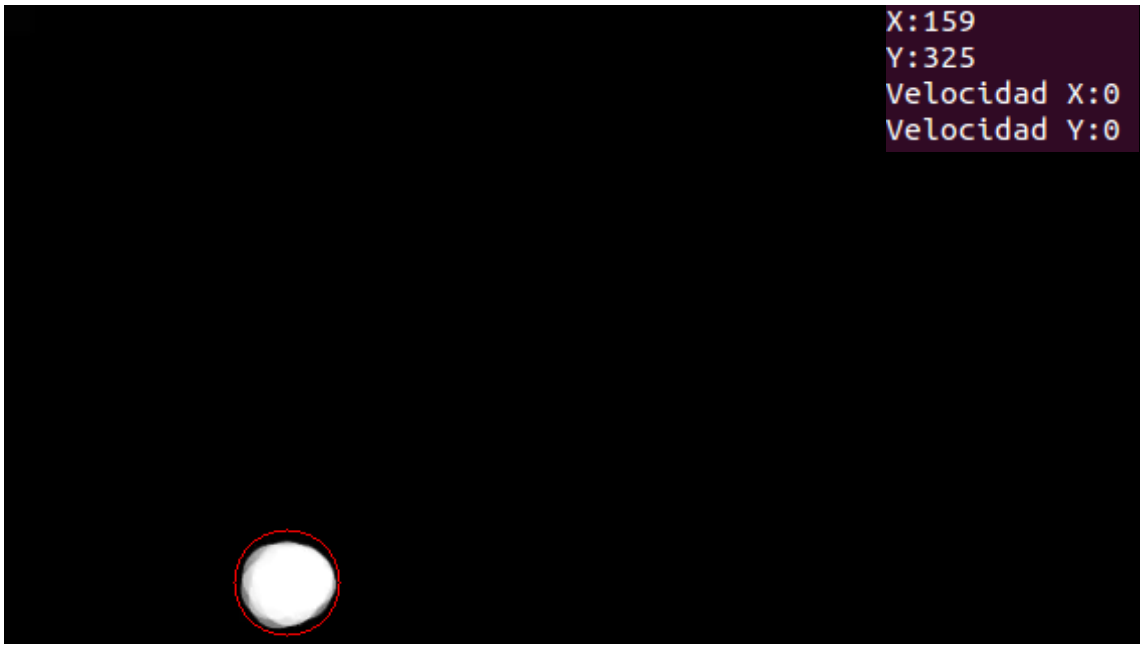

Figura 4-2. Región detectada en la imagen para la cual se crea un nuevo track.

Al no conocer nada sobre este objeto en imágenes anteriores, las velocidades se ponen inicialmente a 0. En la figura 4.3 se ve la siguiente imagen del vídeo, en la que el track se ha actualizado asociándose a una nueva región y al contar con información previa, se puede calcular la velocidad. Sin embargo, la posición predicha (marcada en verde) al contar con una velocidad previa de valor 0, va a situarse justo donde estaba el track en el instante de tiempo anterior, mostrado en la figura 4.2.

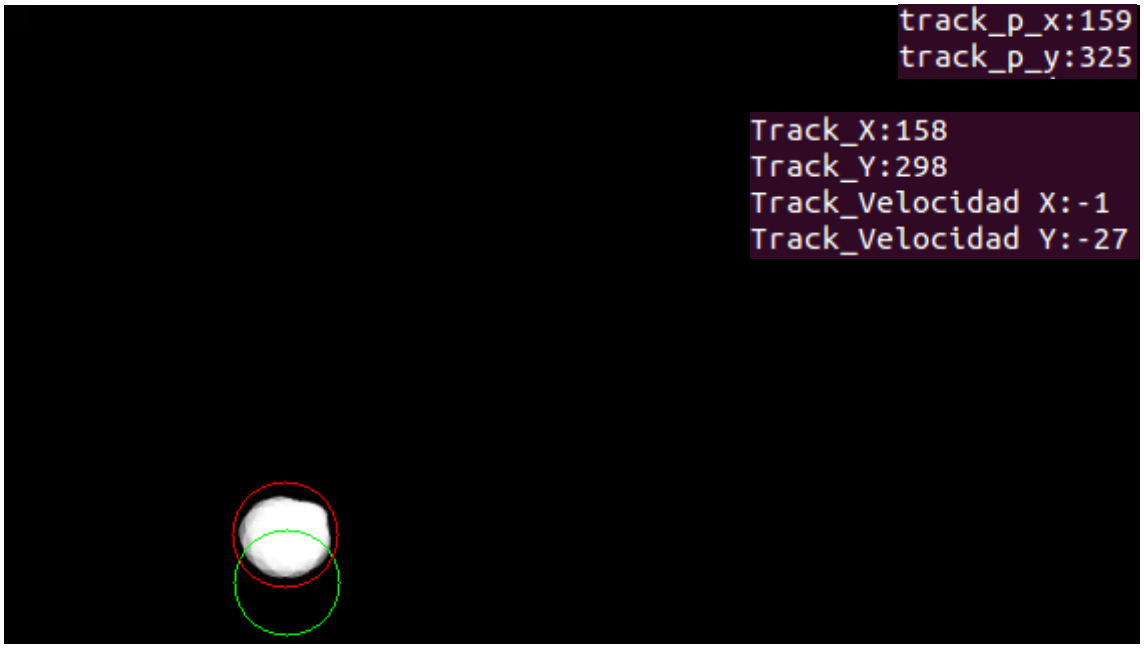

Figura 4-3. Track creado y asociado en una nueva imagen con su predicción correspondiente.

Hasta la tercera imagen desde la creación del track, vista en la figura 4.4, no se realiza una predicción completa, estimando correctamente donde podría estar el track en la imagen actual en función de qué velocidad tenía y dónde estaba en la imagen anterior. Si el movimiento del objeto es bastante lineal, mientras más imágenes pasen, mejor será la predicción. Si se da el caso de que dicho objeto no se mueve de forma lineal, estas predicciones no serían lo suficientemente precisas.

Figura 4-4. Track asociado con su predicción correspondiente.

4.1.2.2 Asociación

La fase de asociación es quizás la más importante del algoritmo de seguimiento. Gracias a ella, el programa es capaz de reconocer que regiones detectadas en imágenes distintas son en realidad, un mismo objeto en movimiento.

En esta fase, cada track se relaciona con cada región, para ver cuál es la más acorde con dicho track. Esto se hace usando las posiciones predichas de los centroides de los tracks y las posiciones de los centroides de las regiones detectadas en la imagen actual. Mediante el uso del criterio de mínima distancia, se asociarán track y región. Por tanto, se usa la siguiente expresión:

$$d = \sqrt{\left(x_{track\ predicho} - x_{región}\right)^2 + \left(y_{track\ predicho} - y_{región}\right)^2} \qquad (4.7)$$

Se calcula una distancia para cada track con cada región y todas estas se almacenan en un vector para cada track y todos de tamaño el número de regiones que hay en la imagen. De todas las distancias en el vector, asociadas cada una a una región, habrá una que sea mínima. Por tanto la región que posea la mínima distancia en el vector de dicho track será la región que se asocie a este.

Al igual que se asocian regiones a tracks, hay que también asociar tracks a regiones de una manera análoga a la explicada. Para así crear la matriz de asociación y pasar a la clasificación de tracks según sus características. Esto se explica detalladamente más adelante en el tema.

A continuación se muestra un ejemplo de cómo funciona la fase de asociación. Se tiene una imagen,

mostrada en la figura 4.5, en la que hay dos regiones detectadas (marcadas en rojo) y dos tracks activos cuyas predicciones vienen representadas en verde en la imagen. Los datos de las posiciones de las regiones y de las posiciones predichas de los tracks son los siguientes:

- Región de la izquierda:
  - Posición X: 154
  - Posición Y: 123
- Región de la derecha:
  - Posición X: 435
  - Posición Y: 162
- Track 1:
  - Predicción X: 155
  - Predicción Y: 122
- Track 2:
  - Predicción X: 438
  - Predicción Y: 155

Al calcular la distancia de cada track a cada región, se obtiene:

- Track 1:
  - Distancia a región de la izquierda: **1.414**
  - Distancia a la región de la derecha: 282.843
- Track 2:
  - Distancia a región de la izquierda: 285.797
  - Distancia a la región de la derecha: **7.616**

Por tanto, viendo las distancias mínimas, el track 1 se asocia con la región de la izquierda y el track 2 con la región de la derecha.

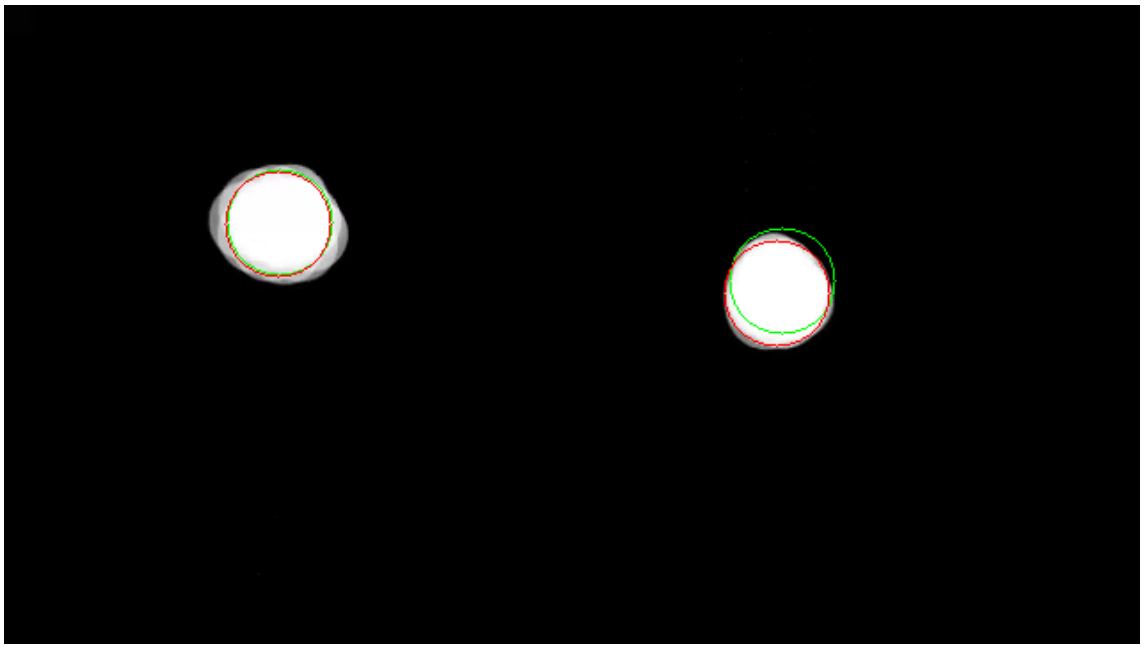

Figura 4-5. Dos tracks asociados usados en el ejemplo para cálculo de las distancias.

4.1.2.2 Actualización

Esta es la última fase del algoritmo de seguimiento. En ella se modifica toda la información del track en función de cómo se clasifica este en la fase de asociación. Los tracks pueden clasificarse como tracks que se asocian, tracks que dejan de asociarse, tracks nuevos y otros casos que se explican en detalle a lo largo del capítulo.

Para cada tipo de clasificación hay una actualización diferente. Por ejemplo para la actualización más común, es decir, que el track se asocia con una región, la actualización se basa en darle a este las posiciones y los tamaños de la región y calcular la velocidad mediante las expresiones anteriormente explicadas:

$$v_x = x_{región} - x_{track} \tag{4.8}$$

$$v_y = y_{región} - y_{track} \tag{4.9}$$

El resto de tipos de actualizaciones se explican en las siguientes subsecciones del capítulo.

También en esta fase se realiza la gestión de memoria en lo que respecta a la matriz de tracks. Cuando un track acaba, se elimina y por tanto deja un hueco reutilizable por otro track en la matriz. La gestión de memoria consistirá en hacer que a la hora de crear nuevos tracks, estos se alojen en los huecos libres si se da el caso de que hay y sino, que se alojen en la primera posición libre después del último track.

4.2 Asociación de datos

4.2.1 Matriz de asociación

Como se explica en la subsección anterior, la asociación de regiones y tracks se basa en el criterio de la mínima distancia. Se relacionan todos los tracks con todas las regiones así como todas las regiones con todos los tracks mediante el uso de la siguiente expresión:

$$d = \sqrt{\left(x_{track\ predicho} - x_{región}\right)^2 + \left(y_{track\ predicho} - y_{región}\right)^2} \tag{4.10}$$

Cada distancia obtenida para cada track se almacena en un vector. De este vector, se busca la mínima distancia y se almacena de nuevo en otro vector de mínimas distancias en las que cada track tendrá asociado una de estas distancias. El seudocódigo de este cálculo viene dado en el algoritmo 4.1.

Para las regiones se hace exactamente igual, es decir, se tendrá un segundo vector con todas las mínimas distancias asociadas a cada región.

Algoritmo 4.1 Algoritmo para obtención de distancias para tracks

```
for s← 0 ..  número de tracks, +1
   for i← 0 .. número de regiones en la imagen, +1
            Calcular d
            Almacenar d en vector distancias[i]
   end for
   Obtener mínima distancia de vector distancias
   Almacenar mínima distancia en vector mindist_tracks[s]
end for
```

Con este algoritmo se tendrá una distancia mínima asociada a cada track existente. Para la obtención del vector de mínimas distancias asociadas a cada región, se realiza un algoritmo muy similar pero en este caso, se invierten el orden de los bucles de condición dándole a las regiones la prioridad. Puede verse en el algoritmo 4.2.

Algoritmo 4.2 Algoritmo para obtención de distancias para regiones

```
for i← 0 .. número de regiones en la imagen, +1
    for s← 0 ..  número de tracks, +1
            Calcular d
            Almacenar d en vector distancias[s]
   end for
   Obtener mínima distancia de vector distancias
   Almacenar mínima distancia en vector mindist_regiones[i]
end for
```

Una vez se tienen los dos vectores de mínimas distancias, ya puede generarse la matriz de asociación. Esta va a tener tantas filas como número de tracks existentes y tantas columnas como el número de regiones que haya en la imagen. Si se recorre cada región para cada track, se restan

sus distancias asociadas y el resultado es 0, significa que existe asociación, por lo que en esta casilla se pone un 1. Si el resultado de la resta no es 0, se pone 0 en la casilla correspondiente, lo que quiere decir que no hay asociación entre ese track y esa región. Con esto, se genera una matriz de asociación para cada imagen del vídeo. A continuación se muestra un ejemplo de una matriz de asociación en la tabla 4.2:

Tabla 4–2. Ejemplo de matriz de asociación

Regiones / Tracks	R_1	R_2	R_3
T_1	**0**	**1**	**0**
T_2	**0**	**0**	**1**
T_3	**1**	**0**	**0**

En la matriz del ejemplo existen tres tracks y se han detectado tres regiones. El track 1 se ha asociado con la región 2, el track 2 con la región 3 y el track 3 con la región 1. Existen otros casos, como por ejemplo que ningún track se asocie, que son estudiados a continuación.

Cabe mencionar que las regiones detectadas por la función de OpenCV *"findContours"*, no siempre vienen dadas en el mismo orden. Esto es porque en cada imagen tan solo se reconocen regiones que en principio, nada tienen que ver con las regiones detectadas en las imágenes anteriores. Sin embargo, los tracks, una vez creados, si se mantienen en la misma posición hasta su eliminación.

A continuación se muestra el pseudocódigo de lo explicado anteriormente en el algoritmo 4.3.

Algoritmo 4.3 Algoritmo para creación de la matriz de asociación

```
for s← 0 ..  número de tracks, +1
   for i← 0 .. número de regiones en la imagen, +1
            if (mindist_tracks[s] – mindist_regiones[i])==0
                matriz_asociacion[s][i]=1
            else
                matriz_asociacion[s][i]=0
            end if
   end for
end for
```

4.2.2 Clasificación de tracks

Una vez que se tiene la matriz de asociación, el siguiente paso es interpretar los resultados dados por esta. En este apartado se van a estudiar tres casos diferentes y son los siguientes: existe asociación entre track y región, el track no se asocia y una región no se asocia. Existen más casos que se verán más adelante en el capítulo o que salen del desarrollo de este proyecto.

4.2.2.1 Track y región se asocian

Este es el caso más común de todos y el más simple. Cuando resulta que al restar la mínima distancia asociada al track con la mínima distancia asociada a la región da 0, es decir, que ambas distancias son la misma, se pondrá un 1 en su casilla correspondiente en la matriz.

La interpretación de la matriz y la clasificación de los tracks según los resultados de esta, se hacen en un nuevo algoritmo en el que se van a contemplar todos los casos de tracks estudiados en el proyecto.

Para el caso de este subapartado, asociación de track y región, se crea una nueva matriz que va a tener tantas filas como tracks se asocien en la imagen y dos columnas, una para tracks y otra para regiones. Por tanto, esta matriz contendrá los índices de los tracks asociados en la imagen y sus regiones correspondientes. Estos datos pertenecientes a esta matriz de asignación se usarán en la fase de actualización, ya que cada caso de track clasificado se actualiza de una manera concreta.

En el algoritmo, se va a recorrer la matriz de asociación completa y cada vez que se encuentre un 1, se añade a la matriz de asignación explicada previamente una nueva fila. Esto quiere decir que cada vez que se asocien un track con una región, aumentará el tamaño de la matriz de asignación.

A continuación, se muestra la matriz de asignación (tabla 4.3) correspondiente a la matriz de asociación mostrada en el apartado anterior:

Tabla 4–3. Ejemplo de matriz de asignación

Track	*Región*
1	2
2	3
3	1

Como puede verse, dicha matriz contiene los índices de los tres tracks en ella y de las tres regiones, asociadas. El track 1 con la región 2, el track 2 con la región 3 y el track 3 con la región 1.

Una vez que se tienen los índices de los tracks asociados almacenados, el siguiente paso para estos tracks es la actualización. Iterando mediante el uso de un bucle *"for"* desde cero hasta el tamaño de tracks asociados (número de filas de la matriz de asignación) para cada track, se actualiza la siguiente información:

- Velocidad X: Será la resta de la coordenada X del centroide de la región (con índice correspondiente al su track asociado) y la coordenada X del centroide del track en la imagen anterior.
- Velocidad Y: Será la resta de la coordenada Y del centroide de la región (con índice correspondiente a su track asociado) y la coordenada Y del centroide del track en la imagen anterior.
- Coordenada X: Pasa a valer lo que vale la coordenada X del centroide de la región asociada.
- Coordenada Y: Pasa a valer lo que vale la coordenada Y del centroide de la región asociada.
- Ancho: Exactamente igual que los anteriores, pasa a valer lo que el ancho de la región asociada.
- Alto: De nuevo, igual que los anteriores.
- Visibilidad: Se mantiene a 1 ya que el objeto está visible.
- Edad: Se aumenta en 1 su valor.
- Contador de inactividad: Se mantiene a 0.
- Señal de eliminación: Permanece inactiva.

Por último se muestra en el algoritmo 4.4 el seudocódigo básico de la creación de la matriz de asignación descrito en este subapartado. Cabe decir que al ir añadiendo casos de clasificación de tracks, este algoritmo se irá complicando, por tanto el mostrado abajo se corresponde a una versión simplificada:

Algoritmo 4.4 Algoritmo para clasificación de tracks. Primera versión.

```
0 ← int
for s← 0 ..  número de tracks, +1
   0 ← contador
   for i← 0 .. número de regiones en la imagen, +1
            if (asociacion[s][i])==1
                contador + 1 ← contador
               if contador==1
                0 ←  contador
                matriz_asignación[int][0]=s
                matriz_asignación[int][1]=i
                int + 1 ← int
               end if
            end if
   end for
end for
```

4.2.2.2 Track no se asocia

El siguiente caso es el de un track que no se asocia con ninguna región. Esto ocurre cuando el objeto del cual se está realizando el seguimiento, deja de estar visible. En otras palabras, ha salido de la zona de visión de la cámara por lo que ya no se asocia con ninguna región ni va a asociarse con ninguna otra en imágenes futuras.

Observando la matriz de asociación dada en la tabla 4.4, se puede ver que en la fila correspondiente al track 2 no hay ningún 1. Eso quiere decir que dicho track no se ha asociado con ninguna región, entonces, se clasifica como track no asociado.

Tabla 4–4. Ejemplo de matriz de asignación con track no asociado

Regiones / Tracks	R_1	R_2
T_1	**0**	**1**
T_2	**0**	**0**
T_3	**1**	**0**

Cabe mencionar que las personas entran y salen de la zona de paso por dos lugares concretos y bien definidos. Para facilitar la clasificación se creará una condición en la que si el track deja de asociarse y está en uno de los dos límites del vídeo se confirma con seguridad que el objeto del track ha salido del vídeo y no va a asociarse más. Estos límites se definen al principio del programa.

