Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

SEGUIMIENTO DE OBJETOS SIMPLES SOBRE SECUENCIAS DE IMÁGENES. APLICACIÓN A HERRAMIENTA DE "PLAYER TRACKING"

Alumno: Fernando Fernández Burgos

Tutor: Prof. D. Raúl Mata Campos
Depto.: Ingeniería de Telecomunicación

Septiembre, 2017

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

Trabajo Fin de Grado

Seguimiento de objetos simples sobre secuencias de imágenes.

Aplicación a herramienta de "Player Tracking"

Septiembre, 2017.

Alumno: **Fernando Fernández Burgos** Tutor: **Raúl Mata Campos**

ÍNDICE

Contenido:

1- RESUMEN: 1

1.1-ABSTRACT: 2

2- INTRODUCCIÓN: 3

2.1- EL VÍDEO: 3

2.1.1- DEFINICIÓN: 3

2.1.2- CARACTERÍSTICAS RELACIONADAS CON EL ENTORNO DEL VÍDEO: 3

2.2.3- PRINCIPALES FORMATOS: 4

2.2- FUNCIONAMIENTO BÁSICO Y ETAPAS: 4

2.3- POSIBLES PROBLEMAS A TENER EN CUENTA: 5

3- OBJETIVOS Y MOTIVACIÓN: 6

4- ESTADO DEL ARTE: 6

4.1- HISTORIA: 6

4.2- SOBRE EL OBJETO: 7

4.3- ALGORITMOS Y MÉTODOS EMPLEADOS: 10

4.4- TECNOLOGÍA SPORTVU Y SIMILARES: 12

5- DESARROLLO: 15

5.1- INTRODUCCIÓN DEL DESARROLLO DE LA APLICACIÓN: 16

5.2 – FILTRO KALMAN: 18

5.2.1 – INTRODUCCIÓN: 18

5.2.2- COMPONENTES DEL FILTRO KALMAN: 19

5.2.3 – FUNCIONAMIENTO DEL FILTRO KALMAN: 20

5.3: ANÁLISIS DE BLOB: 21

5.4: FOREGROUND Y BACKGROUND: 22

5.4.1- FUNCIONAMIENTO DEL MODELO DE MÚLTIPLES GAUSSIANAS DE STAUFFER Y GRIMSON: 22

5.5- PROGRAMA MATLAB DESARROLLADO: 25

5.5.1: OBJ: 26

5.5.2- TRACKS: 28

5.5.3- LECTURA DE FRAMES: 29

5.5.4- FUNCIÓN DETECCIÓN: 29
5.5.5- FUNCIÓN NUEVA UBICACIÓN: 31
5.5.6- FUNCIÓN DE SEGUIMIENTO: 32
5.5.7- FUNCIÓN ACTUALIZA TRACKS ASIGNADOS: 33
5.5.8- FUNCIÓN ACTUALIZA TRACKS NO ASIGNADOS: 36
5.5.9- FUNCIÓN DE BORRADO DE TRACKS: 36
5.5.10- FUNCIÓN NUEVOS TRACKS: 37
5.5.11- FUNCIÓN DISPLAY: 38
5.6- INTERFAZ GRÁFICA DE USUARIO DE MATLAB: 39
7- RESULTADOS: 42
7.2- PARA DOS PERSONAS: 47
8- TRABAJO FUTURO: 49
9- CONCLUSIÓN: 51
BIBLIOGRAFÍA: 52
ANEXO 1: EQUIVALENCIA MEDIDA REAL EN MEDIDA PIXELS DEL VÍDEO: 54

1- RESUMEN:

La finalidad del presente proyecto consiste en la localización y seguimiento de un atleta en alguna disciplina deportiva y su posterior toma de estadísticas interesantes para los propios deportistas. El objetivo final será: El diseño de una interfaz donde se encuadre todo esto de la forma más intuitiva posible. La aplicación se desarrolla por completo en el entorno Matlab.

A lo largo de esta memoria se expondrá todo aquello relacionado con la inteligencia artificial, donde el enfoque se realizará en el tracking de objetos, las diferentes herramientas para la detección del movimiento de objetos, los algoritmos empleados, en especial el que se ha empleado: El filtro de Kalman, y por último se analizará los resultados y detallaremos los métodos utilizados en el TFG para alcanzar el objetivo final.

1.1- ABSTRACT:

The purpose of this project is to locate and track an athlete in any sportive discipline and their subsequent collection of interesting statistics for the athletes themselves. The final objective will be: The design of an interface where all this is framed as intuitively as possible. The application is fully developed in the Matlab environment.

During this report will be exposed everything related to artificial intelligence, where the main focus will be on the tracking of objects, the different tools for the detection of movement of objects, the algorithms used, especially that has been used: The Kalman filter, and finally we will analyse the results and detail the methods used in this final project to reach the final goal.

2- INTRODUCCIÓN:

El seguimiento de objetos o tracking se refiere a la realización de una serie de mediciones o estimaciones para determinar la trayectoria, ubicación y características del objetos/s de interés a través de un sensor como puede ser un radar, sonar, cámara, sensor de infrarrojos, micrófono, ultrasonido o cualquier otro sensor que pueda utilizarse para recopilar información del entorno. Pertenece a la rama de Inteligencia Artificial.

En su forma más simple, el seguimiento puede definirse como el problema de estimar la trayectoria de un objeto en la imagen mientras se mueve alrededor de una escena. En otras palabras, un rastreador asigna etiquetas a los objetos rastreados en diferentes frames de un vídeo.

Lo más común es encontrarse un sistema compuesto por una o varias cámaras y la mayoría de éstos se emplea en cámaras estáticas, pero existen determinadas situaciones en las que se necesita una solución con una cámara en movimiento, debido por ejemplo a posibles vibraciones en el lugar donde se encuentra situada la cámara (plataforma) o porque se quiere detectar objetos en áreas de gran tamaño o de difícil acceso, una de las principales ventajas es la variación de la altura de la cámara.

2.1- EL VÍDEO:

2.1.1- DEFINICIÓN:

Se trata de la tecnología utilizada para a capturar, grabar, procesar, transmitir y reproducir una secuencia de imágenes de una escena en movimiento, es decir se trata de la exposición de un número de imágenes o fotogramas por segundo que corresponde a los frames por segundo (fps).

2.1.2- CARACTERÍSTICAS RELACIONADAS CON EL ENTORNO DEL VÍDEO:

- Dimensiones del vídeo: Es el tamaño del video (ancho x alto) expresado en píxeles. Por ejemplo la calidad HD tiene una dimensión de 1290x1080 píxeles.

- Velocidad de transmisión (bitrate): define la cantidad de espacio en bits que ocupa un segundo de duración de ese video, en el caso de los vídeos proporcionados para este TFG son 30 fps, es decir 30 fotogramas por segundo.

- Códec: es un algoritmo especial que reduce el número de bytes que ocupa un archivo de video, se requiere el mismo códec para ser decodificados y reproducidos.

- Proporción aspecto: Se trata de la proporción entre la anchura y altura de un video que cuando se reproduce se suele mantener por defecto esta proporción para evitar deformación de las imágenes, por lo general se trabaja con 4:3 y 16:9.

2.2.3- PRINCIPALES FORMATOS:

Los más conocidos son:

- AVI (Audio Video Interleaved = Audio y Video Intercalado):

  • Es el formato estándar para almacenar video digital.
  • Puede contener vídeo de gran calidad. Sin embargo el peso del archivo resulta siempre muy elevado. Aún así, puede ser visualizado por la mayoría de los reproductores.
  • Admite distintos códecs de compresión, uno de los más comunes es DV (Digital Video).

- MPEG (Moving Pictures Expert Group = Grupo de Expertos de Películas):

  • Se trata de un formato estándar para la compresión de video digital.
  • Los archivos tienen como extensión tanto .mpg como .mpeg.
  • Admite distintos tipos de códecs de compresión: MPEG-1 (calidad CD), MPEG-2 (calidad DVD), MPEG-3 (orientado al audio MP3) y MPEG-4 que utiliza la extensión .mp4 que es usada en los diferentes vídeos proporcionados para la elaboración del TFG.

- A parte de estos dos mencionados, existen más como FLV (flash videos) que utiliza el reproductor Adobe Flash para visualizar vídeo en Internet, también destaca MOV que es el formato de video desarrollado por Apple y destaca por la utilización de un códec propio que evoluciona con bastante rapidez.

2.2- FUNCIONAMIENTO BÁSICO Y ETAPAS:

Se parte de una secuencia de video en vivo o grabada y para poder llevar a cabo esta tarea se podrá emplear un algoritmo de búsqueda, o un conjunto de algoritmos, en el que se va a realizar un proceso a priori lento por toda la información que se encontrar en un vídeo, además el seguir a un objeto específico en una secuencia de imágenes donde existen otros objetos que también forman parte de la escena complica la acción. Por lo que una de las ventajas que se puede mencionar ya es que se puede realizar de manera mecánica, es decir sin contar con la acción humana, ya que no deberá observar durante un determinado tiempo el entorno donde se desarrolla la acción.

Existen tres etapas clave a partir de la secuencia de imágenes a tratar: detección de los objetos de interés, seguimiento de dichos objetos de un frame al siguiente y, por último, análisis del movimiento que permite reconocer algún patrón de comportamiento (ver figura 1 y 2).

A partir de este escenario específico, el objetivo es no perder de vista el objeto a seguir. Las cámaras de vídeo capturan información sobre los objetos en forma de un conjunto de píxeles. El seguidor de objetos relacionará el aspecto del objeto de interés y el valor de los píxeles correspondientes.

Figura 1: Tracking mono objetivo [6].

Figura 2: Tracking multi objetivo [6].

2.3- POSIBLES PROBLEMAS A TENER EN CUENTA:

A la hora de realizar el tracking pueden aparecer diversos problemas [4]:

- Pérdida de información de un objeto en una secuencia de 2D.
- Las imágenes son ruidosas, es decir aleatorias.
- Los cuerpos no son rígidos.
- Proceso de la información en tiempo real.
- Aparición de sombras o cambios de brillo de la luz.
- Uno de los problemas más importantes es a partir de la detección del objeto, asignar éstas, que se van produciendo a lo largo de la reproducción de un vídeo, al mismo objeto.

Por todo ello y para facilitar el trabajo se intenta que el movimiento sea continuo y sin un cambio abrupto o aplicando una serie de restricciones al tamaño y forma del objeto.

3- OBJETIVOS Y MOTIVACIÓN:

El objetivo de este Trabajo Fin de Grado (TFG) es realizar una aplicación de seguimiento de objetos simples, tipo "player tracking". Por lo tanto es importante familiarizarse con todas las posibilidades que este campo, inteligencia artificial, que puede llegar a ofrecer ya que es numeroso.

Me decanté por este proyecto por el simple hecho de que está muy ligado al deporte, que es un tema que me encanta, por lo que me gustaría realizar algo referente a esta cuestión. Para ello los pasos a seguir serán los siguientes:

- Revisión bibliográfica y estudio de las técnicas de análisis, procesamiento y reconocimiento orientadas al seguimiento de trayectorias.

- Estudiar librerías y herramientas comerciales existentes de características similares como SportVU, analizando las técnicas empleadas y su alcance.

- Desarrollo en MATLAB de algoritmos/herramientas/librería que integrarán las funciones básicas para los objetivos de seguimiento y análisis de objetos con la finalidad de un diseño de una aplicación sencilla para seguimiento del jugador (player tracking).

- Implementación y comprobación del correcto funcionamiento que se pretende alcanzar de la aplicación fundamental para conseguir el fin de este TFG.

- Desarrollo de la memoria, manuales y documentación habitual a la presentación de trabajo fin de grado cumpliendo con las normas básicas de estilo y estructura proporcionadas por la escuela.

4- ESTADO DEL ARTE:

4.1- HISTORIA:

Su origen se debe, en la mayoría de los casos, al ámbito militar cuyo objetivo era la localización de objetos en movimiento y anteponerse a posibles ataques e intentar prevenirlos [3].

Actualmente, se ha expandido a aplicaciones civiles donde podremos llegar a conseguir una gran diversidad, ya que este mundo está en evolución constante.

Algunos de sus usos más importantes son:

- Reconocimiento basado en movimiento: Identificación humana, detección automática de objetos…

- Supervisión o vigilancia automática: Sobre una escena para detectar actividades sospechosas.

- Indexado de video: Anotación automática de parámetros y recuperación de videos en bases de datos multimedia.

- Interacción ordenador-humano: reconocimiento de gestos, seguimiento de la boca al hablar, etc.

- Monitorización del tráfico: Obtención de estadísticas del tráfico en tiempo real para dirigirlo adecuadamente.

- Navegación de vehículos: Capacidades de planificación de rutas basada en video, evitar obstáculos…

4.2- SOBRE EL OBJETO:

La forma del objeto se puede definir como [5] [6]:

- Puntos: El objeto se representa por un punto, normalmente su centroide (Figura 3(a)) o por un conjunto de puntos (Figura 3(b)), normalmente puntos característicos. En general es adecuada para objetos que ocupan una región pequeña en una imagen. Se va a trabajar especialmente en este punto ya que vamos a tomar la referencia del centroide como punto de seguimiento, el centroide puede definirse como el punto correspondiente al centro geométrico de un objeto, pueden ser necesarias una, dos (en este trabajo, es decir coordenada x e y) o tres coordenadas para definir su posición exacta en el espacio.

- Formas geométricas básicas: Como rectángulos, elipses, etc... El movimiento se modela normalmente mediante translación, homografías y otras transformaciones afines. Es más adecuada para el tracking de objetos rígidos, tan bien se usa para trackear objetos no rígidos, por ejemplo, personas en una imagen.

- Siluetas o contornos: Define los límites de un objeto (Figura 3(g), (h)).

- Figura articulada: Los objetos articulados están compuestos de distintos elementos unidos por articulaciones. Por ejemplo, el cuerpo humano es un objeto articulado con torso, piernas, brazos, cabeza, etc... unidos por articulaciones. La relación entre las partes está gobernada por un modelo de movimiento cinemático, por ejemplo, el ángulo de la unión. A la hora de representar modelos articulados se pueden usar cilindros o elipses para representar las distintas partes tal como se muestra en la figura 3(e).

- Modelos esqueléticos: El esqueleto de un objeto se puede obtener mediante la aplicación de una transformación al eje medio de la silueta del objeto. Esta forma de representar un objeto se puede usar tanto para objetos articulados como rígidos (ver Figura 3(f)).

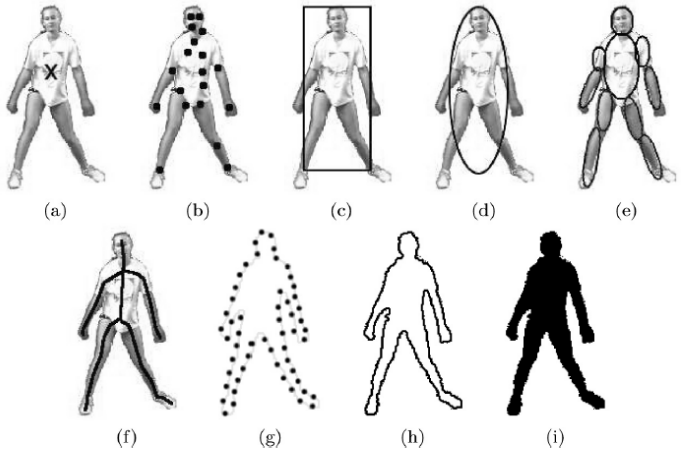


Figura 3: Diferentes formas de representación de objetos.

A la hora de la representación de la apariencia se puede emplear [5] [6]:

- Densidad de probabilidad de la apariencia del objeto: La estimación de la densidad de probabilidad de la apariencia del objeto puede ser paramétrica, como una Gaussiana o una mezcla de Gaussianas (realizado en este proyecto y se detallará en posteriores capítulos), o no paramétrica, como las ventanas de Parzen y los histogramas (Figura 4). Las densidades de probabilidad de las características de apariencia del objeto (color, textura) pueden ser calculadas a partir de las regiones de la imagen especificadas por los modelos de forma (regiones interiores de elipses o contornos).

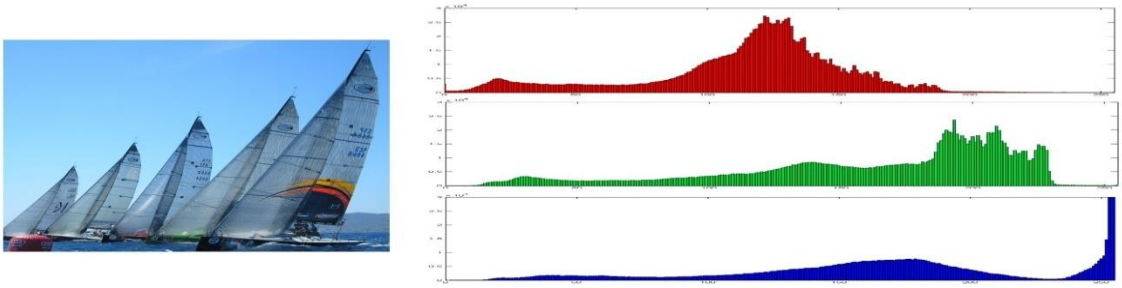


Figura 4: Histograma RGB.

- Templates: Las plantillas se extraen de figuras geométricas sencillas. Una ventaja es que llevan tanto información espacial como de apariencia. Sin embargo, sólo llevan la información de un solo punto de vista. Se debe emplear cuando se siguen objetos cuya posición en la imagen no cambie drásticamente durante el transcurso del tracking.