Por tanto, al recorrer la matriz y no encontrar ningún uno en toda una fila y al estar dicho track en los límites del vídeo, o sea en zonas de salida de personas, se clasifica como track no asociado. Al igual que en el anterior caso en el que se generaba una matriz de asignación, en este caso se genera un vector de no asignación que funciona igual que dicha matriz. Cada vez que se clasifique un track como no asociado, se incluye su índice en dicho vector. Este servirá para actualizar posteriormente de forma correcta los tracks que no han sido asociados.

El cumplimiento de las condiciones explicadas en el anterior párrafo clasifican un track que acaba de pasar del tipo asociado al tipo no asociado. Esto hace que en su actualización se cambie su visibilidad de visible (1) a invisible (0). Y aunque dicho track ya no se asocie, no significa que se elimine directamente, sino que su información sigue actualizándose. Por tanto, visto que existe esta posibilidad, hay que tener en cuenta un nuevo caso en la clasificación de los tracks.

En este caso, si el track en la imagen anterior no era visible y de nuevo, si se encuentra en las zonas límites del vídeo, entonces se clasifica también como track no asociado y se incluye su índice en el vector de no asignación.

A continuación se muestra el pseudocódigo actualizado de la clasificación de tracks en el algoritmo 4.5:

Algoritmo 4.5 Algoritmo para clasificación de tracks. Segunda versión.

```
0 ← int
0 ← int2
for s← 0 .. número de tracks, +1
  0 ← contador
  0 ← contador2
  if track era visible
    for i← 0 .. número de regiones en la imagen, +1
          if (asociacion[s][i])==1
             contador + 1 ← contador
            if contador==1
             0 ← contador
              matriz_asignación[int][0]=s
              matriz_asignación[int][1]=i
              int + 1 ← int
            end if
          end if
          if (asociacion[s][i])==0
            contador2 + 1 ← contador2
            if contador2==número de regiones en imagen
              if track está en los límites del vídeo
                vector_no_asignacion[int2]=s
                int2 + 1 ← int2
              end if
            end if
          end if
    end for
  end if
  if track no era visible
     if track está en los límites del vídeo
        vector_no_asignacion[int2]=s
        int2 + 1 ← int2
     end if
  end if
end for
```

Una vez que se tiene completo el vector de no asignación y se ha salido del algoritmo de clasificación de tracks, hay que actualizar dichos tracks de una manera determinada. La información de los tracks identificados en el vector que se actualiza es la siguiente:

- Visibilidad: Pasa de estar visible (valor 1) a estar invisible (valor 0).

- Edad: Se aumenta en 1 su valor.
- Contador de inactividad: Empieza la cuenta del contador de inactividad. Se aumentará en 1 su valor en cada imagen.
- El resto de datos del track permanecen iguales sin modificarse.

Como ya se ha dicho anteriormente, el track no asociado ya no va a asociarse más a ninguna región en imágenes futuras. La única modificación que va a sufrir el track es que se va a eliminar una vez haya pasado tres imágenes inactivo, dejando un hueco libre en la memoria para que otro track nuevo ocupe su lugar. Para esto, en la actualización, se habilita una nueva característica de valor lógico que permanezca siempre a nivel bajo (0) salvo cuando el contador de inactividad llegue a 3, que se pondrá a nivel alto (1). Esto indicará la existencia de un hueco libre apto para ser sobreescrito.

A continuación se muestra un ejemplo gráfico del proceso completo de un track que deja de asociarse y su eliminación.

En la figura 4.6 se observa un track que ha llegado a uno de los límites y que por tanto va a dejar de asociarse (para una mejor y más simple explicación, prestar atención únicamente al objeto de la izquierda, obviar el objeto de la derecha).

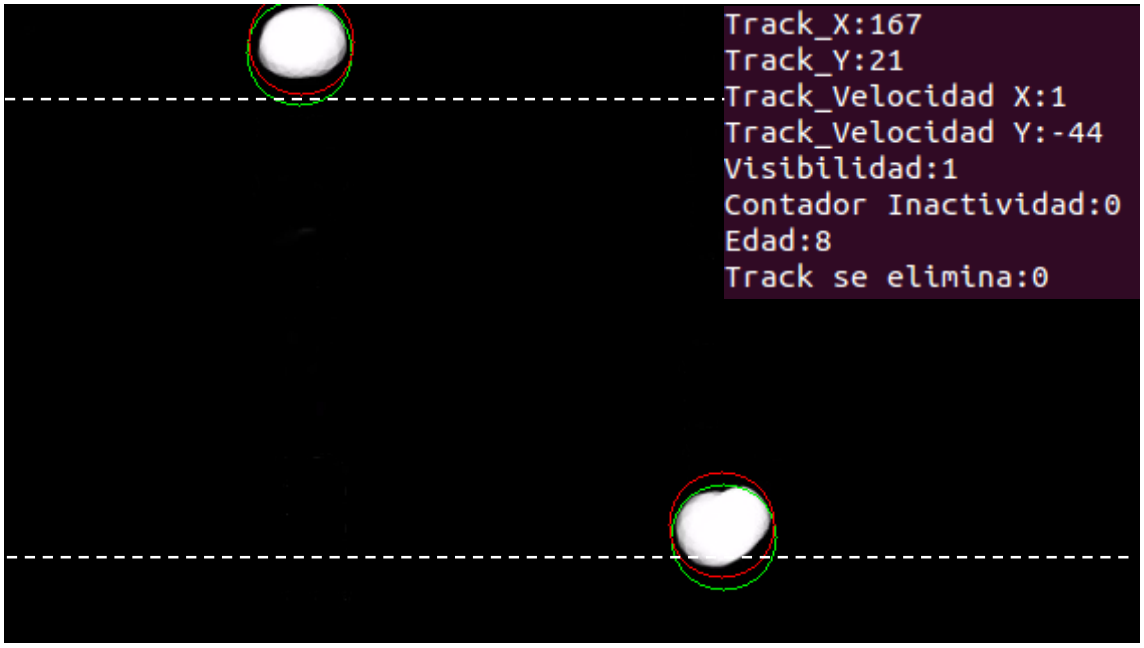

Figura 4-6. Imagen con un track que llega a uno de los límites del vídeo.

El objeto desaparece y por tanto, la visibilidad pasa a valer 0, se inicia el contador de inactividad y la edad aumenta. Puede verse que se continúa prediciendo (círculo verde) aunque ya haya terminado el track (Figura4.7).

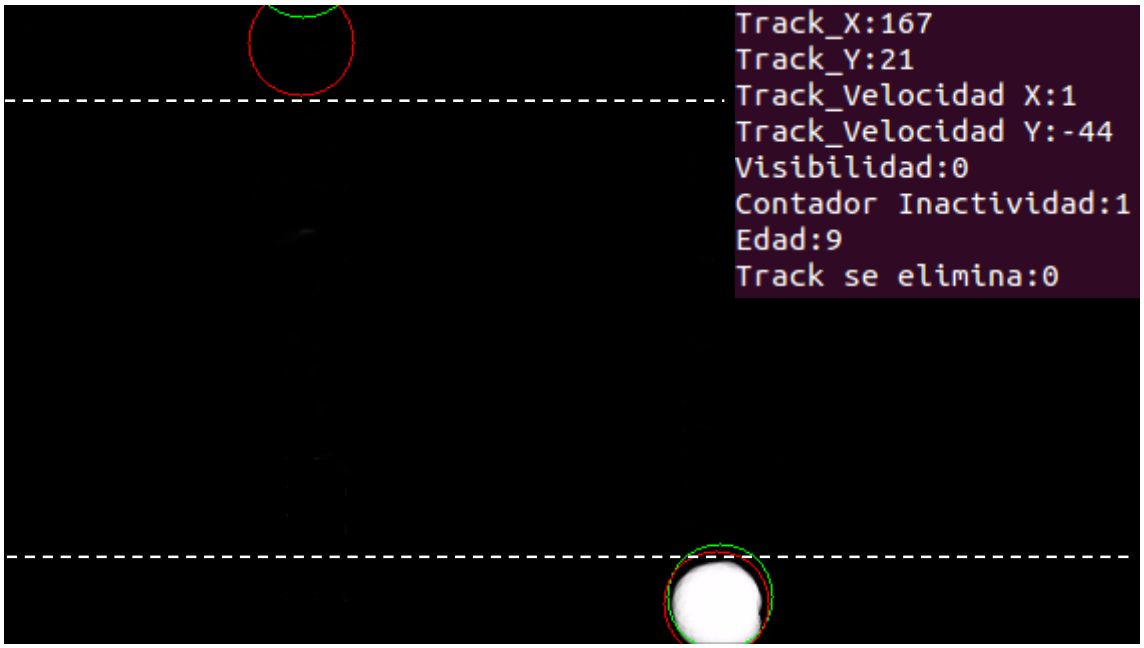

Figura 4-7. Imagen con un track ya acabado.

Después de la imagen mostrada en la figura 4.7 pasará otra imagen más donde aumentarán en uno su valor tanto el contador de inactividad como la edad. Después de esta, en la siguiente imagen, el contador de inactividad alcanzará el valor 3 por lo que se activa la señal de eliminación y por tanto, ya no se representa dicho track mediante el círculo rojo. Esto se ve en la figura 4.8.

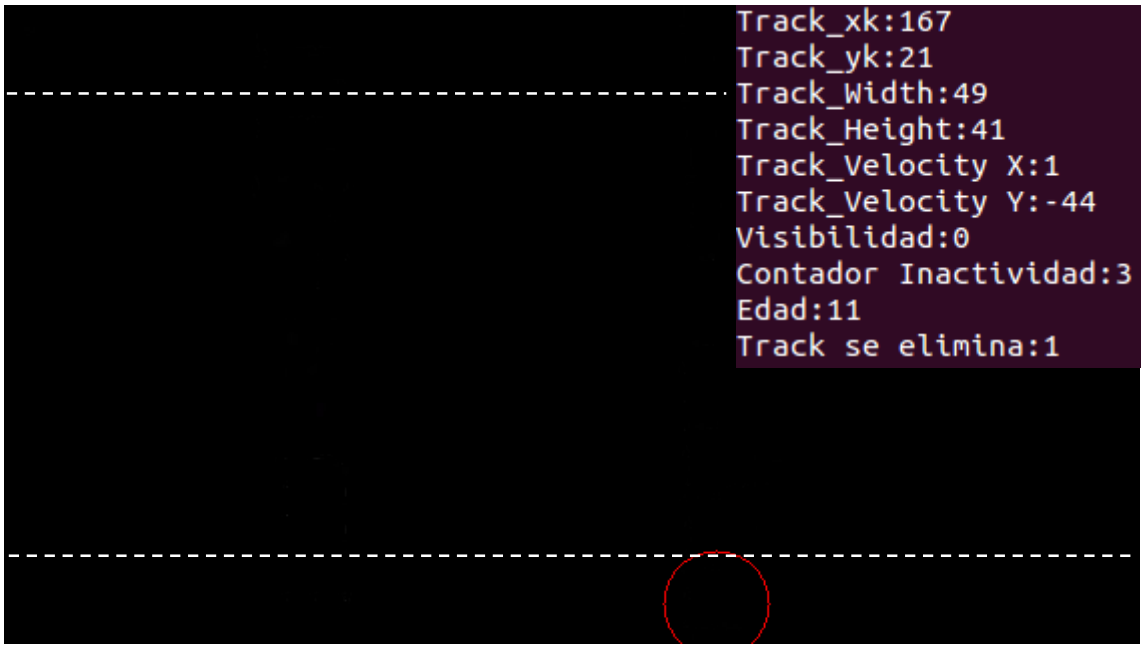

Figura 4-8. Imagen en la que el track examinado se ha eliminado.

4.2.2.3 Creación de nuevos tracks

El último de los casos contemplados en esta subsección del capítulo es el de la creación de tracks nuevos. Al igual que en el anterior caso de tracks que acaban, los objetos que entran en la zona de visión de la cámara, entran por los mismos límites que por donde pueden salir. Entonces, dichos tracks sólo se crearán para las regiones que aparecen en las dos bandas de entradas y salidas. Esto permitirá que no se cree un track para cada región detectada en la imagen, ya que puede tratarse de ruido.

Al igual que en el caso del track que no se asocia, se recorre la matriz de asociación buscando regiones que no tengan ningún 1 en la columna y que estén en las franjas límites del vídeo. Si se observa la tabla 4.5 puede verse que la región 4 no tiene ningún track asociado, por tanto si cumple la condición de los límites, se creará un nuevo track con los datos de esta.

Tabla 4–5. Ejemplo de matriz de asignación con región no asociada

Regiones / Tracks	R_1	R_2	R_3	R_4
T_1	**0**	**1**	**0**	**0**
T_2	**0**	**0**	**1**	**0**
T_3	**1**	**0**	**0**	**0**

A diferencia de los otros dos casos, que se clasifican en un mismo algoritmo, para la clasificación de regiones nuevas, se realiza uno nuevo e independiente. Esto se debe a que ahora el bucle que recorre las columnas es el de mayor prioridad. Para cada columna se recorre cada fila de la matriz (sólo si la región está en los límites) almacenando el número de ceros que encuentra. Si el número de ceros coincide con el número de tracks, es decir, no hay ningún 1 en toda la columna, se almacena el índice de dicha región en un vector de nuevas regiones similar al vector de no asignación. También puede darse que tracks que ya han acabado se asocien erróneamente a nuevas regiones que aparezcan por el mismo límite del vídeo. Si esto ocurre, automáticamente se clasificará la región como una nueva región sin la necesidad de tener toda su columna llena de ceros.

A continuación se muestra el seudocódigo del algoritmo de clasificación de nuevos tracks en el algoritmo 4.6.

Algoritmo 4.6 Algoritmo para clasificación de nuevos tracks.

```
0 ← int
for i← 0 .. número de regiones en la imagen, +1
   0 ← contador
   if región está en los límites del vídeo
```

```
    for s← 0 ..  número de tracks, +1
        if track era visible
            if (asociacion[s][i])==0
                contador + 1 ← contador
                if contador==número de tracks
                 vector_nuevas_regiones[int]=i
                 int + 1 ← int
                end if
            end if
        end if
        if track no era visible
          if (asociacion[s][i])==1
                vector_nuevas_regiones[int]=i
                int + 1 ← int
          end if
          if (asociacion[s][i])==0
                contador + 1 ← contador
                if contador==número de tracks
                 vector_nuevas_regiones[int]=i
                 int + 1 ← int
                end if
          end if
        end if
    end for
  end if
end for
```

Una vez que se sale del algoritmo con el vector generado, se crean los tracks en la fase de actualización. Para ello, se crean actualizando la información de la siguiente manera:

- Coordenada X: Se le da el valor de la coordenada X del centroide de la nueva región.
- Coordenada Y: Se le da el valor de la coordenada Y del centroide de la nueva región.
- Ancho: Exactamente igual que los anteriores, se le da el valor del ancho de la nueva región.
- Alto: De nuevo, igual que los anteriores.
- Velocidad X: Se pone a 0, ya que no se cuenta con ninguna información previa.
- Velocidad Y: También se pone a 0.
- Visibilidad: Se pone a 1 porque el objeto está visible.
- Edad: Se pone a 1 su valor.
- Contador de inactividad: Se mantiene a 0.

Una vez que se conoce cómo se clasifican las nuevas regiones y cómo se crean los tracks, el siguiente paso es ¿en qué lugar de la memoria colocar el nuevo track? Como ya se ha explicado, cuando se crea un track y se le asigna una posición, ese track ya no va a moverse de ese lugar hasta que sea eliminado por inactividad. Cuando un track se elimina, deja un hueco vacío disponible en la memoria que puede ser asignado a nuevos tracks. Por tanto, si existen huecos disponibles, los nuevos tracks van a ser colocados en dichos espacios. En el caso de que no existan huecos disponibles, se colocan al final de la cola de tracks.

4.2.2.4 Gestión de memoria

La gestión de memoria en este proyecto se basa en la asignación y liberación de huecos en la matriz de tracks, la cual tiene un número definido de secciones de memoria. Tal y como ha sido explicado en el subapartado anterior, a la hora de crear nuevos tracks es necesario asignarles un lugar en dicha matriz. Este lugar que se le asigna al nuevo track dependerá principalmente de si existen tracks que ya han sido eliminados y han creado un hueco, pudiendo así alojar a uno nuevo.

A continuación, en el algoritmo 4.7 se representa el seudocódigo correspondiente a la creación de nuevos tracks con gestión de memoria:

Algoritmo 4.7 Algoritmo para creación de nuevos tracks con gestión de memoria

```
0 ← int, 0 ← act, 0 ← act2
for s← 0 .. número de tracks, +1
  if track se elimina
      vector_tracks_eliminados[int]=s
      int + 1 ← int
  end if
  act + 1 ← act
  act2 + 1 ← act2
end for
número de tracks que se eliminan ← huecos
if huecos != 0
 if número de regiones nuevas > huecos
   for s← 0 .. huecos, +1
     Crear nuevo track en lugar de anterior track con identificador = vector_tracks_eliminados[s]
   end for
   for s← huecos .. número de regiones nuevas, +1
     Crear nuevo track en lugar de anterior track con identificador = act
     act + 1 ← act
   end for
  end if
  if número de regiones nuevas <= huecos
   for s← 0 .. número de regiones nuevas, +1
     Crear nuevo track en lugar de anterior track con identificador = vector_tracks_eliminados[s]
   end for
```

```
   end if
end if
if huecos == 0
  for s← 0 ..  número de regiones nuevas, +1
     Crear nuevo track en lugar de anterior track con identificador = act2
     act2 + 1 ← act2
  end for
end if
```

En el seudocódigo se tienen en cuenta todas las opciones posibles:

- Si hay huecos y el número de nuevas regiones es mayor que el número de huecos, se crearán nuevos tracks en los huecos hasta rellenarlos todos y los que falten por crear se crean detrás del último track de la lista.
- Si hay huecos y el número de nuevas regiones es igual o menor que el número de huecos, se crean los tracks directamente en los huecos, empezando por el primer hueco que se encuentre en la matriz.
- Si no hay huecos se crean los tracks detrás del último track de la lista.

4.3 Cross-tracking

En el apartado anterior se han visto varios de los casos de clasificación de un track. Se han estudiado los casos de tracks que se asocian, tracks que no se asocian y la creación de nuevos tracks. Todos estos casos son necesarios y muy importantes, pero a la hora de poner en práctica este proyecto con metraje real, aparecen nuevos casos que también hay que incluir a la hora de clasificar los tracks y actualizarlos de una manera determinada.

El caso explicado en esta subsección es denominado *"cross-tracking"* y ocurre cuando dos objetos a los cuales se les está realizando el seguimiento, pasan uno muy cerca del otro, provocando que al segmentarse las regiones de cada objeto no se detecten dos regiones sino una sola. Para evitar problemas a la hora de la clasificación, hay que añadir un nuevo caso en el algoritmo de clasificación de tracks.

Tal y como ha sido diseñada la matriz de asociación, cuando se da este caso de cross-tracking, de los dos tracks implicados, uno se clasifica como asociado con la región segmentada que contiene a los dos objetos mientras que el otro pasa a ser clasificado como track no asociado.

Existe otro caso distinto también parecido al cross-tracking y es cuando existe un error total de segmentación. Es decir, un track que ha estado asociándose con regiones en distintas imágenes pasa a dejar de asociarse de repente porque en la imagen actual no se ha segmentado ninguna región a la que asociarse. Este caso se explica de manera más detallada en la siguiente subsección del capítulo. Pero a la hora de clasificarlo, ha de clasificarse exactamente igual que el track perteneciente al cross-tracking que no se asocia.

En el algoritmo de clasificación, cuando se da el caso de cross-tracking, el track que se asocia la doble región permanece como asociado inicialmente y se cambiará su clasificación una vez se sale del algoritmo, en uno nuevo. Para el track que no se asocia, es importante distinguirlo de los tracks que

acaban por lo tanto, se va a establecer que cuando un track no se asocia y no se trata de un track que acaba, es decir que no está en los límites del vídeo, entonces se trata de un caso de "cross-tracking o no segmentación de la región". Para todos estos, al igual que en los casos ya explicados, se guardan sus índices en un vector.