- Modelo de apariencia activa: Se modela de forma simultánea la forma del objeto y su apariencia. En el primer caso se define por una serie de puntos de referencia (De manera similar a la representación basada en contorno). Por cada punto de referencia, se almacena un vector de apariencia con información de color, textura o gradiente. Se requiere de una fase de entrenamiento, donde la forma y apariencia asociada, es aprendida de un conjunto de muestras. (Figura 5).

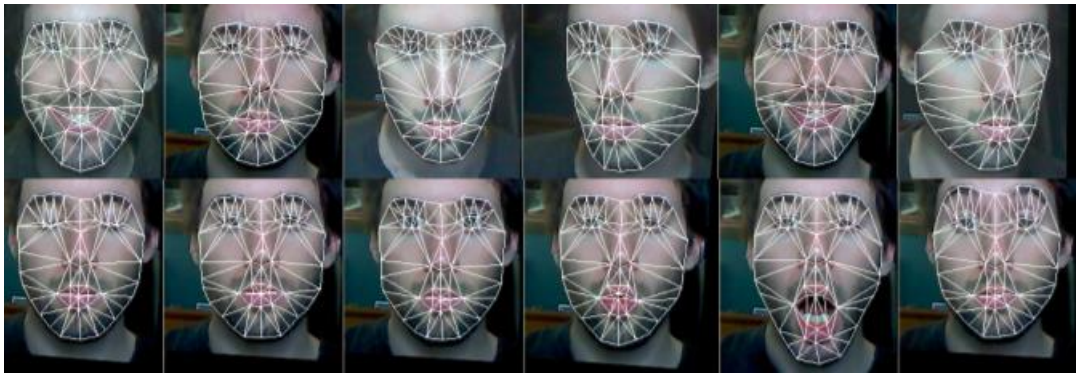


Figura 5: Modelado de apariencia activa.

- Modelo multivista: Estos modelos codifican varias vistas de un objeto. Una aproximación para representar las diferentes vistas de un objeto es, generar un subespacio de las vistas obtenidas. Uno de los métodos más empleados es darle un ángulo de rotación al objeto a estudiar.

Figura 6: Modelo multivista.

Para la selección de las características del objeto: Para poder distinguirse de forma clara en el espacio se va a localizar una serie de características, es decir del fondo de la imagen, o de otros posibles objetivos de apariencia similar.

- El color aparente de un objeto está influenciado principalmente por los factores físicos como son la densidad espectral de potencia de iluminación y las propiedades reflectantes de la superficie del objeto iluminado. En el procesado de imágenes, el espacio RGB es usado habitualmente. Sin embargo, el espacio de color RGB no es un espacio de color perceptualmente uniforme, ya que no corresponden con las diferencias perceptuales de color percibidas por el ojo humano. Usado en las representaciones de histograma (Figura 4).

- Bordes: Los límites de un objeto habitualmente generan cambios bruscos en las intensidades de una imagen. Una propiedad importante de los bordes es que son menos sensibles a los cambios de iluminación comparados con los espacios de color.

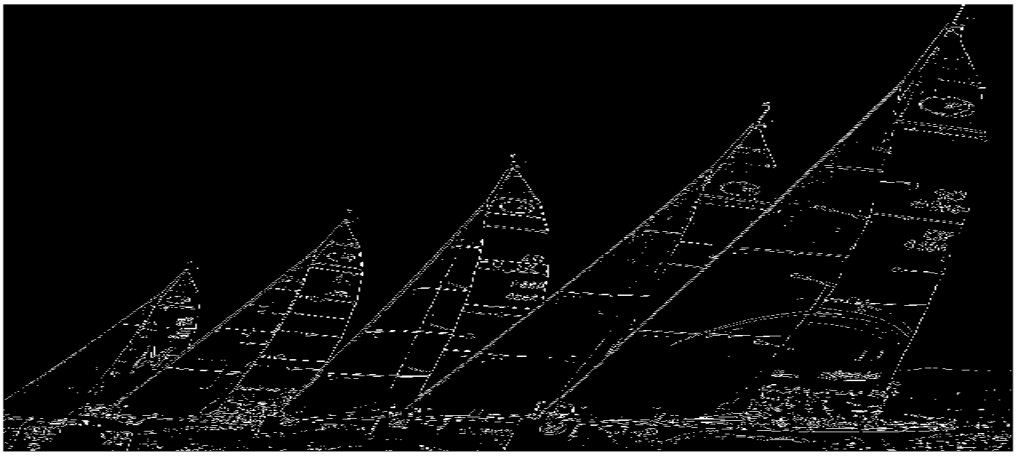


Figura 7: Detección de los bordes.

- Flujo óptico: Conjunto de vectores de desplazamiento que definen la translación de cada pixel dentro de una región. El flujo óptico se usa habitualmente como característica en aplicaciones de segmentación basada en movimiento.

- La textura: Es una medida de la variación de intensidad de una superficie que cuantifica propiedades tales como la suavidad y la regularidad. En comparación con el color, la textura requiere una etapa de procesamiento para generar los descriptores, aunque son menos sensibles a los cambios de iluminación de la imagen en comparación con el color.

4.3- ALGORITMOS Y MÉTODOS EMPLEADOS:

A partir de un tracker se puede localizar la trayectoria de un objeto a lo largo de un período de tiempo, localizando su posición en cada frame. Hay que distinguir entre las tareas de detectar el objeto y establecer una correspondencia (localizar dicho objeto en otra imagen) de forma separada o conjunta, para lo primero es preciso la utilización de un algoritmo de detección y lo segundo se estiman conjuntamente actualizando iterativamente la localización del objeto y la información de la región obtenidas de frames anteriores [8].

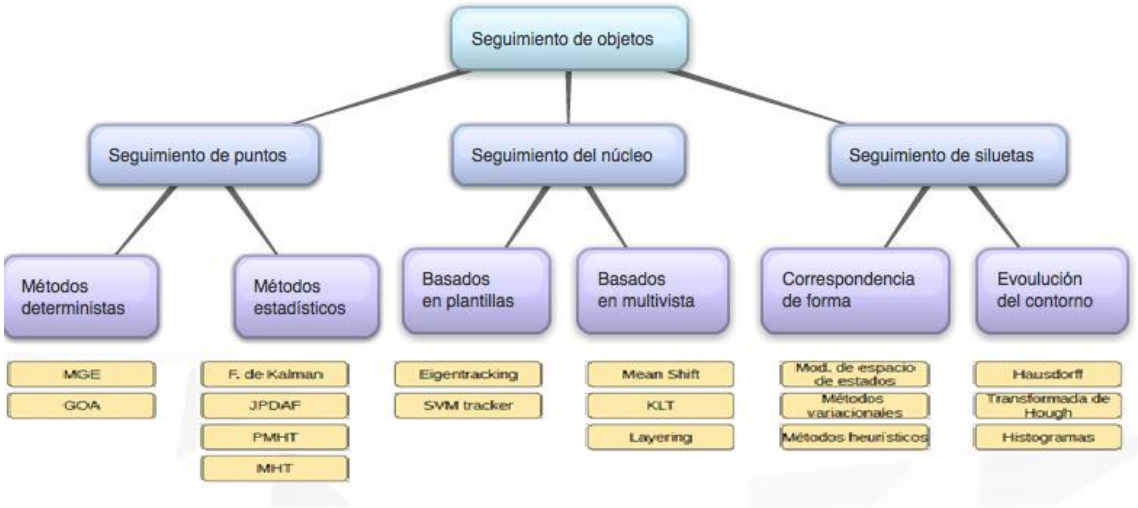


Figura 8: Algoritmos comunes empleados para el seguimiento.

- Seguimiento por puntos: Los objetos detectados en puntos y su asociación de se basa en el estado previo de su posición y de su movimiento. La aproximación requiere un mecanismo externo que detecte los objetos en cada frame, se trata un problema complicado, y más si existen oclusiones, errores de detección, o entradas y salidas de otros objetos.

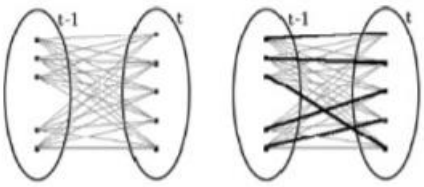


Figura 9: Seguimiento por puntos.

En la primera imagen aparecen todas las posibles asociaciones de un punto (objeto) del cuadro en t-1 con puntos (objetos) del cuadro en t y en la segunda conjunto único de asociaciones.

  - Método determinista: Definen la asociación de cada objeto en el plano t-1 con un objeto en el plano t usando un conjunto de restricciones de movimiento como por ejemplo la proximidad que asume que no hay un cambio drástico de posición entre frames, otra puede ser la rigidez de los cuerpos así las distancias entre dos puntos del mismo objeto se mantendrán.

  - Métodos estadísticos: A partir de conocer que los movimientos de los objetos pueden sufrir perturbaciones aleatorias. En estos algoritmos se tiene en cuenta la incertidumbre de las medidas y del modelo durante la estimación del estado del objeto. Empleando una aproximación de las propiedades del objeto tales como posición, velocidad y aceleración. Aquí entra el filtro Kalman que será empleado en el algoritmo desarrollado explicando su funcionamiento posteriormente.

- Seguimiento por el núcleo (Kernel): Se refiere a la forma del objeto y su apariencia. El seguimiento de un objeto se realiza mediante una evaluación del movimiento del kernel en frames sucesivos.

  - Método basado en plantillas: Son ampliamente usados por su simplicidad y bajo coste computacional. La aproximación más común es la denominada coincidencia de plantilla. Se basa en la búsqueda en toda la imagen una región similar a la plantilla del objeto. Se suele utilizar la correlación cruzada para medir la similitud, y características de color o intensidad para formar la plantilla.

        - Método de apariencia multivista: Representa la información obtenida del objeto en las observaciones más recientes y para ello es necesario el conocimiento del objeto desde diferentes vistas, ya que puede ser diferente, sino podría llevarnos al error de forma muy sencilla.

- Seguimiento de siluetas: Utilizan la información codificada dentro de la región del objeto, como los modelos de densidad de aparición o forma que se utilizan para generar mapas de bordes y entonces, se realiza un seguimiento de las siluetas mediante la búsqueda de:

        - Coincidencia de formas: Busca la silueta del objeto en el cuadro actual donde sólo se traslada al siguiente. La búsqueda se realiza calculando la similitud del objeto con el modelo generado a partir de la silueta del objeto obtenida en el cuadro anterior.

        - Evolución del contorno: En esta aproximación se evoluciona iterativamente un contorno inicial del cuadro previo hasta su nueva posición en el cuadro actual, se requiere que parte del objeto en el cuadro actual solape con la región del objeto del cuadro previo, se puede ver en la figura 10.

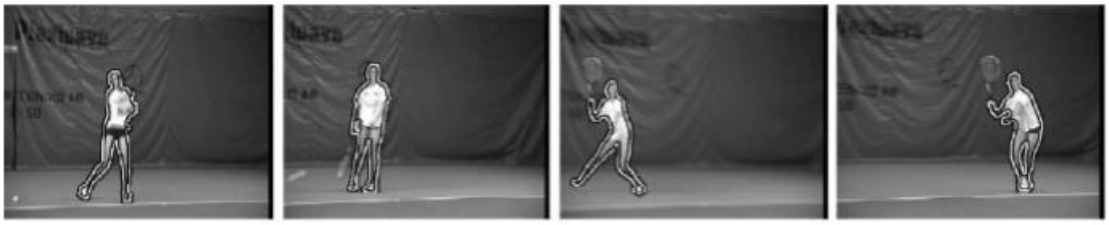


Figura 10: Tracking por evolución del contorno.

4.4- TECNOLOGÍA SPORTVU Y SIMILARES:

El BigData consiste en la recogida de una gran cantidad de datos con un objetivo concreto, es una herramienta muy útil para las finanzas, el marketing o incluso el sector de la medicina [7].

Pero ahora se ha introducido de forma muy directa en el deporte, ya que se trabaja con una gran cantidad de datos en cuanto a estadísticas se refiere en tiempo real principalmente gracias a lo que conocemos como "Internet of things" (uso de dispositivos inteligentes). La forma más común de la recogida de datos es a través de cámaras con detección de movimiento. La más popular es la solución de STATS, un proveedor de tecnología de datos y de análisis deportivo. Su sistema de seguimiento SportVU.

Se trata de un sistema de cámaras que divide la pista en una cuadrícula tridimensional y graba veinte imágenes por segundo de cada jugador para así medir y analizar cada movimiento, la biomecánica del jugador, si duerme bien o mal, su ritmo cardiaco a tiempo real e incluso su nivel de estrés a la hora de tirar un penalti. Por lo que su misión principal es ayudar a optimizar el potencial de un conjunto o de un jugador en especial. También incluye la posibilidad de crear gráficos donde aparezca diferente tipo de información. Esta tecnología está muy empleada en la NBA.

Unas de las últimas innovaciones emergentes son las tecnologías de línea de gol y de seguimiento de la pelota. En el club alemán TSG Hoffenheim se está colocando sensores en las espinilleras donde se transmiten, analizan y almacenan utilizando SAP HANA, una plataforma de datos en memoria para análisis en tiempo real.

Pero como siempre existen una serie de competidores [7]:

- En España, el tema deportivo y en especial del fútbol, queda en manos de Mediapro y su sistema "Media Coach" el cuál no se puede comercializar a nivel internacional por un acuerdo con la Liga Profesional de Fútbol (LPF). Este sistema que es menos preciso, necesita un mayor despliegue de cámaras, pero tiene puntos a favor y una de sus ventajas, por ejemplo es la de señalar a nuestros cuatro defensas y el portero: con la cual se puede obtener gran información como la distancia que guardan entre ellos y el portero, cómo realizan la técnica del fuera de juego... Además de proporcionar los datos globales como SportVU.

Figura 11: Media Coach.

- Prozone: Es una empresa británica y uno de los pioneros en el mercado de esta industria cuyos servicios incluyen análisis de juegos en tiempo real, evaluación del desempeño… Como SportVU, se van a necesitar el mismo número de cámara que la tecnología española hablada anteriormente, pero la principal desventaja es que se requiere intervención humana para corregir el seguimiento y también para la anotación de los eventos de juego tales como: goles, pases…

- OptaPro5: Otra tecnología en la que es la mano humana la que lo dirige ya que se necesita hasta tres personas para el análisis, una de cada equipo y otra para supervisar el proceso y destaca por el uso de imágenes de radiodifusión televisiva.

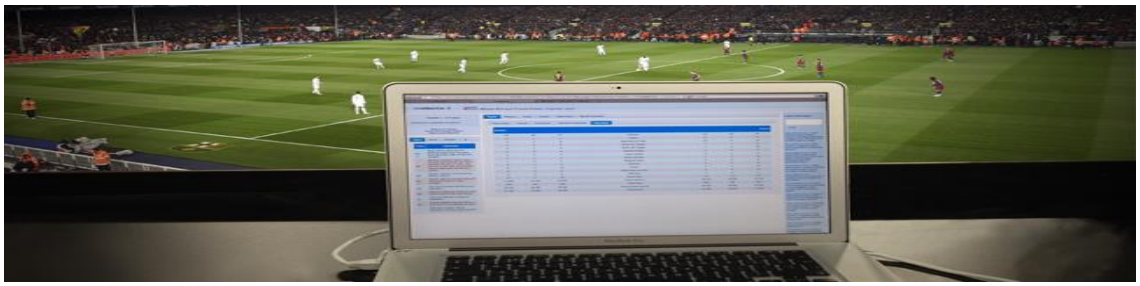


Figura 12: Optapro5.

- ChyronHego6: Es el más usado en los estadios de Inglaterra, a su favor se puede hablar que es capaz de extraer la posición 3D de todos los jugadores y la pelota en tiempo real bajo diferentes condiciones ambientales.

Figura 13: ChyronHero 6.

Por último también hay que hablar de dispositivos más individuales, cuya ventaja principal es el beneficio es propio, como Nike Fuelband, que sigue su movimiento y lo convierte en una puntuación, pide que se establezcan objetivos y continuamente aumentar esos niveles de actividad o Golf 360, aplicación de Nike, capaz de actuar como un entrenador con consejos en tiempo real puede subir sus puntuaciones a tablas de clasificación online y competir por los primeros puestos.

a) b)

Figura 14: a) Fuelband b) Aplicación Golf360

5- DESARROLLO:

A partir de este punto se centra en el desarrollo de la aplicación con el objetivo de alcanzar una aplicación que sea adecue a la intención final de este TFG y además de tener un correcto funcionamiento, se propone la siguiente metodología:

Figura 15: Estructura de la aplicación.