Una vez que se han preclasificado los tracks, hay que clasificarlos para posteriormente pasar a la actualización. El algoritmo de clasificación consiste en recorrer todos los tracks que están en el vector de "cross-tracking o con región no segmentada" para cada track asociado y calcular la distancia entre la predicción del track asociado y la predicción del track clasificado como "cross-tracking o con región no segmentada". La distancia se calcula mediante la siguiente expresión:

$$d = \sqrt{\left(x\_p_{track\ asociado} - x\_p_{track\ desaparecido}\right)^2 + \left(y\_p_{track\ asociado} - y\_p_{track\ desaparecido}\right)^2} \quad (4.11)$$

Una vez calculada la distancia, el siguiente paso es saber si dicha distancia es menor a un cierto parámetro definido. Es decir, si los centroides de las predicciones de ambos tracks implicados en el cross-tracking están a una distancia menor a la indicada, entonces ambos tracks se clasifican como partes del cross-tracking. Y por tanto, se actualizan de la misma manera. En el momento en el que deje de haber cross-tracking, los tracks se asociarán con normalidad a sus regiones correspondientes.

A continuación se muestra en el algoritmo 4.8 el seudocódigo de la clasificación de cross-tracking.

Algoritmo 4.8 Algoritmo para clasificación de cross-tracking.

```
0 ← int, 0 ← int2,
for s← 0 .. número de tracks asociados, +1
   for i← 0 .. número de tracks en cross-tracking o con región no segmentada, +1
     Calcular distancia entre las predicciones de track asociado y el otro track
     if distancia calculada < parámetro
        vector_crosstrack[int]=vector_crosstrack_o_no_segmentado[i]
        vector_crosstrack2[int2]=matriz_asignación[s][0]
        int + 1 ← int, int2 + 1 ← int2
     end if
   end for
end for
```

En la fase de actualización de este caso, al no contar con las dos regiones separadas de cada uno de los tracks, se van a usar los datos predichos para ambos. Por lo que las coordenadas X e Y de los centroides del track van a ser las predichas, y las velocidades y los tamaños se quedan como la última imagen en la que el track tuvo una asociación normal. El resto de parámetros se actualizan con normalidad: se mantiene visible, la edad se incrementa en 1, la inactividad está a cero y la señal de eliminación también a cero.

Por último, se muestra un ejemplo de cross-tracking. En la imagen mostrada en la figura 4.9, pueden verse dos tracks (marcados en rojo) con sus dos predicciones (en verde). Cada uno con una región asociada.

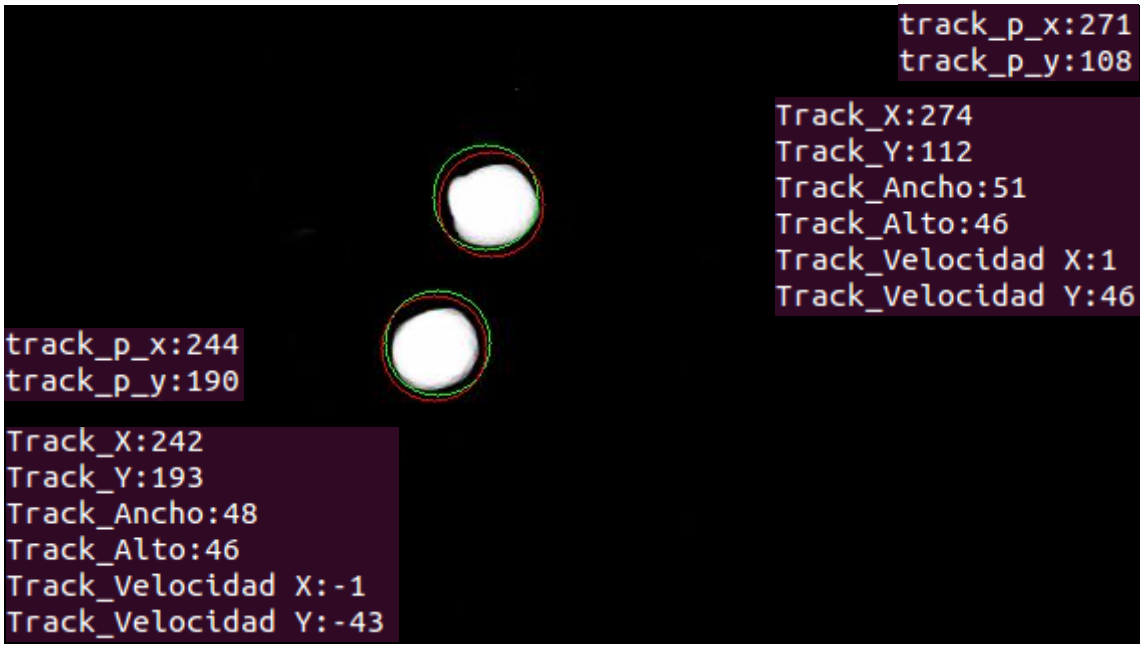

Figura 4-9. Imagen previa al cross-tracking, con dos tracks asociados.

En la imagen de la figura 4.10 se ve que ya se ha producido el cross-tracking. El track se actualiza usando las predicciones. Lo que se ve marcado en rojo es a la vez la posición del track y la posición predicha. El resto de parámetros no se actualiza, salvo la edad que se incrementa en 1.

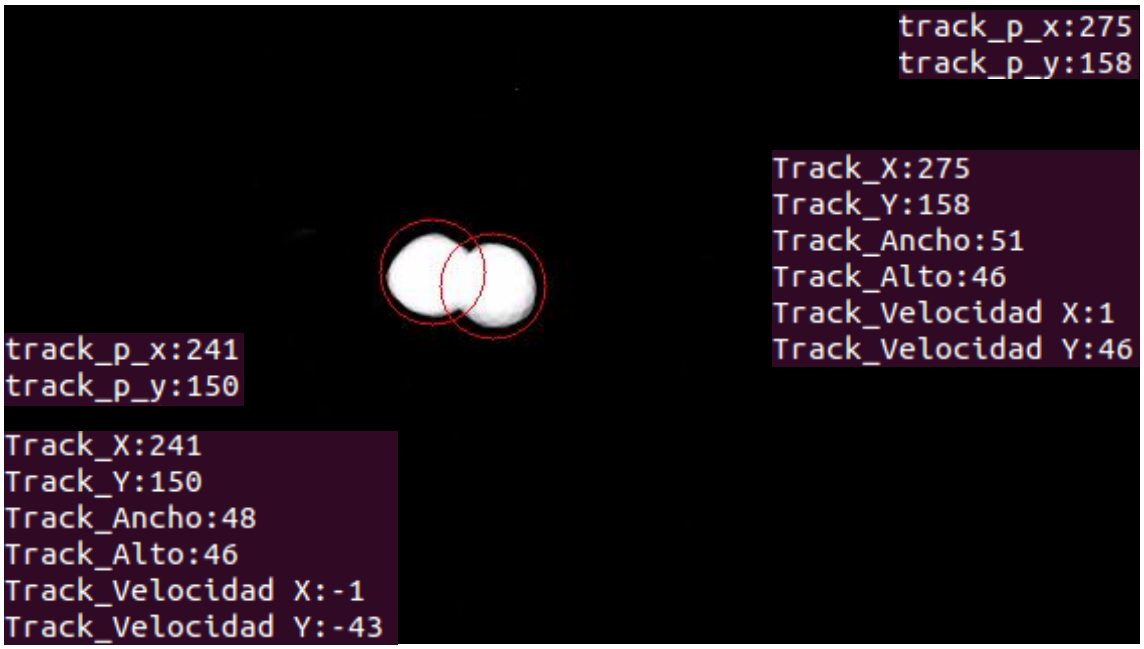

Figura 4-10. Imagen con caso de cross-tracking.

En la imagen de la figura 4.11, los tracks ya han salido del caso de cross-tracking por lo que se asocian a las regiones con normalidad.

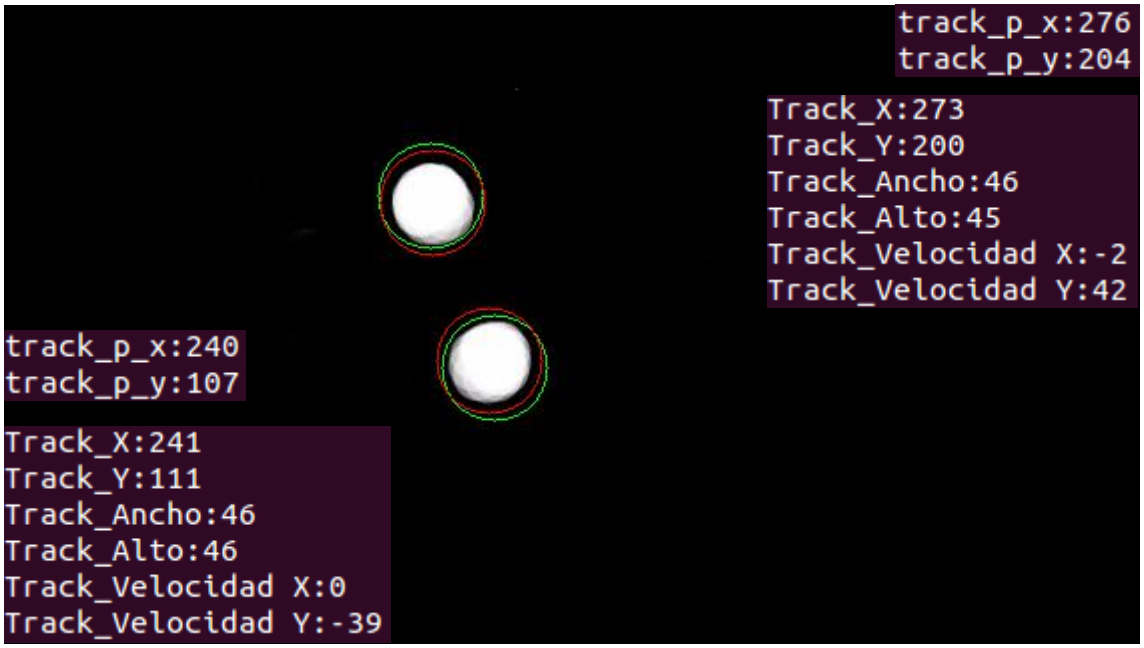

Figura 4-11. Imagen con dos tracks asociados justo después de cross-tracking.

4.4 Errores de segmentación

La segmentación de las imágenes no es perfecta por lo que hay que tener en cuenta algunos de los errores generados en esta. Existen distintos tipos de errores en la segmentación, uno de ellos es el ruido que aparece en cada imagen en lugares aleatorios. Otro tipo de error es cuando una región no se segmenta correctamente. Puede darse que la región no se segmenta al completo o también que se segmente pero en varias partes. A continuación se describen soluciones a estos errores.

Para evitar que el ruido cree falsas asociaciones, en la creación de tracks a partir de regiones detectadas, se limita que dichas regiones han de estar en las zonas de entrada de personas. Esto ayudará a que cualquier región ruidosa que no esté en los límites no se va a tener en cuenta a la hora crear tracks.

Los errores de segmentación de las regiones de interés son de dos tipos principalmente. Se explican cada una de ellas con su solución.

4.4.1 Error de segmentación total

Este tipo de error se da cuando se está realizando el seguimiento de un objeto y de manera no esperada, dicho objeto no se asocia con ninguna región debido a que esta no se ha segmentado en una imagen cuando se esperaba que si lo hiciera.

Como ya se mostró antes en la subsección 4.3 del capítulo, este caso está muy relacionado con el track del cross-tracking que no se asocia ya que se trata de un track que deja de asociarse cuando no debiera. Del vector ya explicado antes, donde se almacenan todos los tracks de "cross-tracking o con región no segmentada", todos los tracks pertenecientes a este vector y que no se clasifican como cross-tracking, se suponen tracks con error de segmentación total.

A la hora de actualizar, se hace igual que como en el cross-tracking, utilizando las posiciones predichas

como las posiciones de los tracks y manteniendo el resto de la información sin actualizar (salvo la edad, que se incrementa en 1).

A continuación se muestra un ejemplo de esto. En la imagen de la figura 4.12 se observa un track (marcado en rojo) con su región asociada y con su predicción (marcada en verde).

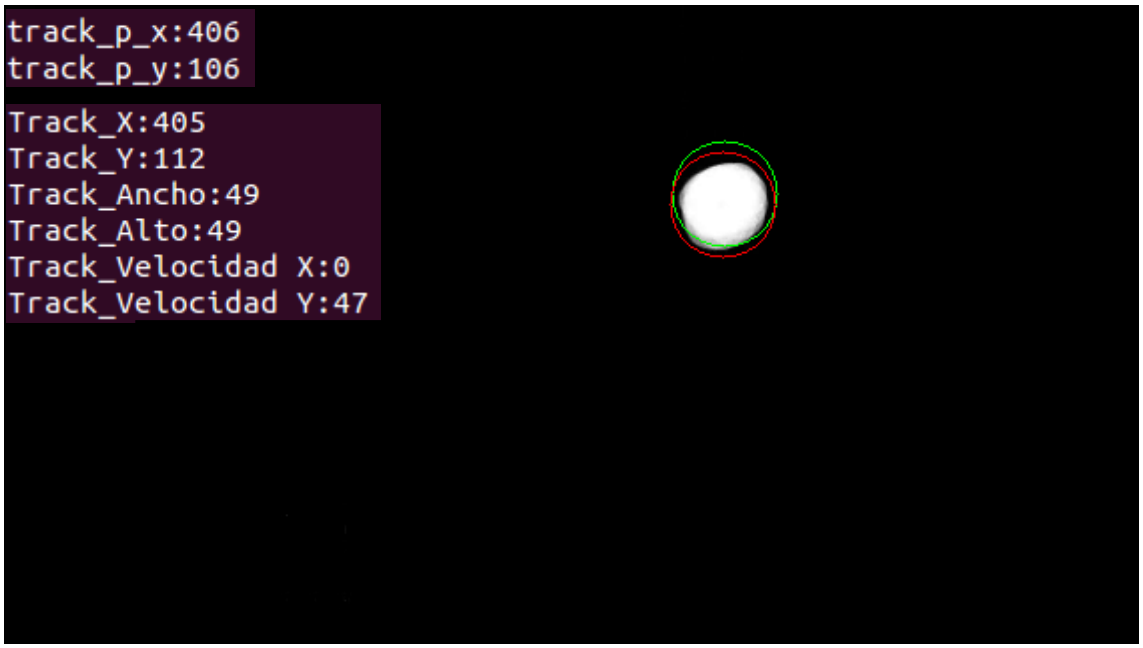

Figura 4-12. Imagen previa a un error de segmentación total.

En la siguiente imagen, en la figura 4.13, se ve que no se ha segmentado ninguna región por lo que el track se clasifica como explicado anteriormente y se actualiza mediante sus predicciones.

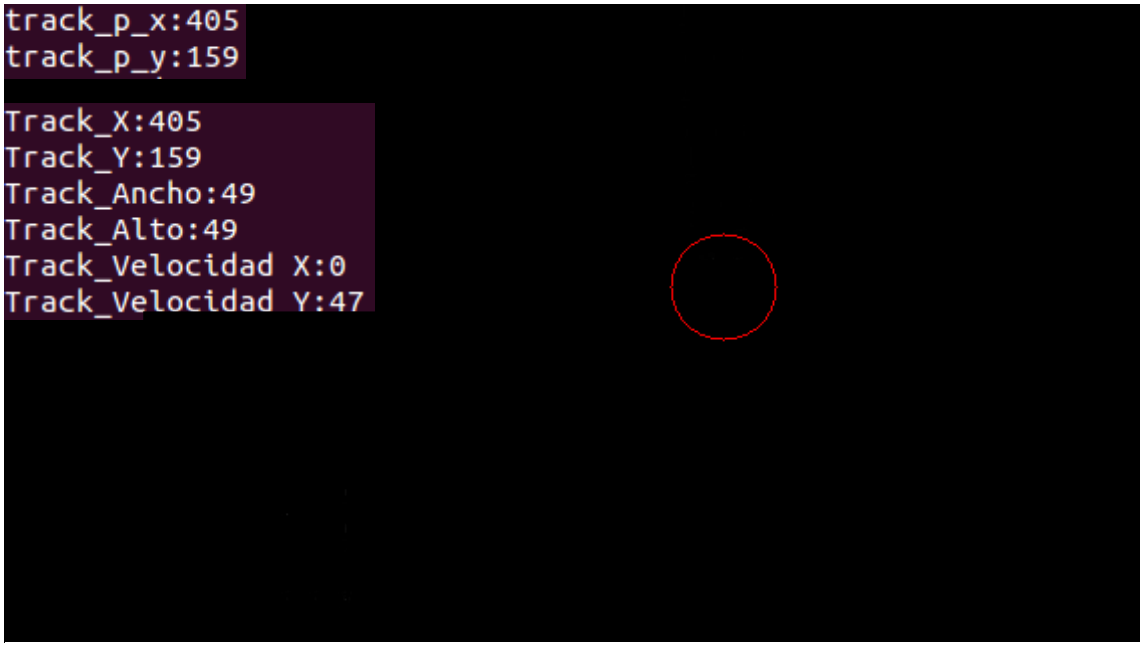

Figura 4-13. Imagen con caso de error total de segmentación.

Por último, en la figura 4.14, se muestra la siguiente imagen, en la que el track se asocia correctamente con la región que se detecta. En el caso de que continuara sin asociarse con ninguna región, se seguiría actualizando con las posiciones predichas.

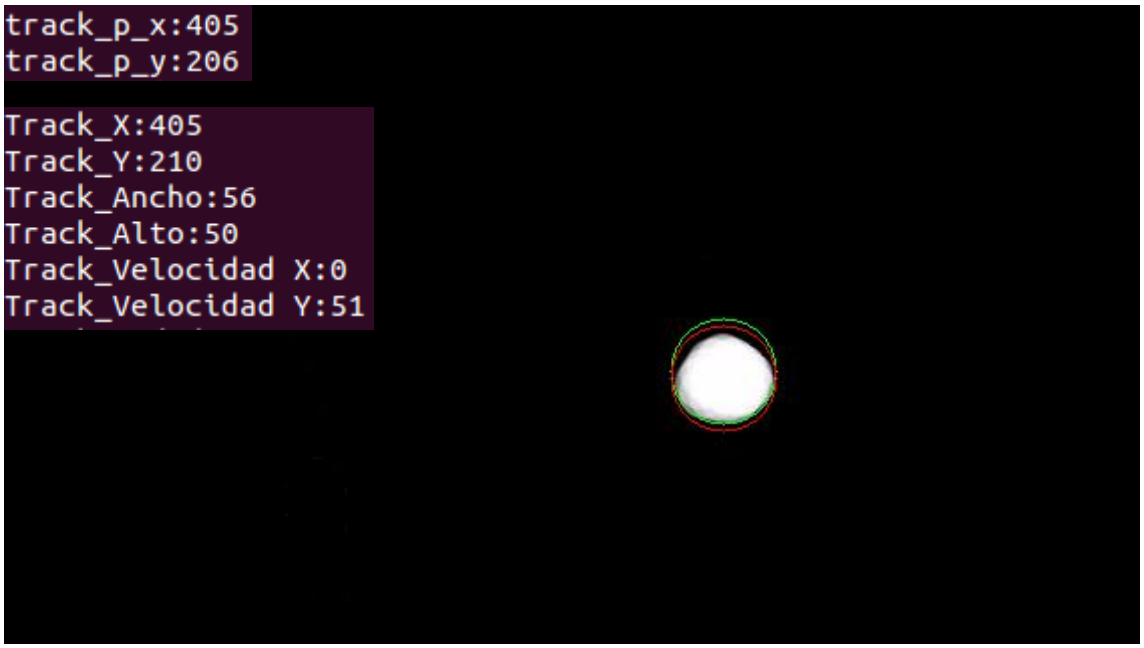

Figura 4-14. Imagen con un tracks asociado justo después de error de segmentación total.

4.4.2 Error de segmentación parcial

El otro caso de error en la segmentación es el más común de todos y se caracteriza porque lo que se supone que debe de ser un único objeto, se segmenta en varias regiones de menor tamaño comparado con el del objeto original. Suele ocasionarse en el momento de binarizar la imagen, por el umbral escogido o al erosionar. Mediante la dilatación se puede conseguir que este error no ocurra tanto, pero es difícil solucionarlo del todo.

Una solución parcial a este problema consiste en añadir los tamaños de las regiones en la expresión de la distancia en la fase de asociación. Al incluir los tamaños, no sólo se tendrá en cuenta la posición de las regiones a la hora de asociar, sino que se buscará la región con el tamaño más afín al tamaño del track en ese momento. Consiguiendo así que en el caso de que un objeto se segmente en varias regiones, se asocie al track la que esté a menor distancia pero que a su vez tenga un tamaño similar.

Para hacer esto, es necesario predecir los tamaños, aunque en vez de calcularlos mediante una suma, directamente se dice que los tamaños predichos son iguales que los del track en la imagen anterior. Reescribiendo la expresión, queda como se muestra a continuación:

$$d_1 = \sqrt{\left(x_{track\ predicho} - x_{región}\right)^2 + \left(y_{track\ predicho} - y_{región}\right)^2} \tag{4.12}$$

$$d_2 = \sqrt{\left(ancho_{track\ predicho} - ancho_{región}\right)^2 + \left(alto_{track\ predicho} - alto_{región}\right)^2} \tag{4.13}$$

$$d = \alpha \cdot d1 + \beta \cdot d2 \tag{4.14}$$

siendo α y β dos parámetros usados para darle mayor prioridad a distancia o a tamaño

respectivamente.

En la figura 4.15 se ve una imagen en la que hay un objeto segmentado en varias partes. A la hora de asociar, se ve que el track elige a la región de mayor tamaño. Esto es para $\alpha = 1$ y $\beta = 2$.

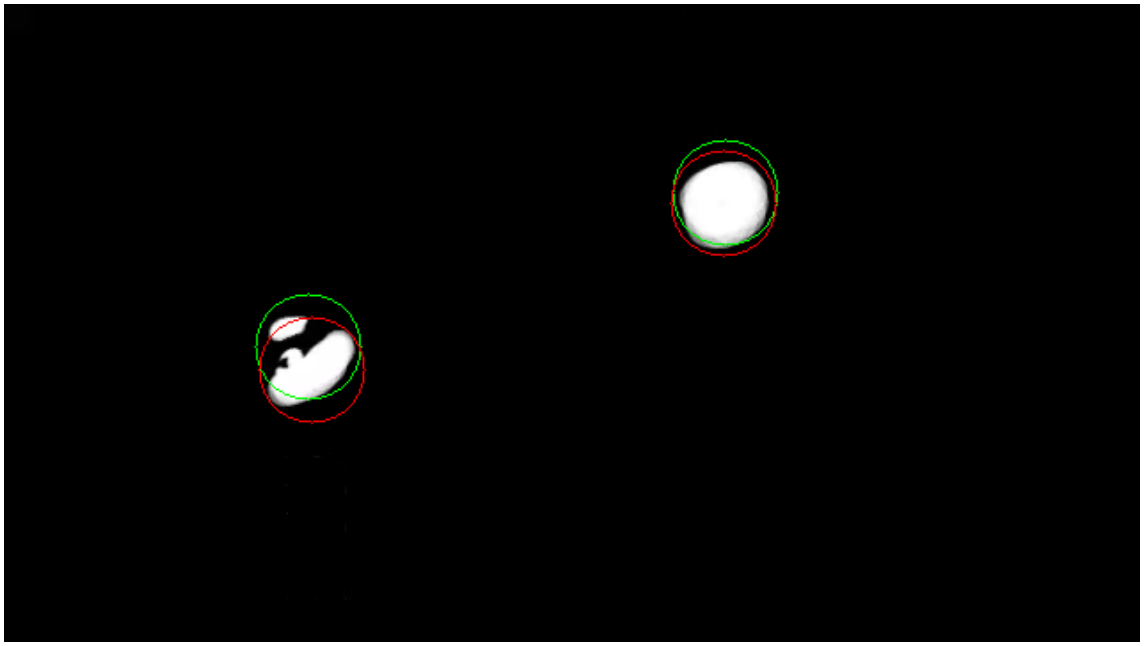

Figura 4-15. Imagen con un track con caso de asociación a una región con error de segmentación parcial.

Si ahora se prueba para una nueva imagen $\alpha = 2$ y $\beta = 1$, en la figura 4.16 se ve que el track de la izquierda no elige a la región de mayor tamaño, sino a la de menor distancia.

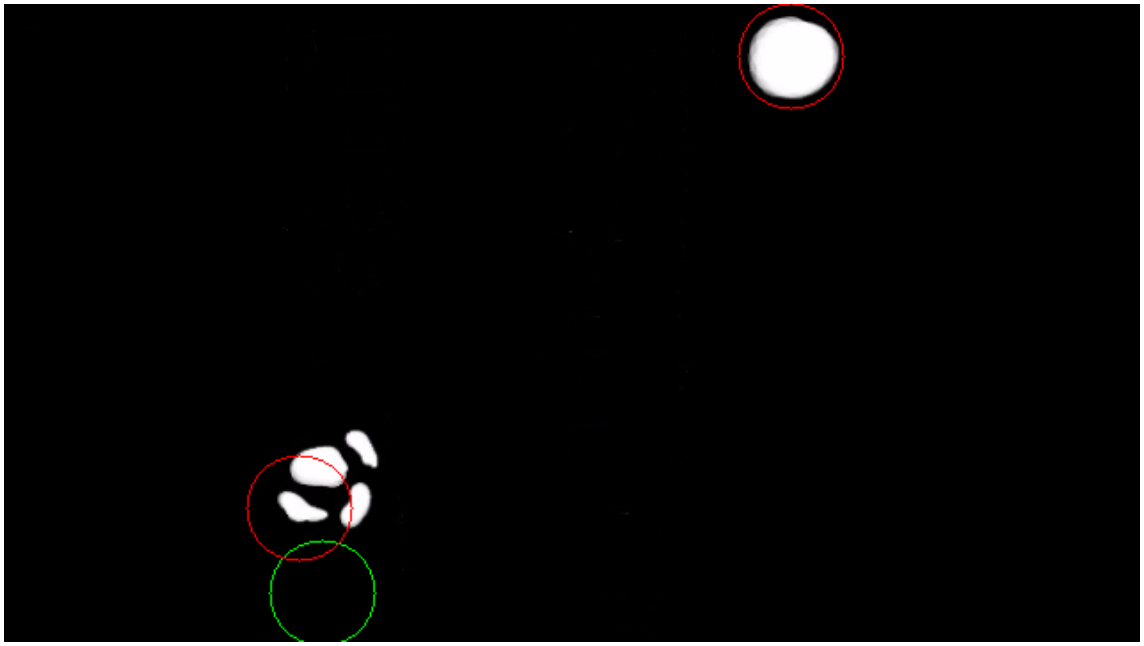

Figura 4-16. Imagen con un track con caso de mala asociación a una región con error de segmentación parcial.

Al probar con esta misma imagen, figura 4.17, para $\alpha = 1$ y $\beta = 2$, se observa que en este caso si elige a la de mayor tamaño.

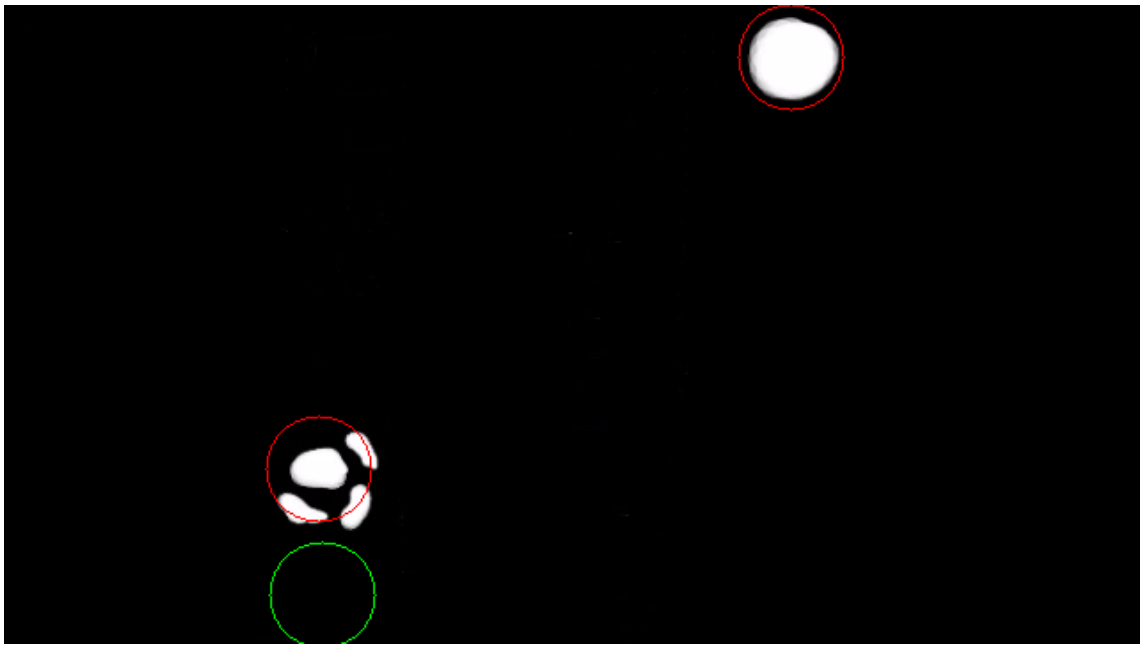

Figura 4-17. Imagen con un track con caso de una correcta asociación a una región con error de segmentación parcial.

4.5 Generación y lectura de fichero

Por último y de cara a desarrollos futuros, cada vez que un track termine y se elimine, se va a incluir su trayectoria completa en un archivo de texto que posteriormente es leído por un programa secundario para contar el número de personas que pasan por la zona de visión de la cámara.

4.5.1 Generación del fichero

Para el almacenamiento de las trayectorias se crean dos matrices en las que cada fila va a corresponder a un track y cada casilla de estas filas a un punto perteneciente a dicha trayectoria. Se crean dos matrices teniendo las coordenadas X en una y las coordenadas Y en la otra. Estas matrices se actualizan con cada nueva imagen añadiendo nuevos puntos hasta que la señal de eliminación se active. En el momento en el que se activa dicha señal, se escriben en un fichero de texto todos los puntos de las trayectorias de los tracks acabados. Cada vez que acaba un track, su trayectoria se añade al fichero, pero sin eliminar las trayectorias de los tracks previamente acabados. Con esto, se consigue un fichero en el que se tiene un historial de la trayectoria que toma cada persona que pasa por la zona donde se encuentra grabando la cámara.

4.5.2 Lectura del fichero

Para probar el correcto funcionamiento de la generación de este fichero, se crea un programa secundario que se ejecuta desde el mismo programa principal. Este tiene la función de leer en cada iteración del bucle general del programa (cada vez que se lee una nueva imagen) el fichero con las

trayectorias de los tracks que acaban y realizar una cuenta de las personas que pasan en un sentido y en otro.

Para la cuenta de personas en un sentido y en otro según las trayectorias del fichero, lo primero que se hace es leer dicho fichero y almacenar en un vector las coordenadas Y de los puntos de las trayectorias. Para que posteriormente se aplique el seudocódigo visto en el algoritmo 4.9.

Algoritmo 4.9 Algoritmo para contar personas según el fichero .txt generado

```
0 ← actual, 0 ← anterior, 0 ← cuenta_arriba, 0 ← cuenta_abajo
for i← 0 .. número de tracks en el fichero, +1
   for j← 0 .. número de elementos Y en la trayectoria, +1
      if vector_trayectorias[i][j] != 0
        vector_trayectorias[i][j] ← actual
        vector_trayectorias[i][j-1] ← anterior
        if ((actual<barrera && anterior>barrera) || (actual>barrera && anterior<barrera))
          if (actual-anterior) > 0
              cuenta_abajo + 1 ← cuenta_abajo
          end if
          if (actual-anterior) < 0
              cuenta_arriba + 1 ← cuenta_arriba
          end if
        end if
      end if
   end for
end for
```

En este algoritmo se recorren todos los puntos de las trayectorias del fichero y en el caso de que haya un momento en el que la trayectoria pase por una línea horizontal situada en la mitad de la zona de paso y definida como una barrera, se contará que una persona ha pasado. Para saber en qué sentido pasa, se realiza la resta entre el primer punto después de pasar la barrera y el punto justamente antes de pasarla. Si esta resta fuera negativa se cuenta que esta persona va hacia arriba en el sentido del video y si la resta fuera positiva, va hacia abajo.

4.6 Conclusiones

Una vez terminado el capítulo del seguimiento se concluye que los resultados pueden ser buenos siempre y cuando la imagen usada para detectar las regiones haya sido segmentada correctamente. En general, esta se segmenta bien pero es imposible eliminar todos los errores que aparecen y por tanto hay que tenerlos en cuenta a la hora de la asociación e intentar corregirlos.

Este proyecto cubre algunos de los casos que pueden darse, como los errores de segmentación ya mencionados o el cross-tracking, pero existen más casos que pueden darse y que por tanto habría que clasificar también. Por ejemplo, que una persona se pare en la zona de paso y vuelva por donde había venido o que dos personas entren en dicha zona juntas provocando así cross-tracking desde la creación de los tracks.

Por tanto, se concluye que los resultados del seguimiento van en función de los resultados obtenidos en la segmentación de imágenes. Y ya que esta no va a solucionar todos los problemas, hay que tenerlos en cuenta a la hora de realizar el seguimiento, más concretamente en la fase de asociación.

5 EXPERIMENTOS

El científico no tiene por objeto un resultado inmediato. Su deber es sentar las bases para aquellos que están por venir, y señalar el camino.

NIKOLA TESLA

Una vez explicados y desarrollados el procesamiento de imágenes y el seguimiento, se realizan los experimentos para comprobar el funcionamiento del proyecto. Se analizan los resultados obtenidos, tanto los positivos como los negativos, para así saber por dónde continuar trabajando en dicho estudio mediante desarrollos futuros.

Este capítulo se divide en varias secciones, en la primera de ellas van a realizarse los experimentos de segmentación usando vídeos reales y viendo qué tipo de resultados se obtienen. En la siguiente sección se van a realizar distintas pruebas con vídeos virtuales como los del capítulo anterior para probar el funcionamiento del bloque de seguimiento para los casos de tracks estudiados. Y por último, en la tercera sección se va a tratar el problema de seguimiento en vídeos reales, haciendo funcionar conjuntamente tanto el procesamiento de imágenes como el seguimiento.

5.1 Experimentos de segmentación

En esta sección se muestran los resultados del funcionamiento del procesamiento de imágenes del proyecto. El objetivo de este bloque consiste en sustraer de la imagen los objetos en movimiento y segmentarlos intentando conseguir que cada objeto sea representado por una única región y no por varios fragmentos separados. Para cada imagen del experimento, se muestra la imagen origen del vídeo y la imagen ya procesada así como la descripción de los resultados. Ahora se muestra el primer ejemplo, cuyas imágenes pertenecen a un mismo vídeo.

En las figuras 5.1 y 5.2, se observa una imagen del vídeo y el resultado de toda la fase de procesamiento.

Figura 5-1. Imagen de vídeo real.

Figura 5-2. Segmentación de la imagen.

A continuación, se enseñan más imágenes pertenecientes a este vídeo. Las imágenes de las figuras 5.3 y 5.4 siguen representando a la misma persona pasando. La segmentación es buena debido a que dicha persona está representada como una única región. Lo mismo pasa en la imagen de las figuras 5.5 y 5.6.

Figura 5-3. Imagen de vídeo real.

Figura 5-4. Segmentación de la imagen.

Figura 5-5. Imagen de vídeo real.

Figura 5-6. Segmentación de la imagen.

En las figuras 5.7 y 5.8 se puede ver que en este caso la pierna de esta persona se ha segmentado separada del cuerpo, ocasionando así una segmentación fragmentada.

Figura 5-7. Imagen de vídeo real.

Figura 5-8. Segmentación de la imagen

Al seguir experimentando con el mismo vídeo y con distintos casos de personas se obtienen resultados distintos, como los mostrados en las figuras 5.9, 5.10 y 5.11, 5.12. En este caso se ve que la segmentación no es tan precisa como antes.

Figura 5-9. Imagen de vídeo real.

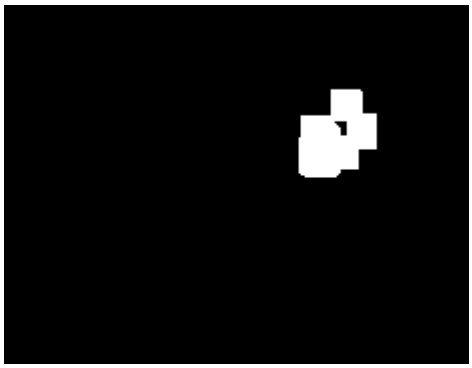

Figura 5-10. Segmentación de la imagen.

Figura 5-11. Imagen de vídeo real.

Figura 5-12. Segmentación de la imagen.

Esto mostrado arriba indica que se corren más riesgos de que la región se fragmente tal y como se muestra en las figuras 5.13 y 5.14 o casi se fragmenta como en las imágenes de 5.15 y 5.16.

Figura 5-13. Imagen de vídeo real.

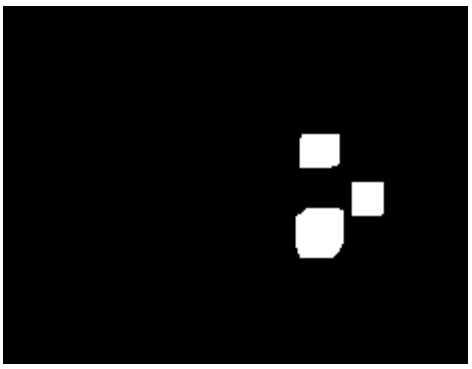

Figura 5-14. Segmentación de la imagen.

Figura 5-15. Imagen de vídeo real.

Figura 5-16. Segmentación de la imagen.

También existe la posibilidad de que las personas vengan fragmentadas desde el inicio del vídeo, como en las figuras 5.17, 5.18 y 5.19 y 5.20.

Figura 5-17. Imagen de vídeo real.

Figura 5-18. Segmentación de la imagen.

Figura 5-19. Imagen de vídeo real.

Figura 5-20. Segmentación de la imagen.

Por último, también se da el caso de que dos personas que pasan una muy cerca de la otra, al segmentarse, se vean como una única región. Se ve en las figuras 5.21, 5.22 y 5.23, 5.24.

Figura 5-21. Imagen de vídeo real.

Figura 5-22. Segmentación de la imagen.

Figura 5-23. Imagen de vídeo real.

Figura 5-24. Segmentación de la imagen.

A continuación se muestran los resultados del procesamiento de vídeos para otro vídeo diferente y real. La resolución de este vídeo es mayor y por tanto las imágenes representadas van a ser de mayor tamaño que las del vídeo anterior.

En las figuras 5.25 y 5.26 se ve la nueva zona de paso examinada y el resultado de la segmentación completa, que en este caso resulta buena.

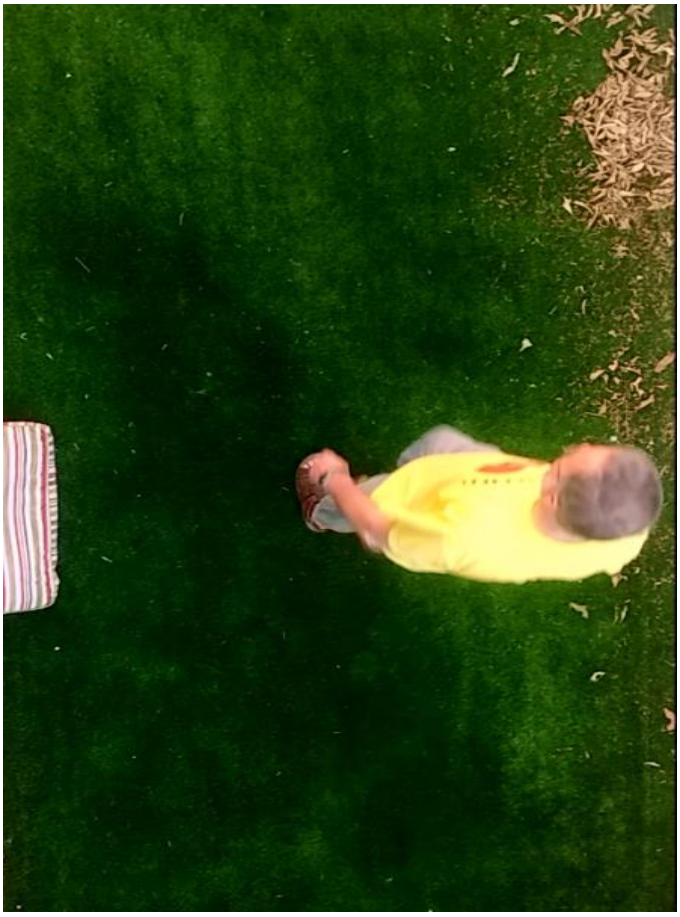

Figura 5-25. Imagen de otro vídeo real.