A su vez el sistema tracking estará compuesto:

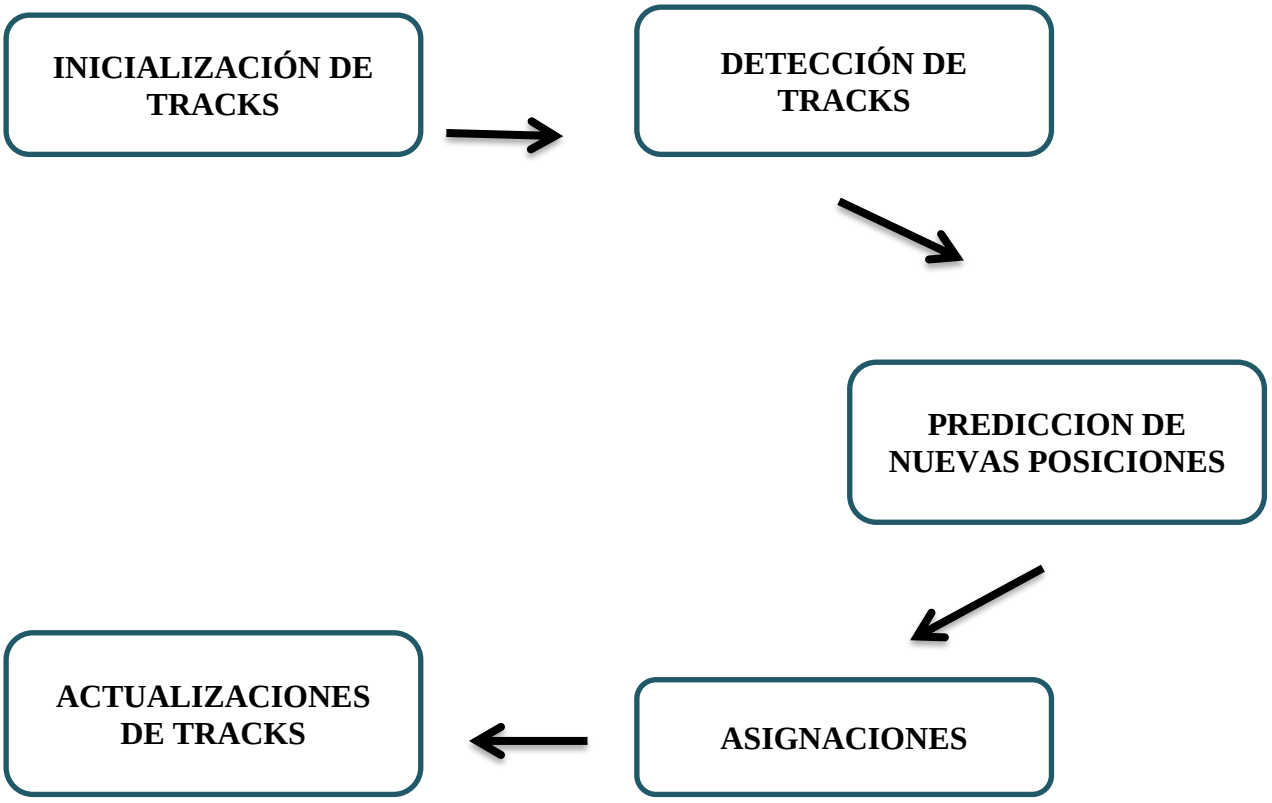
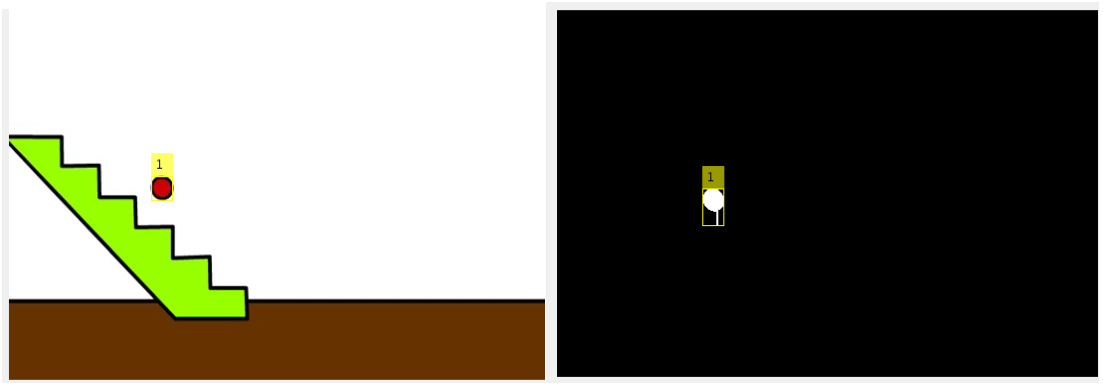


Figura 16: Esquema de funcionamiento del núcleo del sistema tracking desarrollado.

5.1- INTRODUCCIÓN DEL DESARROLLO DE LA APLICACIÓN:

A partir del código encontrado en la página oficial de Matlab, se ha fijado como punto de partida de la aplicación que se ha desarrollado, sobre la cual se ha modificado y añadido lo necesario para alcanzar los requisitos solicitados por parte de este TFG (ver: https://es.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html [9]). Pero antes de empezar estas modificaciones, se ha probado el funcionamiento de este algoritmo para su estudio previo con un video muy sencillo obtenido de YouTube donde se puede llegar a ver una pelota bajando las escaleras y el resultado obtenido es:



a) Visualizando detección b) Visualizando la máscara

Figura 17: Aplicación sobre video del algoritmo

Tras observar estos resultados que se adecuan a la propuesta del TFG, entonces se toma la decisión de ser el punto de partida después de observar otras ideas como el seguimiento por color, por ejemplo de camiseta, como este caso:

a) Visualización vídeo. b) Visualización de la detección.

Figura 18: Aplicación realizada con un algoritmo empleando seguimiento del color.

Desechado por diversos problemas como la necesidad de tener que crear un algoritmo para cada uno de los colores, y además la posibilidad de ver (como en la imagen) otros elementos del fondo de la misma tonalidad.

Por lo que finalmente se vuelve al código visto en la web y se procede a realizar las modificaciones necesarias. Pero antes de todo esto, se comenzó con la búsqueda de vídeos apropiados de alguna de las diferentes modalidades deportivas existentes, pero supuso un problema no encontrar un vídeo acertado en las diferentes webs que ofrece el mundo de Internet, se decide entonces ir a grabar vídeos con los cuales trabajar, y tras el posterior estudio realizado del algoritmo observe que un punto necesario es un fondo estático y el uso de una cámara de grabación estática por lo que con la ayuda de otras personas, y además de realizar la solicitud de los permisos adecuados se fue a realizar una serie de grabaciones al pabellón Julián Jiménez del Parque Deportivo de San José de Linares, Jaén.

Figura 19: Lugar de grabación.

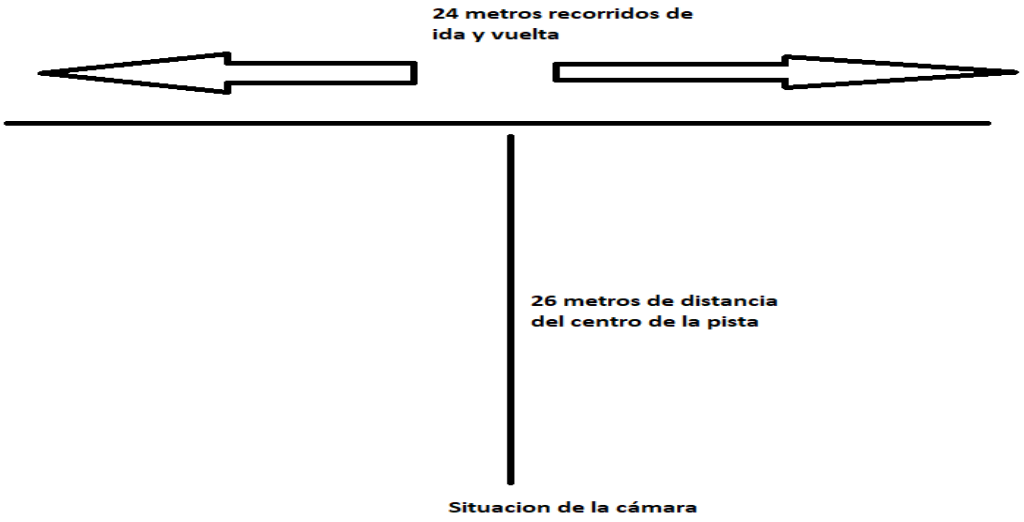


Figura 20: Plano de grabación.

Se realiza varias secuencias para su posterior estudio, algunas de ellas adjuntadas al proyecto. Y todas ellas en formato mp4, por su gran calidad de imagen que ayudará al seguimiento del corredor. Antes de comenzar con la modificación del código, se vuelve a probar el código base obtenido de la web con estos vídeos caseros y se obtiene:

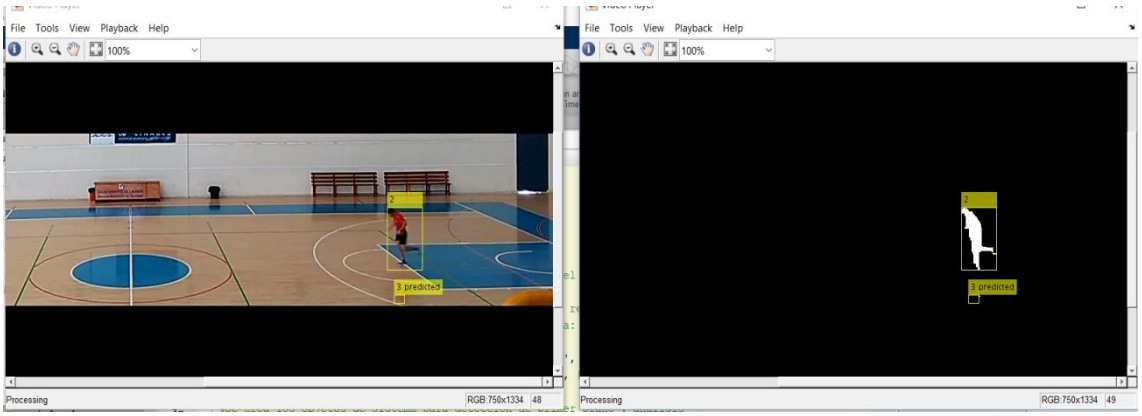


Figura 21: Prueba inicial.