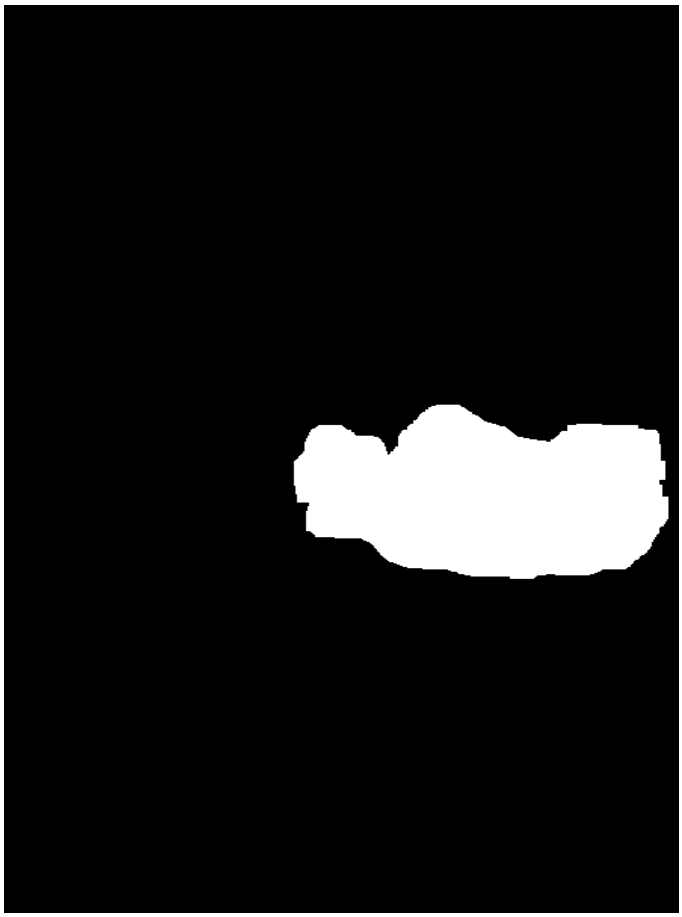

Figura 5-26. Segmentación de la imagen.

Si se observan ahora las figuras 5.27 y 5.28, se ve que la persona está segmentada en varios fragmentos, por tanto la segmentación puede mejorarse para evitar esto.

También puede verse en las figuras 5.29 y 5.30 un caso de error de segmentación parcial, en este caso en la salida de una persona por uno de los límites del vídeo.

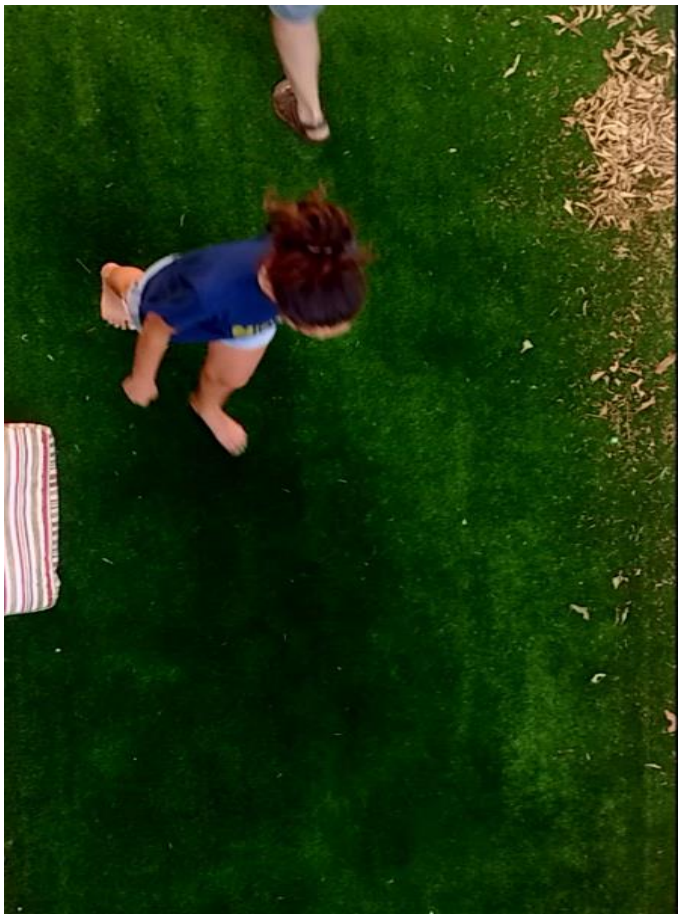

Figura 5-27. Imagen de vídeo real.

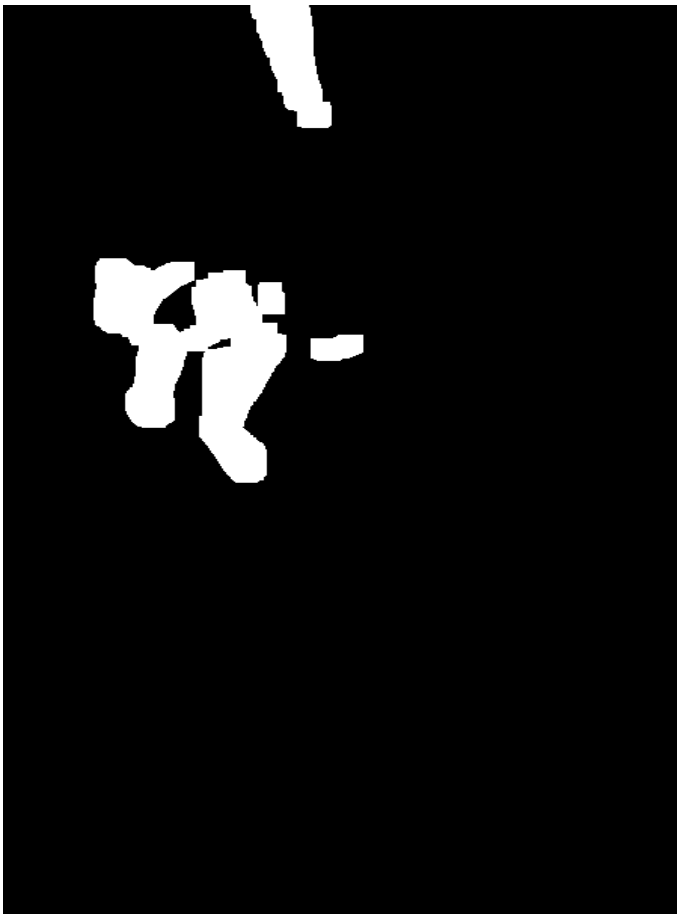

Figura 5-28. Segmentación de la imagen.

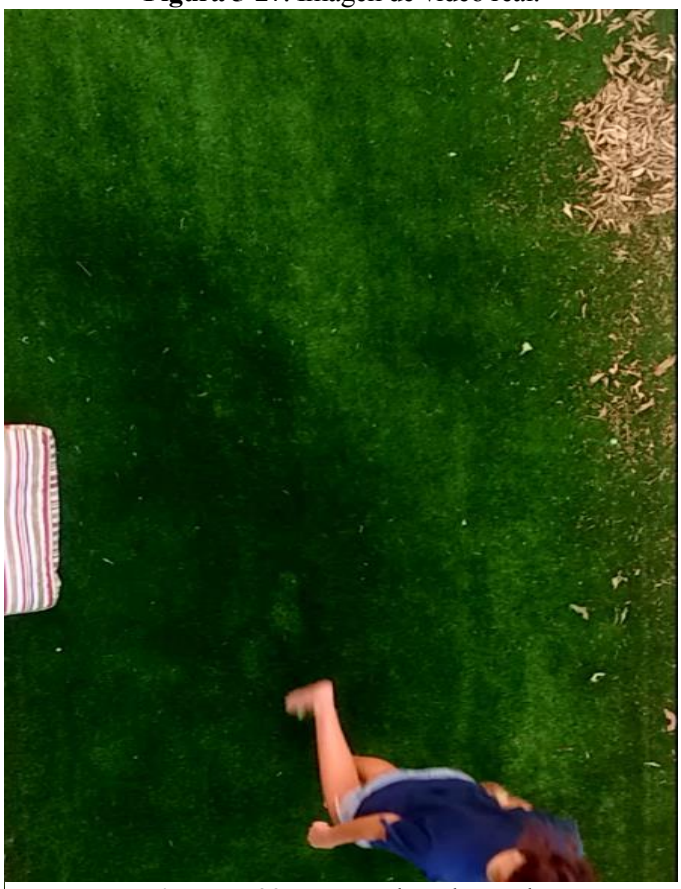

Figura 5-29. Imagen de vídeo real.

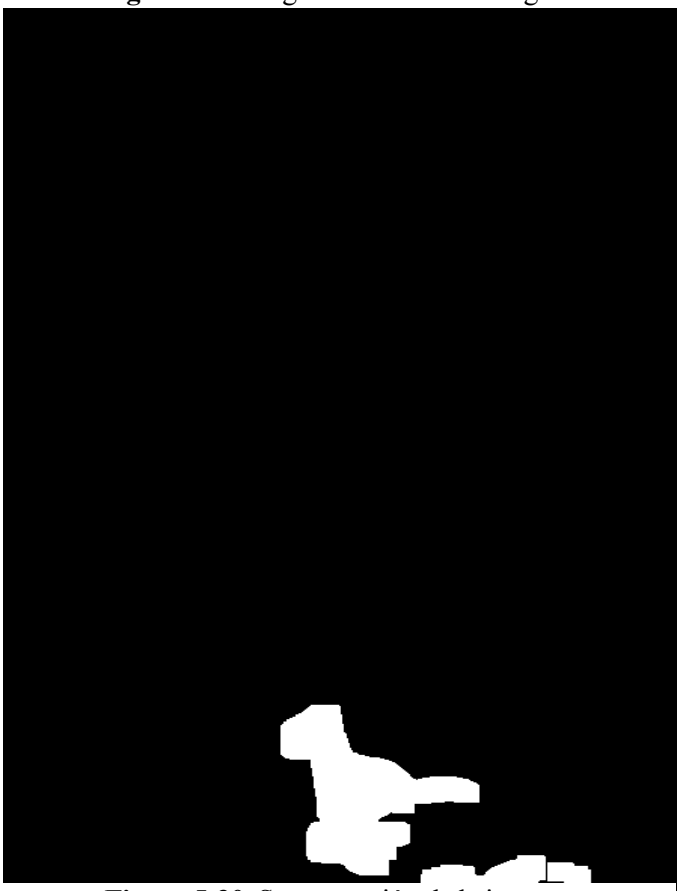

Figura 5-30. Segmentación de la imagen.

En las figuras 5.31 y 5.32 se muestra el caso de dos personas teniendo una misma región segmentada.

Figura 5-31. Imagen de vídeo real.

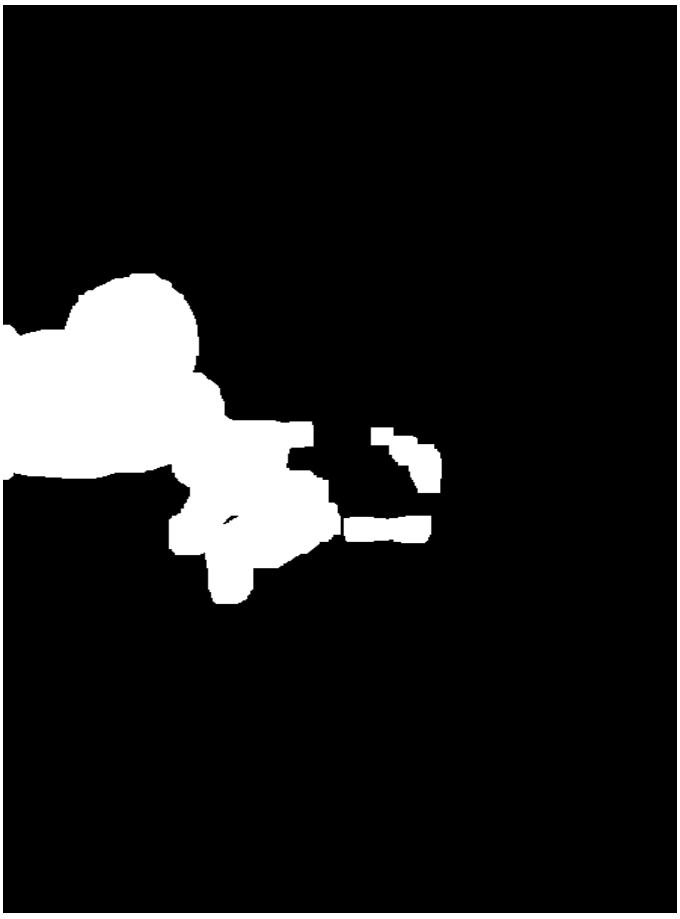

Figura 5-32. Segmentación de la imagen.

5.2 Experimentos de seguimiento

Para los experimentos de seguimiento, se va a usar el mismo tipo de vídeo que se ha usado previamente en el capítulo de seguimiento, es decir, videos virtuales en los que se simulan personas en movimiento ya segmentadas. Con estos vídeos se puede poner a prueba el funcionamiento del algoritmo de seguimiento sin contar con muchos de los fallos que existen en las imágenes provenientes del bloque de procesamiento.

A continuación se verán varias partes de estos vídeos para confirmar que el seguimiento de los objetos se realiza de manera correcta en los distintos casos estudiados. Debido al tamaño de las imágenes, no se representarán todas, sino las más significativas a la hora de verificar el correcto funcionamiento del seguimiento.

Primero, se va a probar un vídeo en el que aparecen los casos de creación, asociación y no asociación de tracks. En la figura 5.33 se ve una imagen correspondiente a dicho vídeo en la que hay una región y para ella se crea un nuevo track (marcado en rojo). Los datos del track se muestran en la figura 5.34.

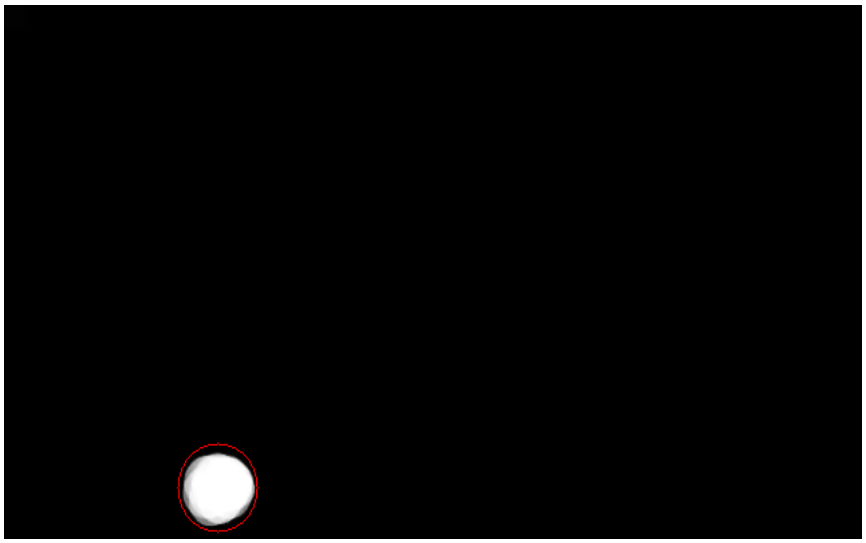

Figura 5-33. Imagen de vídeo virtual con seguimiento.

```
X:159
Y:325
Ancho:53
Alto:49
Velocidad X:0
Velocidad Y:0
```

Figura 5-34. Datos de la creación del track.

En 5.35 se ve la siguiente imagen del vídeo con la actualización del track y con su predicción (marcada en verde). En 5.36 se muestran los nuevos datos del track.

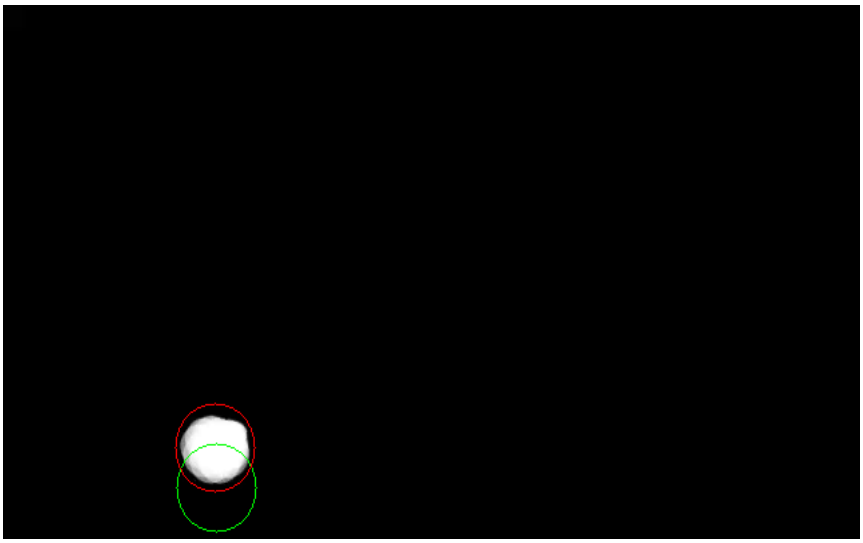

Figura 5-35. Imagen de vídeo virtual con seguimiento.

```
Track nº 0
Track_X:158
Track_Y:298
Track_Ancho:51
Track_Alto:45
Track_Velocidad X:-1
Track_Velocidad Y:-27
Visibilidad:1
Contador Inactividad:0
Edad:2
Track se elimina:0
```

Figura 5-36. Datos del track.

En la imagen de 5.37 se ve que el track 0 sigue asociándose sin problema y a la vez se crea un nuevo track. En 5.38 se representan los datos de ambos tracks.

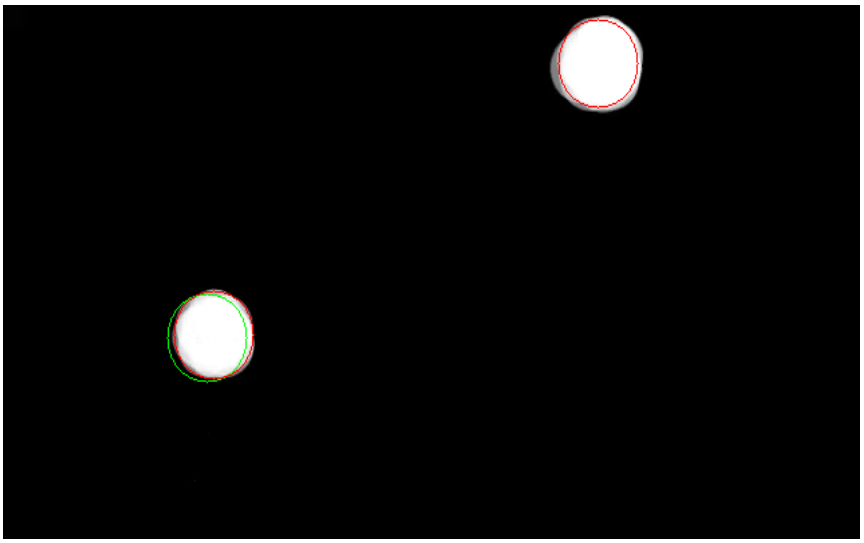

Figura 5-37. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1
Track_X:157	Track_X:444
Track_Y:222	Track_Y:39
Track_Ancho:61	Track_Ancho:69
Track_Alto:60	Track_Alto:65
Track_Velocidad X:2	Track_Velocidad X:0
Track_Velocidad Y:-39	Track_Velocidad Y:0
Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0
Edad:4	Edad:1
Track se elimina:0	Track se elimina:0

Figura 5-38. Datos de los tracks.

Los tracks 0 y 1 siguen asociándose con normalidad y se crea el track 2 (Figura 5.39). Con la información de todos los tracks en la figura 5.40.

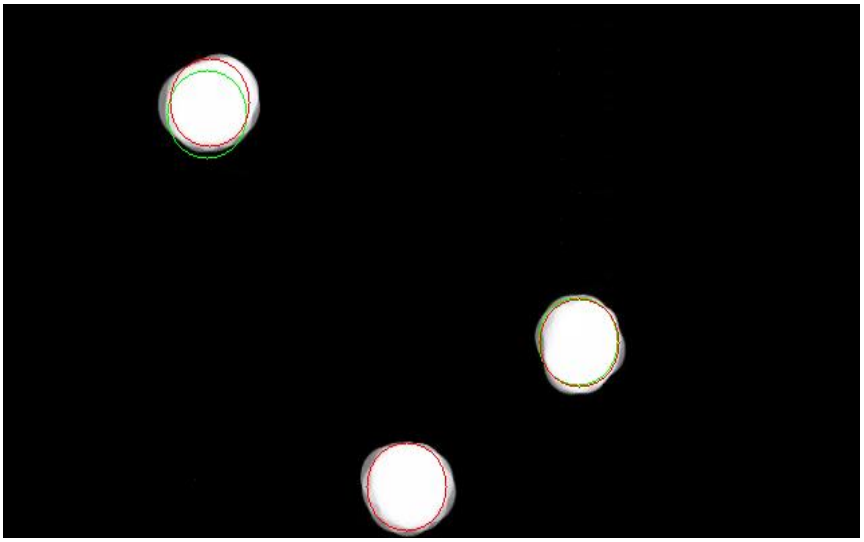

Figura 5-39. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1	Track nº 2
Track_X:154	Track_X:430	Track_X:301
Track_Y:66	Track_Y:228	Track_Y:325
Track_Ancho:76	Track_Ancho:66	Track_Ancho:71
Track_Alto:65	Track_Alto:67	Track_Alto:63
Track_Velocidad X:0	Track_Velocidad X:-5	Track_Velocidad X:0
Track_Velocidad Y:-57	Track_Velocidad Y:66	Track_Velocidad Y:0
Visibilidad:1	Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0	Contador Inactividad:0
Edad:7	Edad:4	Edad:1
Track se elimina:0	Track se elimina:0	Track se elimina:0

Figura 5-40. Datos de los tracks.

En esta imagen (figura 5.41), mientras que los tracks 1 y 2 continúan asociándose, el track 0 acaba y su información se actualiza correctamente. En 5.42 se dan sus datos.

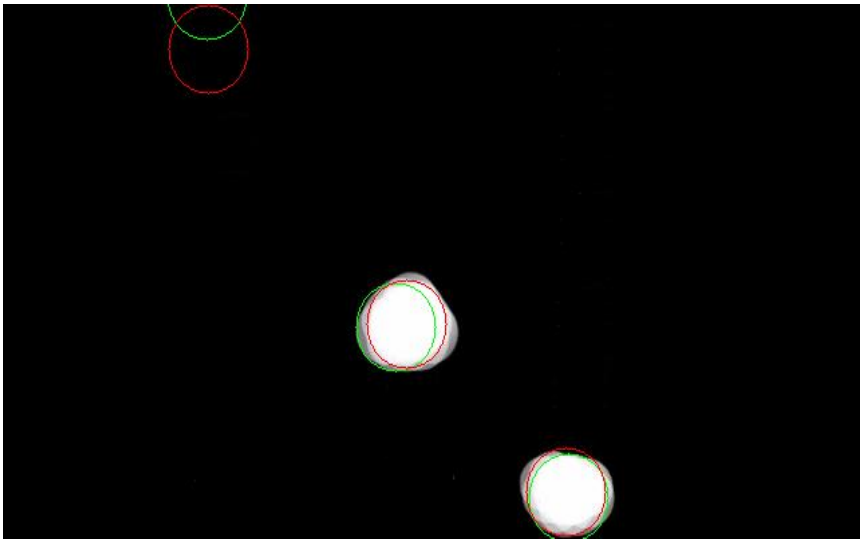

Figura 5-41. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1	Track nº 2
Track_X:153	Track_X:420	Track_X:301
Track_Y:30	Track_Y:328	Track_Y:215
Track_Ancho:69	Track_Ancho:69	Track_Ancho:76
Track_Alto:60	Track_Alto:57	Track_Alto:66
Track_Velocidad X:-1	Track_Velocidad X:-6	Track_Velocidad X:4
Track_Velocidad Y:-36	Track_Velocidad Y:48	Track_Velocidad Y:-56
Visibilidad:0	Visibilidad:1	Visibilidad:1
Contador Inactividad:1	Contador Inactividad:0	Contador Inactividad:0
Edad:9	Edad:6	Edad:3
Track se elimina:0	Track se elimina:0	Track se elimina:0

Figura 5-42. Datos de los tracks.

Por último respecto a este caso, en la figura 5.43, se ve cómo se elimina correctamente el track 0 una vez pasadas tres imágenes. También termina el track 1 y se asocian el track 2 y el 3 (este último se crea en la imagen anterior correspondiente a la mostrada en 5.43). En 5.44 se observa la información de estos tracks.

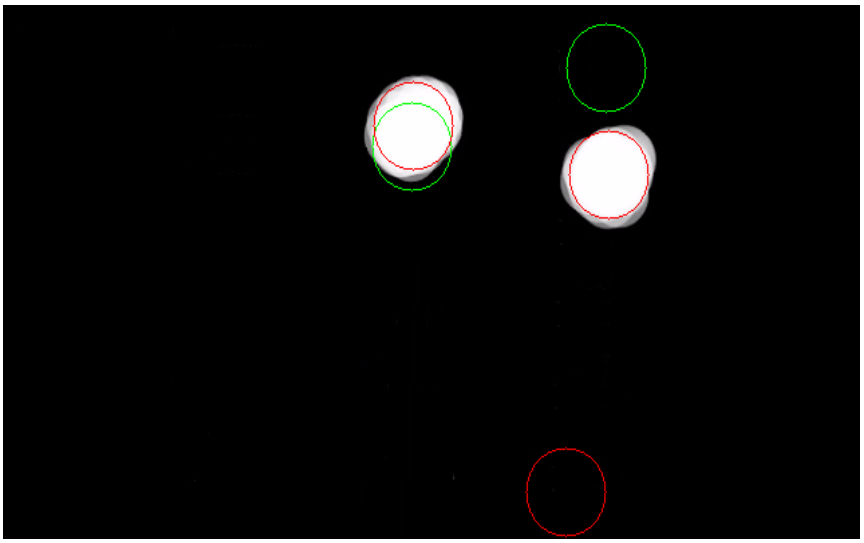

Figura 5-43. Imagen de vídeo virtual con seguimiento.

```
Track nº 0
Track_X:153
Track_Y:30
Track_Ancho:69
Track_Alto:60
Track_Velocidad X:-1
Track_Velocidad Y:-36
Visibilidad:0
Contador Inactividad:3
Edad:11
Track se elimina:1
```

```
Track nº 1
Track_X:420
Track_Y:328
Track_Ancho:69
Track_Alto:57
Track_Velocidad X:-6
Track_Velocidad Y:48
Visibilidad:0
Contador Inactividad:2
Edad:8
Track se elimina:0
```

```
Track nº 2
Track_X:306
Track_Y:81
Track_Ancho:73
Track_Alto:71
Track_Velocidad X:3
Track_Velocidad Y:-74
Visibilidad:1
Contador Inactividad:0
Edad:5
Track se elimina:0
```

```
Track nº 3
Track_X:452
Track_Y:114
Track_Ancho:70
Track_Alto:69
Track_Velocidad X:2
Track_Velocidad Y:72
Visibilidad:1
Contador Inactividad:0
Edad:2
Track se elimina:0
```

Figura 5-44. Datos de los tracks.

Se prueba ahora el algoritmo para un nuevo vídeo en el que se observa el caso de cross-tracking. En la imagen de la figura 5.45 se crean cuatro nuevos tracks. En 5.46 se ven los datos de estos tracks. El track 0 se corresponde con el track de abajo a la izquierda, el track 1 con el de abajo a la izquierda, el 2 con el de arriba a la izquierda y el 3 con el de arriba a la derecha.

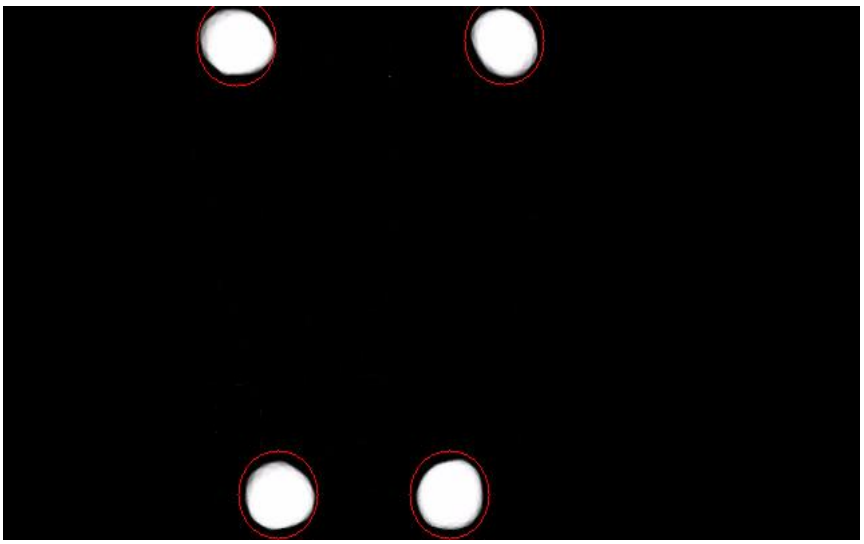

Figura 5-45. Imagen de vídeo virtual con seguimiento.

Track nº 0 Track_X:205 Track_Y:329 Track_Ancho:51 Track_Alto:46 Track_Velocidad X:0 Track_Velocidad Y:0 Visibilidad:1 Contador Inactividad:0 Edad:1 Track se elimina:0	Track nº 1 Track_X:333 Track_Y:329 Track_Ancho:49 Track_Alto:47 Track_Velocidad X:0 Track_Velocidad Y:0 Visibilidad:1 Contador Inactividad:0 Edad:1 Track se elimina:0
Track nº 2 Track_X:174 Track_Y:24 Track_Ancho:55 Track_Alto:43 Track_Velocidad X:0 Track_Velocidad Y:0 Visibilidad:1 Contador Inactividad:0 Edad:1 Track se elimina:0	Track nº 3 Track_X:374 Track_Y:23 Track_Ancho:49 Track_Alto:46 Track_Velocidad X:0 Track_Velocidad Y:0 Visibilidad:1 Contador Inactividad:0 Edad:1 Track se elimina:0

Figura 5-46. Datos de los tracks.

En la imagen de la figura 5.47 se ven a estos cuatro tracks asociados con normalidad, cada uno con una región. Esta imagen se corresponde con el instante previo al cross-tracking. En 5.48 se ve la información de cada track.

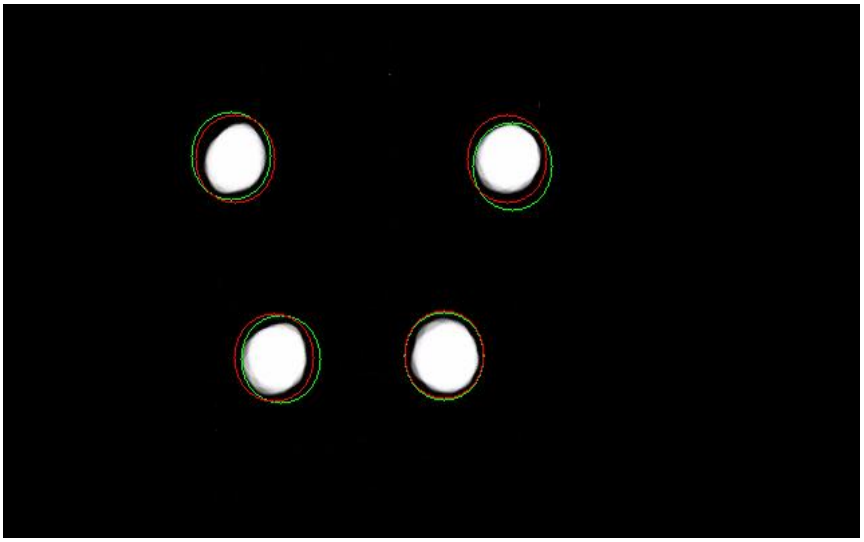

Figura 5-47. Imagen de vídeo virtual con seguimiento.

Track nº 0 Track_X:202 Track_Y:238 Track_Ancho:47 Track_Alto:48 Track_Velocidad X:-4 Track_Velocidad Y:-46 Visibilidad:1 Contador Inactividad:0 Edad:3 Track se elimina:0	Track nº 1 Track_X:329 Track_Y:236 Track_Ancho:50 Track_Alto:50 Track_Velocidad X:-2 Track_Velocidad Y:-47 Visibilidad:1 Contador Inactividad:0 Edad:3 Track se elimina:0
Track nº 2 Track_X:173 Track_Y:104 Track_Ancho:45 Track_Alto:46 Track_Velocidad X:1 Track_Velocidad Y:41 Visibilidad:1 Contador Inactividad:0 Edad:3 Track se elimina:0	Track nº 3 Track_X:376 Track_Y:104 Track_Ancho:48 Track_Alto:46 Track_Velocidad X:-1 Track_Velocidad Y:38 Visibilidad:1 Contador Inactividad:0 Edad:3 Track se elimina:0

Figura 5-48. Datos de los tracks.

En 5.49, los tracks 0 y 2 entran en contacto y se produce cross-tracking, haciendo así que se actualicen las posiciones de estos con las posiciones predichas. Los tracks 1 y 3 se asocian con normalidad. La información de todos estos tracks se ve en 5.50.

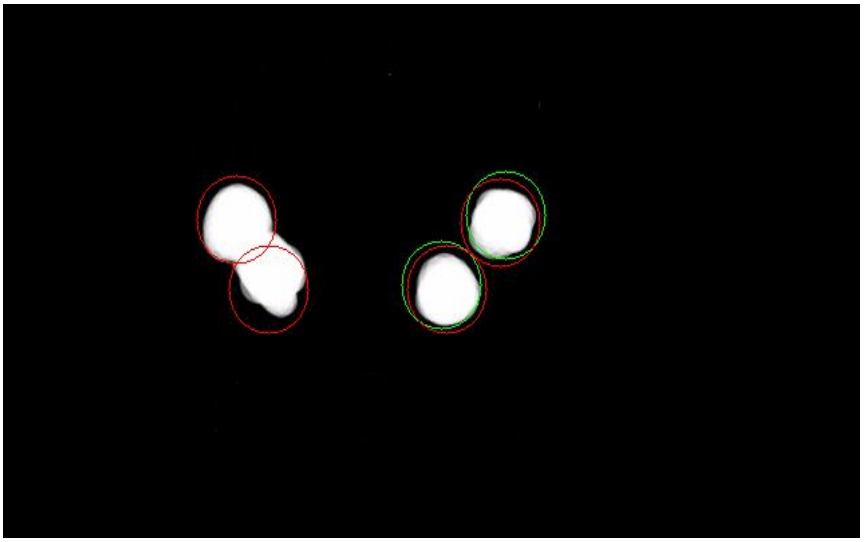

Figura 5-49. Imagen de vídeo virtual con seguimiento.

Track nº 0 Track_X:198 Track_Y:192 Track_Ancho:47 Track_Alto:48 Track_Velocidad X:-4 Track_Velocidad Y:-46 Visibilidad:1 Contador Inactividad:0 Edad:4 Track se elimina:0	Track nº 1 Track_X:331 Track_Y:192 Track_Ancho:47 Track_Alto:49 Track_Velocidad X:2 Track_Velocidad Y:-44 Visibilidad:1 Contador Inactividad:0 Edad:4 Track se elimina:0
Track nº 2 Track_X:174 Track_Y:145 Track_Ancho:45 Track_Alto:46 Track_Velocidad X:1 Track_Velocidad Y:41 Visibilidad:1 Contador Inactividad:0 Edad:4 Track se elimina:0	Track nº 3 Track_X:371 Track_Y:147 Track_Ancho:50 Track_Alto:46 Track_Velocidad X:-5 Track_Velocidad Y:43 Visibilidad:1 Contador Inactividad:0 Edad:4 Track se elimina:0

Figura 5-50. Datos de los tracks.

En 5.51, los tracks 0 y 2 continúan en estado de cross-tracking y ahora los tracks 1 y 3 también entran en este estado. Por tanto, todos los tracks en esta imagen actualizan con las posiciones predichas. Los datos de cada uno se ven en 5.52. En el caso de que se siguiera produciendo cross-tracking, se seguirían utilizando las posiciones predichas para actualizar.

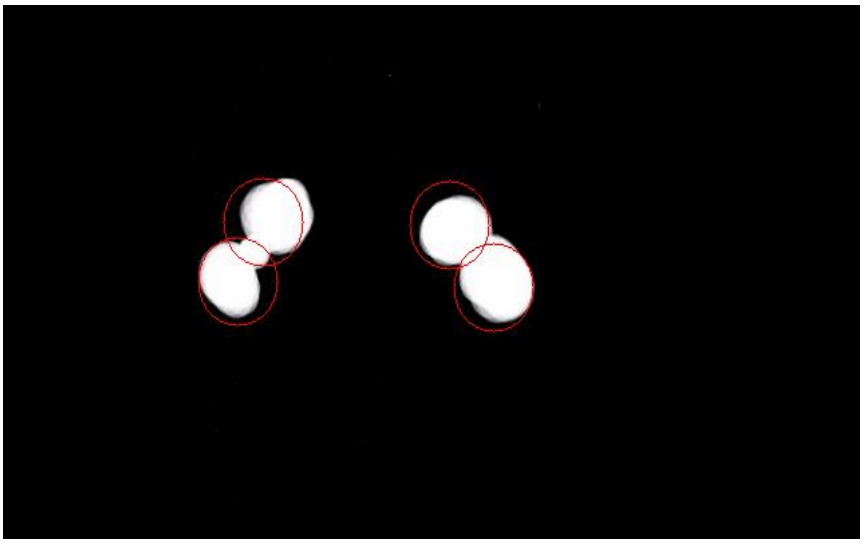

Figura 5-51. Imagen de vídeo virtual con seguimiento.

Track nº 0 Track_X:194 Track_Y:146 Track_Ancho:47 Track_Alto:48 Track_Velocidad X:-4 Track_Velocidad Y:-46 Visibilidad:1 Contador Inactividad:0 Edad:5 Track se elimina:0	Track nº 1 Track_X:333 Track_Y:148 Track_Ancho:47 Track_Alto:49 Track_Velocidad X:2 Track_Velocidad Y:-44 Visibilidad:1 Contador Inactividad:0 Edad:5 Track se elimina:0
Track nº 2 Track_X:175 Track_Y:186 Track_Ancho:45 Track_Alto:46 Track_Velocidad X:1 Track_Velocidad Y:41 Visibilidad:1 Contador Inactividad:0 Edad:5 Track se elimina:0	Track nº 3 Track_X:366 Track_Y:190 Track_Ancho:50 Track_Alto:46 Track_Velocidad X:-5 Track_Velocidad Y:43 Visibilidad:1 Contador Inactividad:0 Edad:5 Track se elimina:0

Figura 5-52. Datos de los tracks.

Por último, en 5.53, todos los tracks salen del estado de cross-tracking y se asocian con normalidad con las regiones correspondientes. En las imágenes siguientes los tracks acaban con normalidad y se eliminan todos liberando los cuatro huecos que cada uno ocupaba en la matriz de tracks. En 5.54 se ve la información de cada uno.

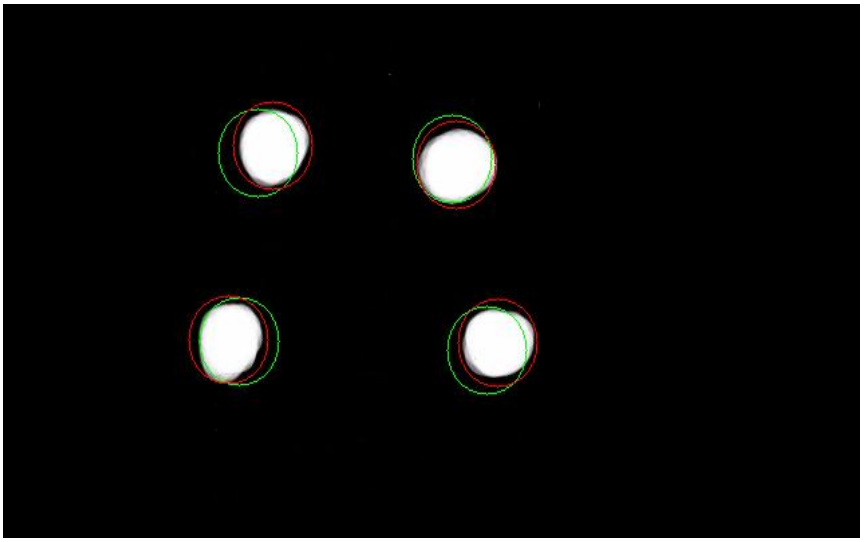

Figura 5-53. Imagen de vídeo virtual con seguimiento.

Track nº 0 Track_X:201 Track_Y:95 Track_Ancho:51 Track_Alto:52 Track_Velocidad X:7 Track_Velocidad Y:-51 Visibilidad:1 Contador Inactividad:0 Edad:6 Track se elimina:0	Track nº 1 Track_X:338 Track_Y:108 Track_Ancho:59 Track_Alto:50 Track_Velocidad X:5 Track_Velocidad Y:-40 Visibilidad:1 Contador Inactividad:0 Edad:6 Track se elimina:0
Track nº 2 Track_X:168 Track_Y:226 Track_Ancho:46 Track_Alto:53 Track_Velocidad X:-7 Track_Velocidad Y:40 Visibilidad:1 Contador Inactividad:0 Edad:6 Track se elimina:0	Track nº 3 Track_X:369 Track_Y:228 Track_Ancho:52 Track_Alto:46 Track_Velocidad X:3 Track_Velocidad Y:38 Visibilidad:1 Contador Inactividad:0 Edad:6 Track se elimina:0

Figura 5-54. Datos de los tracks.