Se puede observar como es capaz de localizar a la persona que está corriendo en ese instante, además de una predicción de un posible cambio de sentido de la marcha. Uno de los objetivos que se marcó desde el principio.

Para ello se va a utilizar un algoritmo de sustracción de fondo basado en modelos de mezcla gaussiana, además el análisis de blob capaz de detectar el conjunto de píxeles conectados, y por último la utilización del filtro Kalman para la detección del objeto en movimiento de cada pista (track). Gracias a este filtro se puede predecir la ubicación del track en cada frame y determinar la probabilidad de que cada detección se asigne a cada pista (todo esto se explicará más los siguientes puntos antes de profundizar de lleno en el código).

5.2 – FILTRO KALMAN:

5.2.1 – INTRODUCCIÓN:

Una de las claves del algoritmo desarrollado es el uso del Filtro Kalman, ya que se trata de un método recursivo que genera una estimación de la posición de frame a otro del vídeo en cuestión. Se va trata de una serie de ecuaciones matemáticas. Su principal objetivo es averiguar un estimador de un sistema en el instante t a partir de la información disponible en el instante t-1, y actualizar dicha estimación a partir de la información actual del instante t, por esta razón es un método recursivo. Otro aspecto positivo es que no se necesita una frecuencia de corte, ya que se basa en la característica del ruido permitiendo de esta manera filtrar en todo el espectro de frecuencias. Además sus ecuaciones solo dependen de una muestra anterior y la muestra presente lo que permite un ahorro considerable de memoria a la hora de ser implementado, a parte de su fácil programación.

Es el principal algoritmo para estimar sistemas dinámicos representados en estados. En la representación el sistema, que puede ser afectada por ruido blanco ya que no hay correlación estadística entre sus valores, está descrito por un conjunto de variables denominadas: variables de estado como son la velocidad, la posición… las cuales no se pueden medir, sino estimar añadiendo así una incertidumbre. El estado contiene toda la información relativa al sistema en un cierto instante de tiempo, la cual es capaz de permitir conocer el comportamiento pasado del sistema, con el objetivo de predecir su comportamiento futuro.

El filtro de Kalman va a consistir en dos pasos muy importantes y se debe cumplir ese orden para el correcto funcionamiento:

- Predicción a través del modelo dinámico: describe la transformación del vector de estado en el tiempo. Se usa la función de Matlab: “predict” capaz de predecir a través de la llamada al filtro de Kalman las posiciones de los centroides.

- Corrección a través del modelo de observación: El modelo de observación representa la relación entre el estado y las mediciones. Se usa la función de Matlab: “Correct” para actualizar la covarianza de error de estimación del filtro Kalman.

Ambos se repiten para cada intervalo de tiempo t.

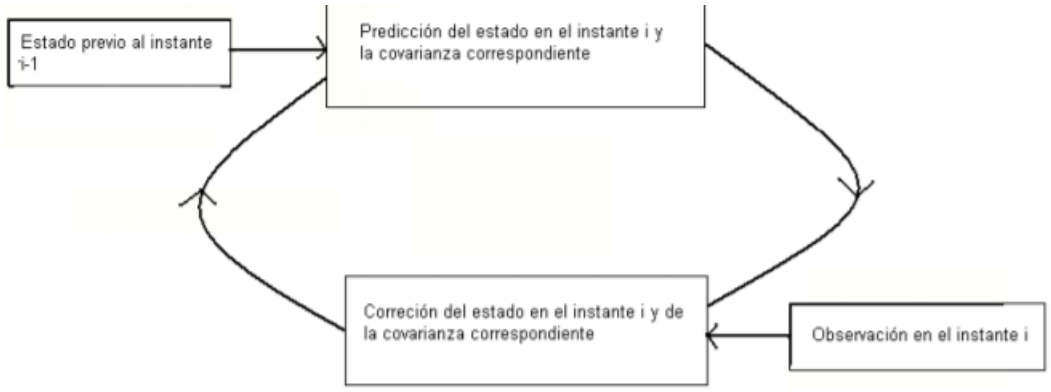


Figura 22: Circuito del Filtro Kalman.

5.2.2- COMPONENTES DEL FILTRO KALMAN:

Existen 3 componentes básicos del filtro de Kalan: el vector de estado, el modelo dinámico y el modelo de observación.

Un ejemplo de vector de estados:

$$x = \begin{bmatrix} espacio & en\ metros \\ velocidad & en\ m/s \end{bmatrix} \qquad (1)$$

Y en cada uno tendrá dos valores el valor previsto (previo a la actualización) y el valor corregido tras la misma.

Para el modelo dinámico se parte de:

$$x_t = Ax_{t-1} + w_{t-1} \qquad (2)$$

Para el modelo de observación se parte de:

$$z_t = Hx_t + v_t \qquad (3)$$

Por último, las variables v y w representan el ruido del proceso y de la medición. Se asumen que son independientes una de la otra, blancas y con distribución normal de probabilidad.

$$p(w) \sim N(0,Q) \qquad (4)$$
$$p(v) \sim N(0,R)$$

Donde Q y R son las matrices de la covarianza que pueden cambiar con cada medición, pero también se se puede constante.

5.2.3 – FUNCIONAMIENTO DEL FILTRO KALMAN:

Va a seguir este proceso de retroalimentación estimando el estado en un instante de tiempo determinado por medio de los datos observados:

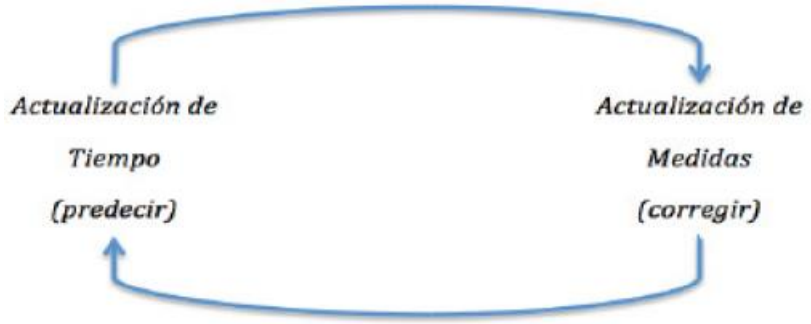


Figura 23: Proceso de retroalimentación del filtro Kalman.

La primera parte se encargada de proyectar la estimación del estado y de la covarianza, es decir grado de variación del error en el momento t tomando como referencia el estado en el momento t-1. La segunda parte son las ecuaciones responsables de la retroalimentación, es decir, incorporan nueva información dentro de la estimación anterior llegando a la estimación a posteriori.

Las ecuaciones de la etapa de predicción son idénticas a la vistas en (2) despreciando el término w que puede afectar a la hora de predecir ya que se interpretará constante y así poder calcular el estado t-1:

$$X_t = AX_{t-1} \qquad (5)$$

$$P_t = AP_{t-1}A^t + Q \qquad (6)$$

Las ecuaciones de la etapa de corrección del filtro:

$$K_t = P_t H^t (HP_t H^t + R)^{-1} \qquad (7)$$

$$X_t = X_t + K_t\,(Z_t - HX_t) \qquad (8)$$

Las ecuaciones (5) y (6):

Realizan una estimación del estado en (5), además de la covarianza del error en (6) desde t-1 a t, A es una matriz con dimensiones NxN que se va a encargar de relacionar el estado anterior t-1 con el estado actual, podría cambiar A con el paso del tiempo, pero normalmente se asume constante. Y Q se encarga de representar la covarianza de la perturbación aleatoria del proceso que trata de estimar el estado.

En las ecuaciones (7) y (8):

El primer paso es el cálculo de la ganancia del Kalman, es decir: K_t que se va a encargar de minimizar la covarianza del error de la nueva estimación. La matriz H es de dimensión MxN es la ecuación en diferencia de la medición, encargado de relacionar el estado con la medición y le pasa lo mismo que A, es decir puede variar con el tiempo, pero se asume normalmente constante. Posteriormente al medir el proceso de nuevo obtenemos Z_t que nos ayudará a tener una nueva estimación de X_t. Se vuelve a la ecuación (9) para obtener una aproximación mejorada de la covarianza del error P_t. Y el proceso se repite tomando como punto de partida las nuevas estimaciones.

$$P_t = (1 - K_t H) * P_t \qquad (9)$$

5.3: ANÁLISIS DE BLOB:

Otro aspecto de gran importancia es conocer el significado de blob (en inglés, Binary Large Object cuya traducción literal Objeto grande binario) usado en el algoritmo, son un tipo de dato que puede almacenar datos binarios. Estos son diferentes a la gran mayoría de tipos de datos que existen ya que se puede utilizar en bases de datos, como números enteros, números de coma flotante, caracteres y cadenas que almacenan letras y números. Dado que los blobs pueden almacenar datos binarios, pueden usarse para almacenar imágenes u otros archivos multimedia. En el caso de este proyecto, los blob serán los encargados de almacenar píxeles de los objetos en movimiento y encontrar la conexión con aquellos que aparecen en frames posteriores.