Al continuar el vídeo, aparecen dos nuevas regiones para las que se crean dos nuevos tracks en los dos primeros huecos liberados por la eliminación de los tracks anteriores. La imagen de 5.55 se corresponde con la segunda imagen desde la creación de estos. El track 0 es el de la izquierda abajo y el track 1 el de arriba a la derecha. Los datos de estos tracks vienen dados en 5.56.

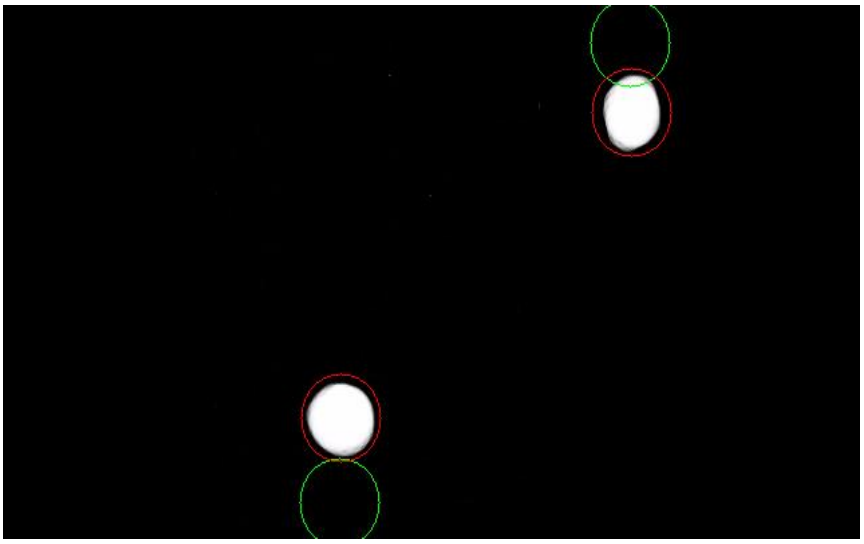

Figura 5-55. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1
Track_X:252	Track_X:469
Track_Y:278	Track_Y:72
Track_Ancho:50	Track_Ancho:41
Track_Alto:49	Track_Alto:50
Track_Velocidad X:1	Track_Velocidad X:1
Track_Velocidad Y:-57	Track_Velocidad Y:47
Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0
Edad:2	Edad:2
Track se elimina:0	Track se elimina:0

Figura 5-56. Datos de los tracks.

En la imagen siguiente, en 5.57, se da un error de segmentación total y ninguna de las dos regiones que se esperaban se segmentan por lo que los tracks, al igual que para el cross-tracking, se utilizan las posiciones predichas en las imágenes en las que se de este error hasta que se segmenten regiones a las que poder asociarse. En 5.58 se ven los datos de los tracks 0 y 1.

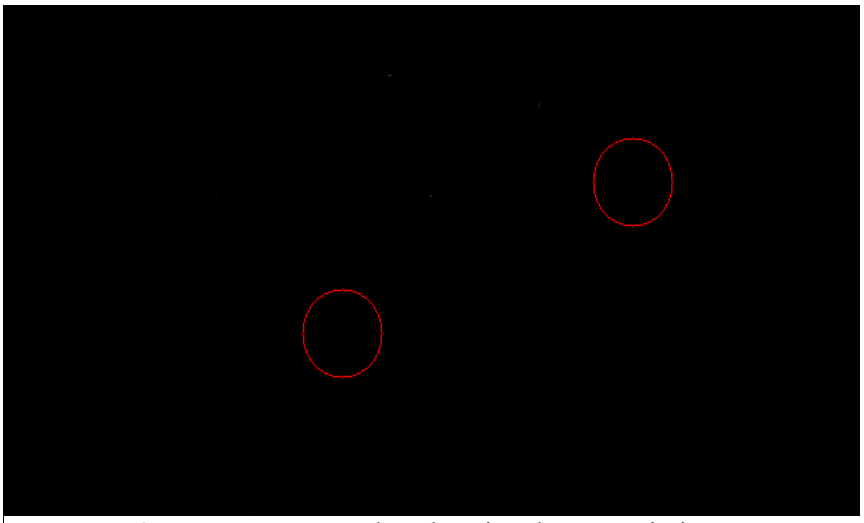

Figura 5-57. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1
Track_X:253	Track_X:470
Track_Y:221	Track_Y:119
Track_Ancho:45	Track_Ancho:46
Track_Alto:44	Track_Alto:45
Track_Velocidad X:1	Track_Velocidad X:1
Track_Velocidad Y:-57	Track_Velocidad Y:47
Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0
Edad:3	Edad:3
Track se elimina:0	Track se elimina:0

Figura 5-58. Datos de los tracks.

En 5.59, se ve que el track 0 ya se asocia con una región que se segmenta, pero sin embargo el track 1 sigue sin asociarse y por tanto siguen usándose sus posiciones predichas para avanzar. Los datos de estos vienen en 5.60.

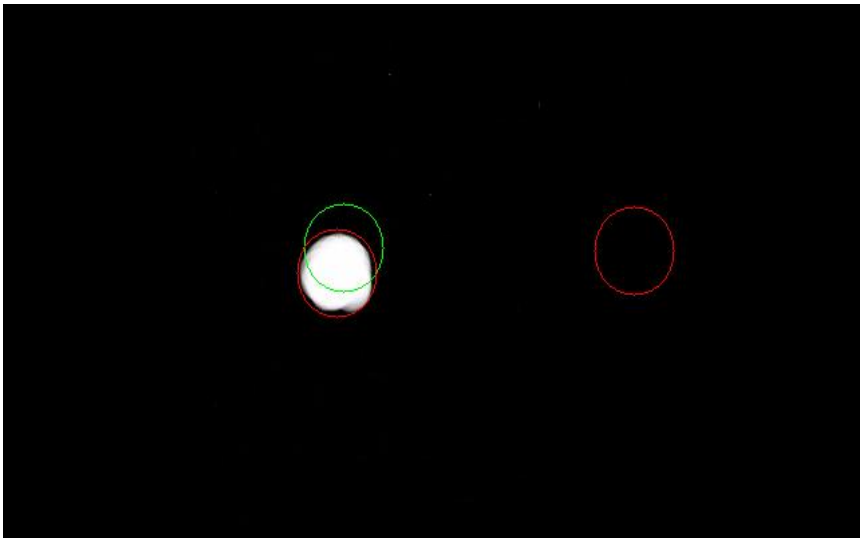

Figura 5-59. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1
Track_X:249	Track_X:471
Track_Y:181	Track_Y:166
Track_Ancho:53	Track_Ancho:46
Track_Alto:52	Track_Alto:45
Track_Velocidad X:-4	Track_Velocidad X:1
Track_Velocidad Y:-40	Track_Velocidad Y:47
Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0
Edad:4	Edad:4
Track se elimina:0	Track se elimina:0

Figura 5-60. Datos de los tracks.

En la imagen de 5.61, el track 0 sigue asociándose y el track 1 ya encuentra una región que se segmenta a la que asociarse. La figura 5.62 muestra los datos.

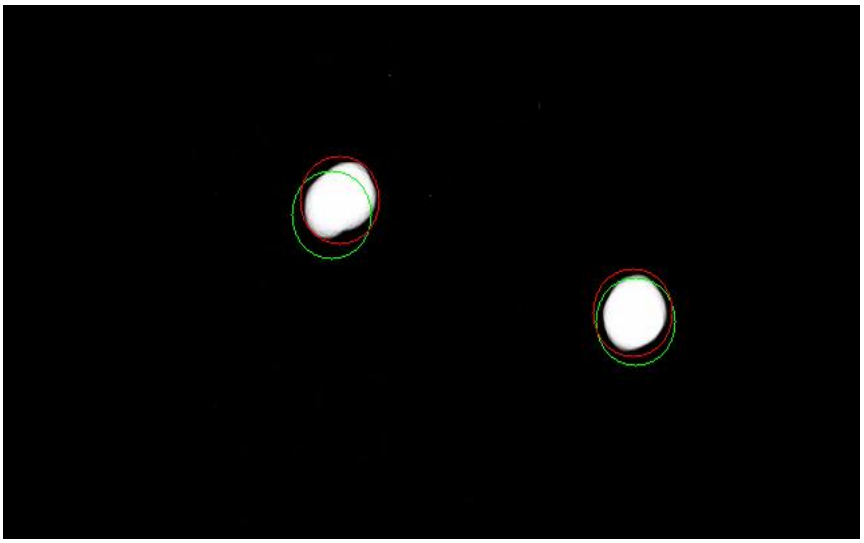

Figura 5-61. Imagen de vídeo virtual con seguimiento.

Track nº 0	Track nº 1
Track_X:251	Track_X:470
Track_Y:131	Track_Y:207
Track_Ancho:52	Track_Ancho:47
Track_Alto:51	Track_Alto:51
Track_Velocidad X:2	Track_Velocidad X:-1
Track_Velocidad Y:-50	Track_Velocidad Y:41
Visibilidad:1	Visibilidad:1
Contador Inactividad:0	Contador Inactividad:0
Edad:5	Edad:5
Track se elimina:0	Track se elimina:0

Figura 5-62. Datos de los tracks.

Por último, se muestran en 5.63, 5.64 y 5.65 tres imágenes consecutivas del track de un objeto cuyas regiones tienen errores de segmentación parcial. Se observa en 5.64 y 5.65 tal y como se explicó en el anterior capítulo, que de las diferentes regiones, el track se asocia con la de mayor tamaño.

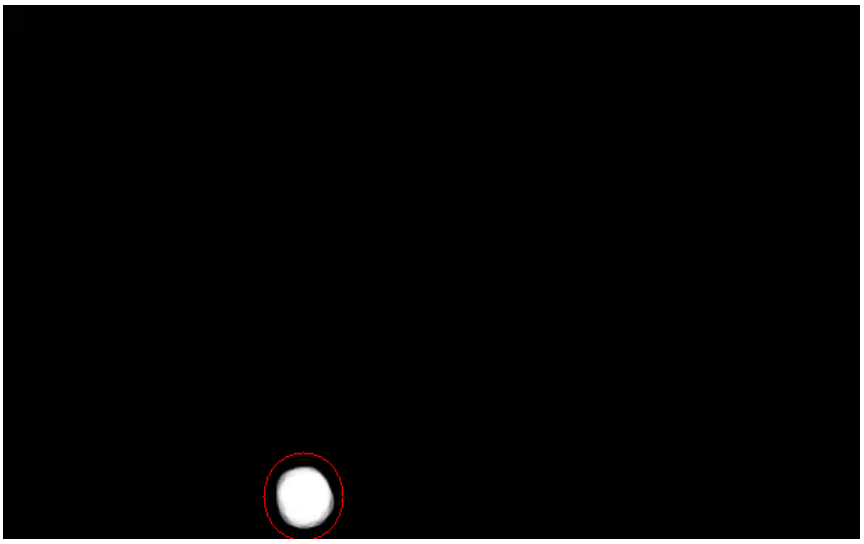

Figura 5-63. Imagen de vídeo virtual con seguimiento.

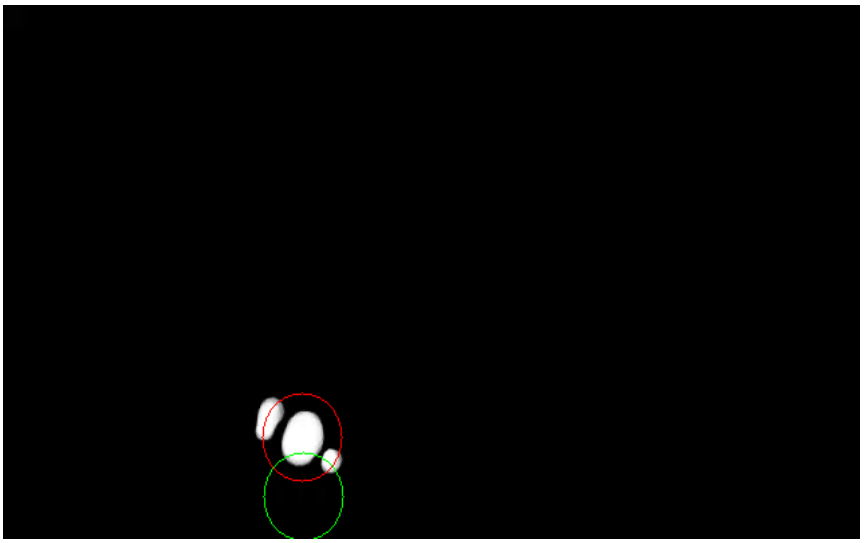

Figura 5-64. Imagen de vídeo virtual con seguimiento.

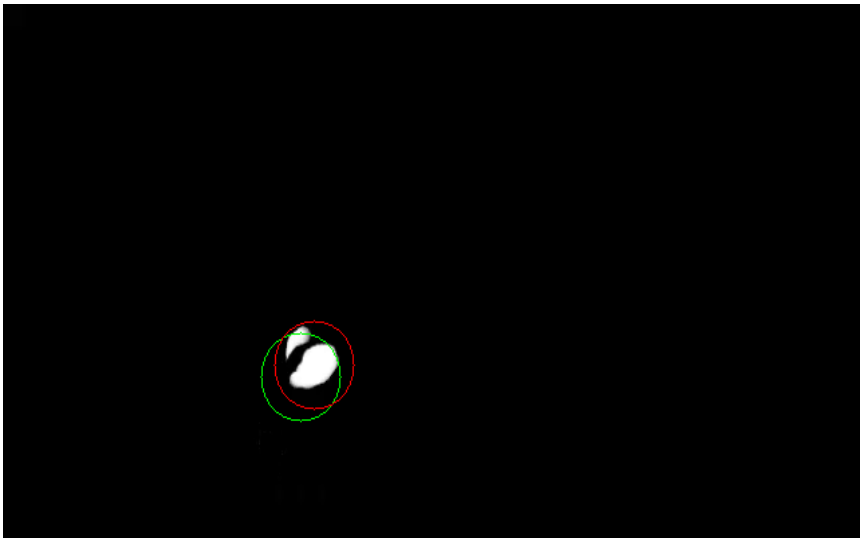

Figura 5-65. Imagen de vídeo virtual con seguimiento.

5.3 Experimentos de validación

Por último en este capítulo, se muestran los resultados de poner en funcionamiento el programa completo. Se prueban vídeos reales a los que se les aplica el procesamiento de imágenes para obtener la sustracción de objetos en movimiento y posteriormente realizar el seguimiento de dichos objetos.

Al tratarse de vídeos reales, aparte de resultados correctos, aparecen errores no contemplados a la hora de realizar el algoritmo de seguimiento. Estos errores se observarán para así saber cómo continuar construyendo el algoritmo y hacerlo más robusto a la hora de clasificar los tracks.

A continuación se prueba el algoritmo con el primer vídeo usado en los experimentos de segmentación. En la figura 5.66 se ve una imagen del vídeo en la que existe un track ya creado y a la vez se está creando uno nuevo. En 5.67 se pueden ver los datos de cada uno de estos.

Figura 5-66. Imagen de vídeo real con seguimiento.

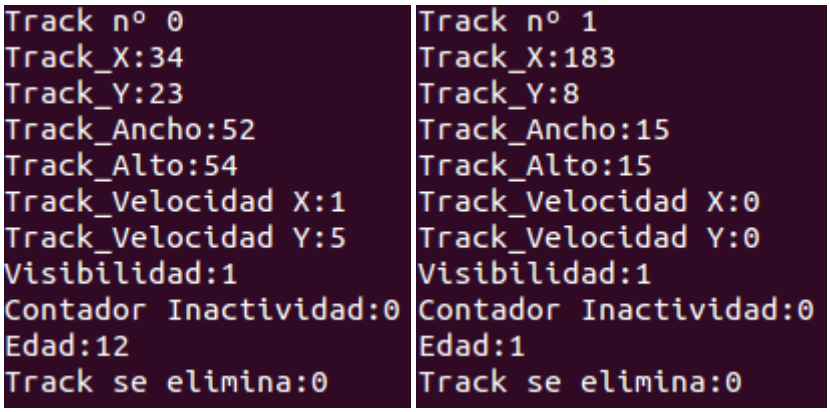

Figura 5-67. Datos de los tracks.

Puede verse en 5.68 y 5.70, que los tracks se asocian y actualizan con normalidad ya que la segmentación tiene buenos resultados.

Figura 5-68. Imagen de vídeo real con seguimiento.

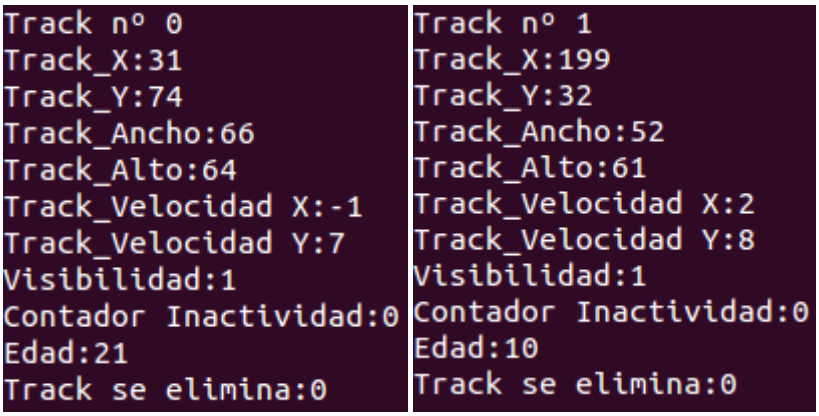

Figura 5-69. Datos de los tracks.

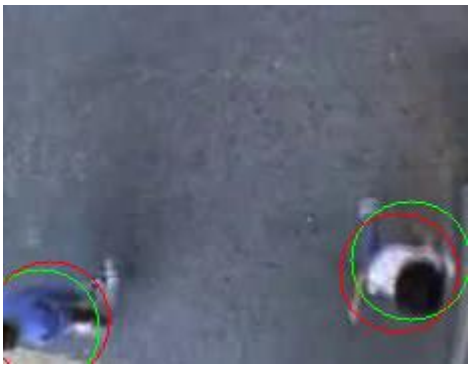

Figura 5-70. Imagen de vídeo real con seguimiento.

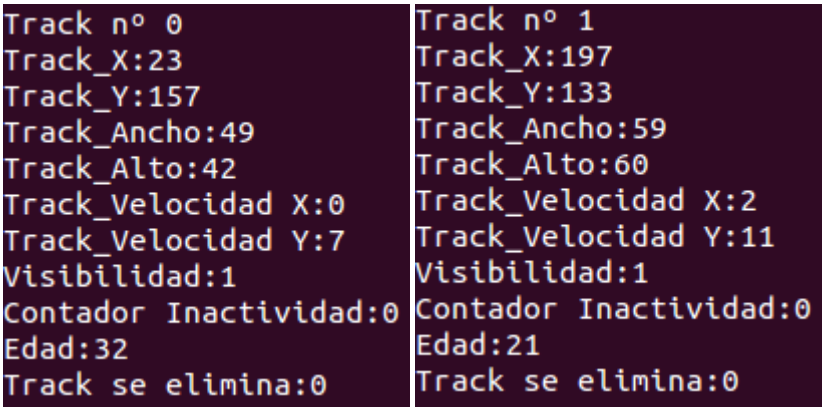

Figura 5-71. Datos de los tracks.

Ambos tracks acaban y también se eliminan con normalidad dejando cada uno un hueco libre. En la imagen mostrada en 5.72 el track 0 se eliminó hace unas cuantas imágenes, mientras que el 1 aún está contando el número de imágenes inactivas para eliminarse. En 5.72 también existe un track 2 que se puede ver arriba a la derecha. Este se asocia con normalidad.

Figura 5-72. Imagen de vídeo real con seguimiento.

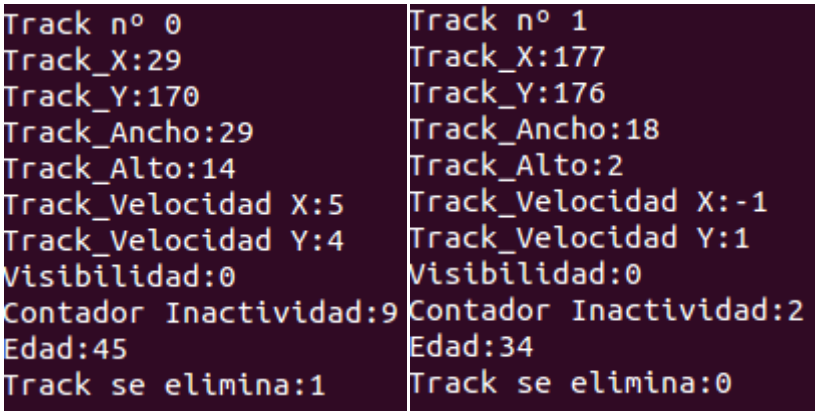

Figura 5-73. Datos de los tracks.

Para el caso visto arriba el seguimiento funciona correctamente, pero no siempre va a ser así. Si la imagen dada por el bloque de procesamiento no es suficientemente buena, pueden darse malas clasificaciones de tracks y llegar al error. Por ejemplo, en las figuras 5.74 y 5.75 se observa a una persona que entra en la zona de paso fragmentada en varias regiones. Esto provoca que se cree un track para cada región.

Figura 5-74. Segmentación de imagen real.

Figura 5-75. Imagen de vídeo real con seguimiento.

También puede provocar que se creen falsas clasificaciones de tracks. En el caso de 5.76 y 5.77, dos de los tracks que se crearon erróneamente en la imagen anterior, ahora se clasifican como cross-tracking.

Figura 5-76. Segmentación de imagen real.

Figura 5-77. Imagen de vídeo real con seguimiento.

En 5.78 y 5.79 se ve la continuación de las imágenes anteriores. En esta, uno de los tres tracks es clasificado como acabado, otro se asocia correctamente y el último se asocia por error a uno de los fragmentos de la segmentación de esta misma persona, creando así dos tracks para un mismo objeto.

Figura 5-78. Segmentación de imagen real.

Figura 5-79. Imagen de vídeo real con seguimiento.

Lo ocurrido anteriormente puede conllevar que al segmentarse correctamente dicha persona se clasifiquen de manera incorrecta otra vez en imágenes posteriores. Por ejemplo, en 5.80 y 5.81 se observa que se clasifican los dos tracks anteriores como cross-tracking por culpa de la mala clasificación y creación de tracks al entrar el objeto en la zona de paso.

Figura 5-80. Segmentación de imagen real.

Figura 5-81. Imagen de vídeo real con seguimiento.

Por último en este capítulo se observan algunos de los resultados del seguimiento en el otro vídeo real usado en el apartado de experimentos de segmentación. De nuevo y al igual que en el anterior, cuando la segmentación es buena, el seguimiento va a ser correcto pero en el caso de que la segmentación no sea lo suficientemente precisa, van a existir errores a la hora de la clasificación de los tracks.

En la figura 5.82 se observa una imagen en la que hay una persona en movimiento y al ser la segmentación buena, se realiza el seguimiento de forma precisa. Los datos del track vienen dados en 5.83.

La imagen de 5.84 y sus datos en 5.85 se corresponden con el mismo track de 5.82 en un punto más avanzado de la trayectoria. Como puede verse, el seguimiento sigue siendo bueno.

Dicho track acaba de manera correcta como se ve en 5.86. Los datos de 5.87 se corresponden con la información del track en la imagen 5.86, mientras que en 5.88 se corresponden a cuando este se elimina.

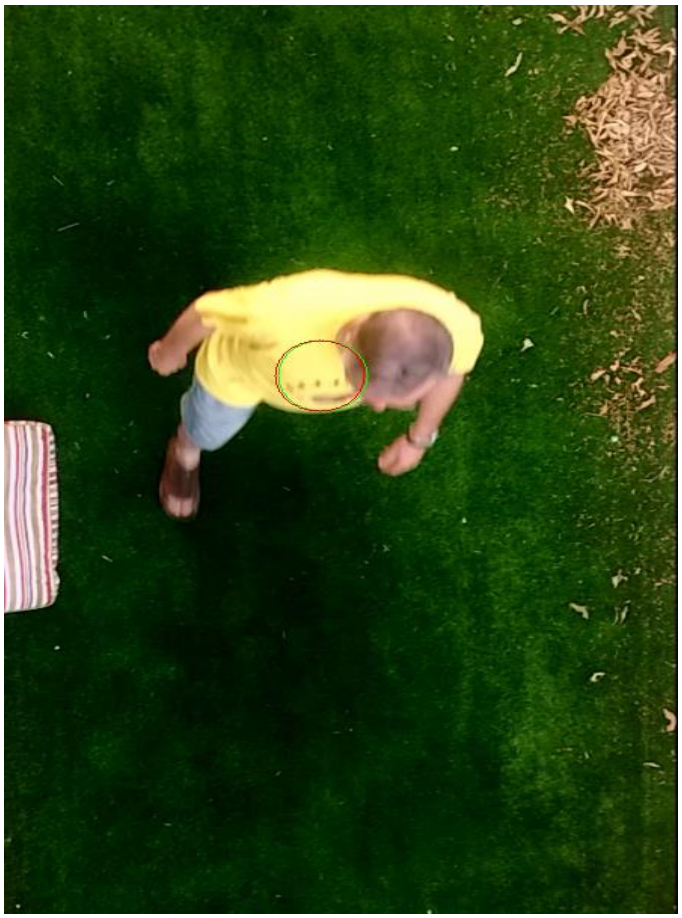

Figura 5-82. Imagen de otro vídeo real con seguimiento.

```
Track nº 1
Track_X:209
Track_Y:310
Track_Ancho:243
Track_Alto:241
Track_Velocidad X:-3
Track_Velocidad Y:32
Visibilidad:1
Contador Inactividad:0
Edad:17
Track se elimina:0
```

Figura 5-83. Datos del track.

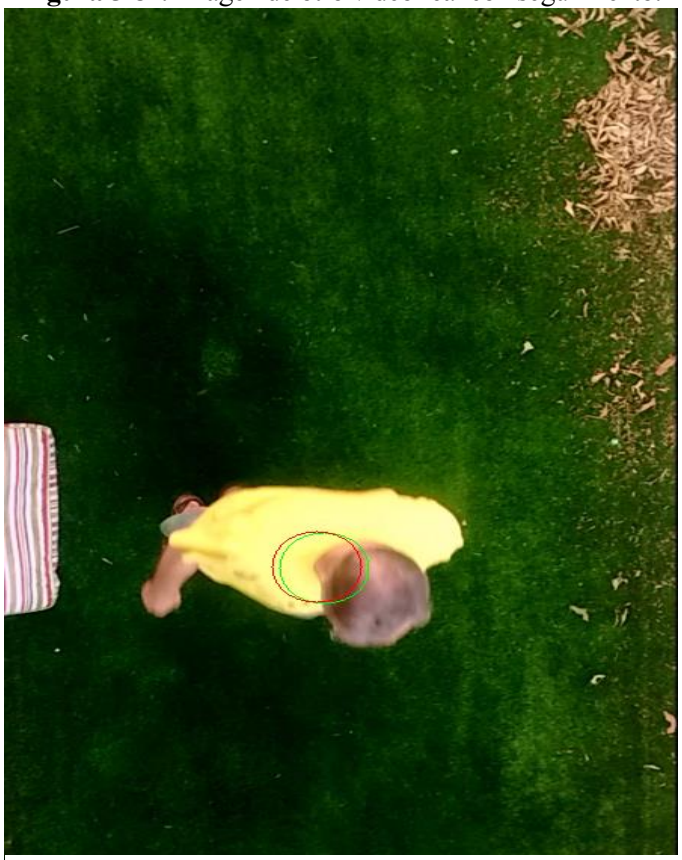

Figura 5-84. Imagen de vídeo real con seguimiento.

```
Track nº 1
Track_X:207
Track_Y:468
Track_Ancho:236
Track_Alto:164
Track_Velocidad X:-1
Track_Velocidad Y:28
Visibilidad:1
Contador Inactividad:0
Edad:23
Track se elimina:0
```

Figura 5-85. Datos del track.

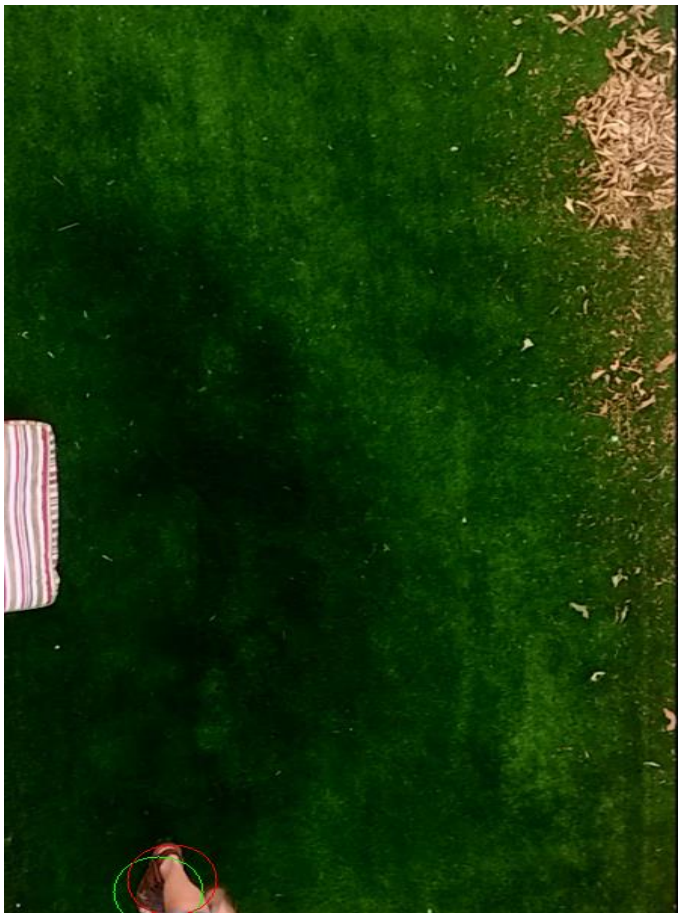

Figura 5-86. Imagen de vídeo real con seguimiento.

```
Track nº 1
Track_X:111
Track_Y:733
Track_Ancho:57
Track_Alto:62
Track_Velocidad X:-16
Track_Velocidad Y:-2
Visibilidad:1
Contador Inactividad:0
Edad:39
Track se elimina:0
```

Figura 5-87. Datos del track.

```
ACTUALIZACIÓN
Track_xk:113
Track_yk:748
Track_Width:34
Track_Height:25
Track_Velocity X:5
Track_Velocity Y:12
Visibilidad:0
Contador Inactividad:3
Edad:46
Track se elimina:1
```

Figura 5-88. Datos del track cuando se elimina.

Sin embargo, y como ya pasaba con el anterior vídeo, el seguimiento no siempre da buenos resultados. Aparecen errores que llevan a la mala clasificación de tracks o a la falsa creación de estos.

En la imagen mostrada en 5.89 y 5.90 se ve como una persona está entrando en la zona de paso. El problema es que al segmentar, la persona viene representada en varios fragmentos. Esto causa que se cree un track para cada fragmento de la segmentación.

En 5.91 y 5.92 se ve una imagen de la misma persona en otro punto de su trayectoria. Esta sigue segmentada en varios fragmentos y por tanto se continúa arrastrando el problema generado en la creación de los tracks, teniendo así varios tracks para una sola persona.

En el caso de 5.93 y 5.94, el seguimiento realizado a esta persona es correcto durante todo el desplazamiento de esta en la zona de paso. Salvo al final, donde la segmentación falla y existen varios fragmentos que deberían ser una única región. Esto provoca que se creen nuevos tracks para estos pequeños fragmentos cuando no deberían crearse. La diferencia con el caso anterior de que la persona entre segmentada en varias partes, es que en este caso, los tracks que se creen, al no asociarse con ninguna otra región, se eliminarán sin provocar más errores.

Figura 5-89. Segmentación de imagen real.

Figura 5-90. Imagen de vídeo real con seguimiento.

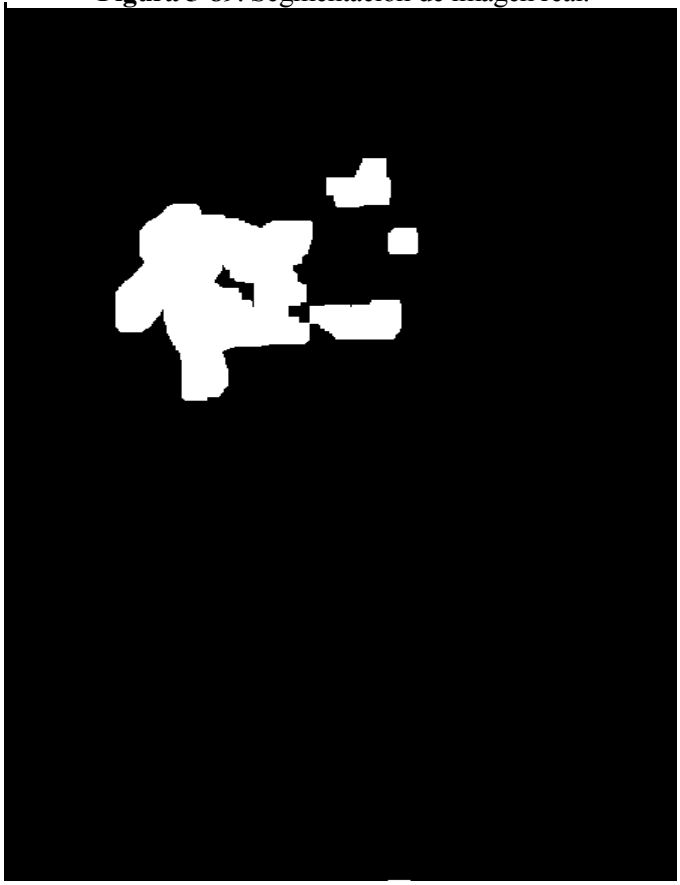

Figura 5-91. Segmentación de imagen real.

Figura 5-92. Imagen de vídeo real con seguimiento.

Figura 5-93. Segmentación de imagen real.

Figura 5-94. Imagen de vídeo real con seguimiento.

5.4 Conclusiones

Una vez vistos los resultados de realizar experimentos sobre cada bloque de forma independiente y sobre el conjunto de ambos, se concluye que el mayor problema radica en que la segmentación no sea lo suficientemente precisa. Ya que si no lo es, genera que el seguimiento no de resultados fiables. A su vez, sabiendo que el bloque de procesamiento no va a resolver el problema de obtener una segmentación perfecta, es necesario tener en cuenta los errores habituales que se dan en esta operación a la hora de realizar el algoritmo de seguimiento para poder contrarrestar sus efectos.

También puede observarse que algunos errores que aparecen a la hora de la creación de tracks provocan errores posteriores en la clasificación de tracks. Estos no aparecerían si este tipo de error se tuviera en cuenta en el algoritmo de seguimiento.

Realizando un breve estudio estadístico en función del número de personas contadas, para el primer vídeo real probado, de las 17 personas que pasan, 12 de ellas son contadas correctamente. Esto significa que un 70.5% de estas personas son detectadas correctamente, mientras que un 29.5% no son detectadas o no lo suficientemente bien detectadas como para ser contadas. Cabe mencionar, que la mayoría de las personas bien contadas, tienen algún error de clasificación de track en algún momento de la trayectoria pero que se corrigen con el paso de las imágenes. También cabe decir que aparte de las personas contadas, se cuentan 5 personas no existentes, es decir, falsas cuentas provocadas por mala clasificación.

Para el segundo vídeo real probado, el de mayor resolución, se cuentan correctamente 14 de 21

personas, es decir, un 66%, mientras que 7 personas no se cuentan, un 33%. Y de nuevo, como pasaba antes, existen falsas cuentas de personas, en este caso 23, un número bastante elevado. Aparte de los vídeos mostrados en este documento, se han probado otros vídeos con la misma cámara y en el mismo lugar que el de este segundo vídeo. El resultado del seguimiento es bastante similar en todos los vídeos, incluso dando menos aciertos y más falsas cuentas. En otro de los vídeos, el número de personas contadas correctamente es 5 de 10, por tanto un 50% y de nuevo con muchas cuentas falsas.

Para obtener resultados más conclusivos, convendría probar el algoritmo con más vídeos y que a su vez, estos sean más largos en los que el número de muestras sea mayor, es decir, que pasen más personas a las que poder realizarle el seguimiento.

La mayoría de errores que se han podido observar a la hora de realizar los experimentos, tienen que ver con la mala segmentación y con que no se ha tenido en cuenta a la hora del diseño del algoritmo el caso de que una persona entre en la zona de paso fragmentada en varias regiones. Esto provoca que se cree para cada fragmento un track y se cuenten como objetos independientes cuando realmente no son más que parte de un mismo objeto. Por tanto, habría que habilitar en el algoritmo un nuevo caso de clasificación de tracks en el que se tuviera en cuenta que una persona u objeto puede entrar en la zona de paso fragmentado en varias regiones.

Se concluye que es necesario conseguir que la segmentación sea lo mejor posible para evitar errores en el seguimiento, pero que también es necesario tener en cuenta más tipos de errores en el algoritmo de seguimiento por si ocurrieran, aunque no sean habituales.

6 CONCLUSIONES Y TRABAJO FUTURO

A continuación se exponen las conclusiones generales obtenidas a lo largo del desarrollo de este proyecto, así como posibles mejoras y desarrollos futuros a partir de lo ya alcanzado para hacer un sistema de seguimiento más robusto frente a errores.

6.1 Conclusiones

En cada capítulo de este documento se han ido obteniendo distintas conclusiones. En el bloque de procesamiento cabe destacar que los resultados obtenidos, si bien, no son negativos, pueden ser mejorados, sobre todo en la generación del fondo y la segmentación. Es crucial obtener un buen fondo porque si no, cualquier error que aparezca en la imagen se arrastrará a lo largo del resto del algoritmo provocando así más errores. A la hora de realizar la segmentación, también se alcanza otro punto crucial, ya que al depender de parámetros, no se da la seguridad de que vaya a funcionar para todos los casos correctamente. Existen otros métodos de generación de fondos y segmentación libres de parámetros con los que se podría experimentar y valorar los resultados.

Los resultados obtenidos en el seguimiento dependen notablemente de la calidad de la imagen calculada en el algoritmo de procesamiento. Si esta resulta muy precisa, el seguimiento no tendrá mayores complicaciones pero si no es tan precisa y cuenta con errores de segmentación, el seguimiento no será bueno y existirán fallos. Sabiendo desde el principio que el procesamiento nunca va a resultar perfecto, hay que tener en cuenta a la hora del diseño del algoritmo los errores de segmentación más comunes que se dan en la imagen. Solo se han tenido en cuenta algunos de ellos, como ya se ha explicado. Por tanto, cabe la posibilidad de mejorar dicho algoritmo teniendo en cuenta más casos a parte de los ya mencionados.

6.2 Desarrollos futuros

A partir de las conclusiones que se obtienen del proyecto, se pueden conocer cuáles son buenos puntos por dónde continuar la investigación y el desarrollo del seguimiento de personas y todo lo que conlleva. A continuación se describen posibles desarrollos futuros:

- Respecto al procesamiento de imágenes, se deja como desarrollo futuro investigar nuevas formas de generar fondos y de segmentar imágenes que aporten resultados más definidos en comparación con el utilizado en este proyecto.
- En el seguimiento, sería necesario hacer más robusto el algoritmo, añadiendo nuevos casos a la clasificación de tracks evitando así que se produzcan tantos errores como pueden verse en el capítulo de experimentos.
- Por último, cabe mencionar que este proyecto sirve como fase previa a otro proyecto que está siendo desarrollado. En el cual, mediante el uso de Matlab se crea un entorno en el que se procesarán los datos entregados por el fichero generado en este proyecto así como los datos dados por unos sensores de rango también instalados en la misma zona de paso que la cámara. Se usarán métodos y características para compararlos y poder emparejarlos con el mínimo error posible y poder sacar distintos tipos de estadísticas.

ÍNDICE DE FIGURAS

ÍNDICE DE ALGORITMOS

ÍNDICE DE TABLAS

BIBLIOGRAFÍA

[1] Javier González Jiménez, *Visión por Computador*. Madrid: Paraninfo, 1999.

[2] Subhash Challa, Mark R. Morelande, Darko Musicki, Robin J. Evans, *Fundamentals of Object Tracking*. Cambridge, 2011.

[3] Wenhan Luo, Junliang Xing, Xiaoqin Zhang, Xiaowei Zhao, Tae-Kyun Kim, *Multiple Object Tracking*. 24 Sep 2014.

[4] M. Andriluka, S. Roth, and B. Schiele. *People-tracking-by-detection and people-detection-by-tracking*, p. 1–8, 2008.

[5] Zenon W. Pylyshyn, Ron W. Storm. Tracking multiple independent targets: evidence for a parallel tracking mechanism. *Spatial Vision*, 1988. Volumen 3, no 3, p. 179-197.

[6] *OpenCV Tutorials* [en línea]. Disponible en: *http://docs.opencv.org/doc/tutorials/tutorials.html*

[7] *Background Subtraction* [en línea]. Disponible en: *http://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html*

[8] *Motion-Based Multiple Object Tracking* [en línea]. Disponible en: http://es.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html