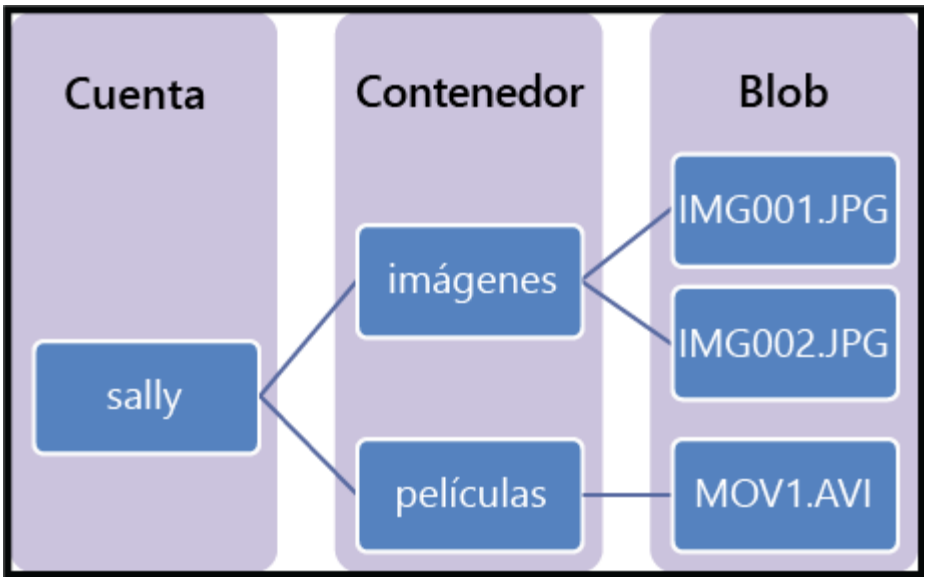


Figura 24: Ejemplo de blob.

Por lo general van a requerir un almacenamiento mayor que la mayoría de los datos con los que se trabajan de forma diaria. Pueden llegar a ocupar hasta gigabytes.

5.4: FOREGROUND Y BACKGROUND:

Dos términos muy importantes con los que se trabaja a la hora del seguimiento y puede llevar a confusión. A la hora de procesar cualquier imagen se entiende por detección de primer plano o foreground al conjunto de técnicas cuyo objetivo se basa en la detección de objetos en movimiento que aparecen en el vídeo sobre el que se trabaje, por el contrario el concepto de background es la parte estacionaria del vídeo, es decir aquello que va a permanecer de forma estática a lo largo de la secuencias del vídeo. Existen varios estimadores para el cálculo del background y foreground, como por ejemplo Running Gaussiana Average, Kernel Densisty Estimation (KDE), en el algoritmo desarrollado se ha empleado un modelo de múltiples Gaussianas creado por Stauffer & Grimson que se utilizará la herramienta de Matlab llamada: "ForegroundDetector" que va a comparar un marco de vídeo con un modelo de fondo para determinar si los píxeles individuales forman parte del fondo o del primer plano usando modelos de mezcla gaussiana.

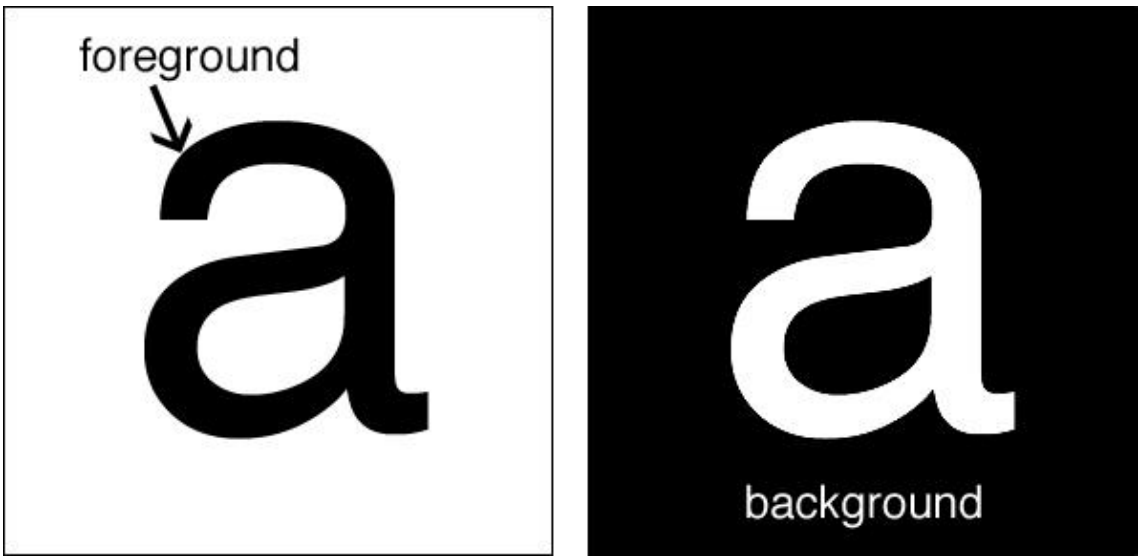


Figura 25: Foreground v.s. Background.

5.4.1- FUNCIONAMIENTO DEL MODELO DE MÚLTIPLES GAUSSIANAS DE STAUFFER Y GRIMSON:

Se trata de un estimador paramétrico [13] que es una evolución de running gaussian average, que modela el fondo de cada píxel mediante una función de densidad de probabilidad de la función Gaussiana, que quedará determinada mediante la media y la varianza. En cambio el estimador usado en este proyecto va a utilizar hasta K, normalmente se aconseja entre 3 y 5, Gaussianas por color para estimar la pdf (funcion de densidad de probabilidad) del modelo de cada píxel, será útil que este número sea grande para las variaciones periódicas en el fondo, por ejemplo la acción del viento que mueva un árbol o una bandera.

$$f(z) = \sum_{i=1}^{k} w_i * G(z, \mu_i, \epsilon_i) \quad (10)$$

Con la G se representa las gaussianas donde la z es el valor del pixel, μ y ϵ son los valores estadísticos de cada Gaussiana. Por último w es una ponderación que tiene este modelo.

Su utilización es simplemente para la obtención del modelo de fondo y así decidir si el pixel es información del fondo o del objeto del primer plano. Las Gaussianas pueden representar el fondo o el primer plano, por lo que se aplica un criterio para decidir de forma dinámica cuáles son de cada uno. Para ello, cada pixel (i,j) de la imagen, donde i representa la posición de fila y j la columna, se modela con el modelo de múltiples Gaussianas (10), se va a combinar entre sí a partir de los factores de ponderación, w_i que se modifican iterativamente en función de las veces que se da el valor que recogen.

El valor de ponderación se quiere normalizar:

$$w_i = \frac{w_i}{\sum_{i=1}^{k} w_j} \tag{11}$$

Intentando conseguir que $\sum_{i=1}^{k} w_i = 1$.

Para conocer si pertenece al primer plano o fondo, se va a utilizar el parámetro B:

$$B = \arg min_b \left(\sum_{i=1}^{b} w_i < T \right) \tag{12}$$

Donde t es normalmente 0.6 y B se trata del número mínimo de distribuciones que se incluyen en el sumatorio. Las B primeras distribuciones Gaussianas serán las que modelan la función de densidad de probabilidad del fondo, ya que se trata de algo más estático y aparece con más frecuencia, correspondiendose con las Gaussianas que se utilizan más veces.

A continuación se compara el valor del pixel de entrada para saber si puede entrar en alguna Gaussiana, de la siguiente forma:

$$|x_t - \mu_i| > k\sigma_i \tag{13}$$

El valor de x_t un canal de color del píxel, μ_i es la media de la Gaussiana y σ_i se trata de la desviación estándar de la Gaussiana. Si la inecuación se cumple para todas las Gaussianas se va a decidir por primer plano ya que no encaja con el modelo probabilístico del píxel aprendido hasta el momento y se genera una nueva Gaussiana para adaptar el modelo al fondo. En cambio, si la inecuación no se cumple para una Gaussiana entonces el valor encaja con el modelo, llegando a encajar con una de las B primeras distribuciones Gaussianas y entonces se asignará al fondo, debido a que la frecuencia con la que ha aparecido es lo suficientemente elevada como para ser considerado fondo.

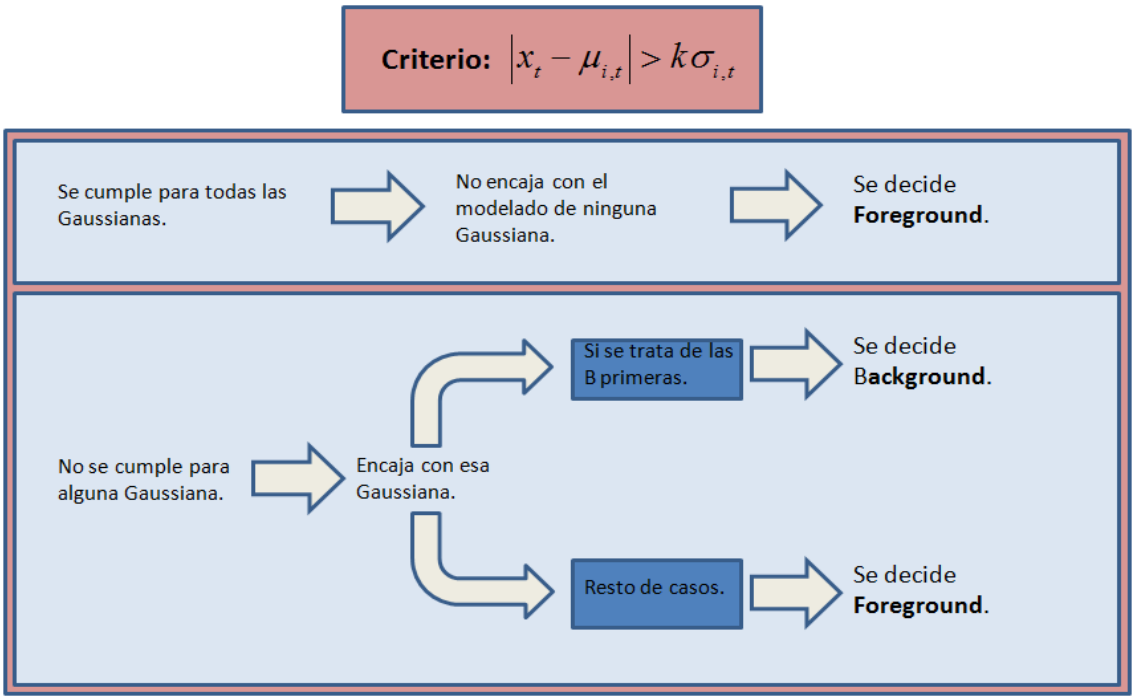


Figura 26: Resumen Modelo múltiples Gaussianas.

Se destaca debido a la calidad de los resultados (solo con observar la figura 26 se ve la poca cantidad de falsas detecciones):

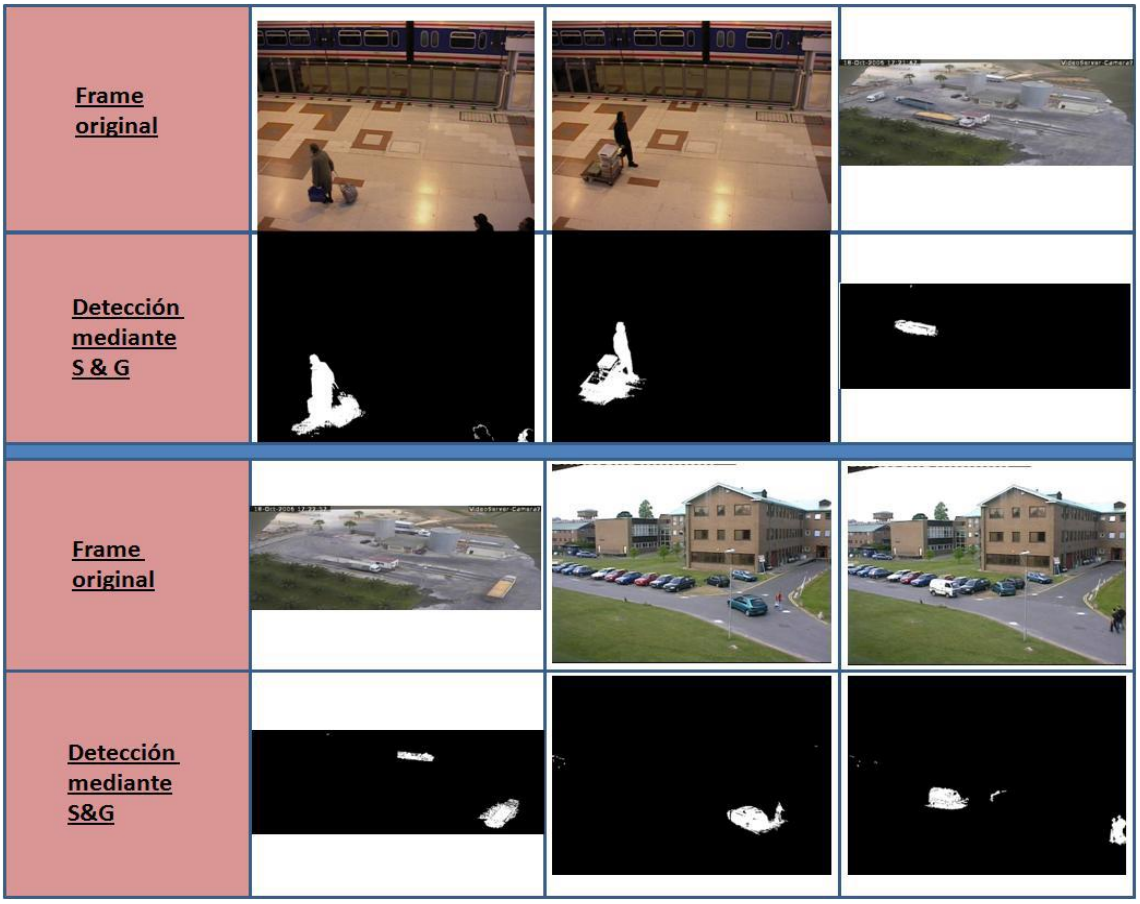


Figura 27: Ejemplos de resultados del método de Stauffer & Grimson.

5.5- PROGRAMA MATLAB DESARROLLADO:

La aplicación desarrollada, como bien se dijo antes es una modificación del código que se puede adquirir de la web de Matlab llamado: Motion-Based Multiple Object Tracking, en el que se va a trabajar a partir de un vídeo desde una cámara estática. La parte más importante y puede que costosa es el mantenimiento a lo largo de los frames del vídeo de la cada detección en su correspondiente track, ya que es donde se puede encontrar los datos a partir de los cuales voy a poder obtener las diferentes estadísticas del ejercicio realizado por el deportista.

Lo primero que se va a encontrar en el código es el cuerpo principal tracking () de la aplicación que para empezar va a llamar a las dos estructuras principales que serán explicadas explicadas en 6.4.1 y 6.4.2. Luego entra en juego el bucle principal encargado de detectar objetos en movimiento y realizar tracking en el que entrará de forma continua, es decir por cada frame que se lea, el funcionamiento irá una detrás de la otra.

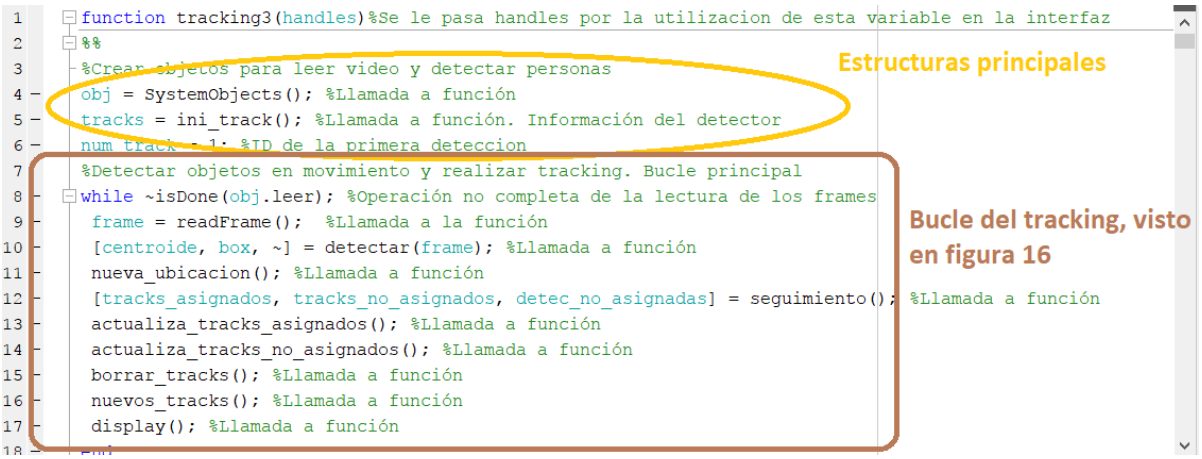


Figura 28: Cuerpo principal del algoritmo de tracking.

5.5.1: OBJ:

Se crea, como se ve en la figura 29, todos los objetos que el detector utilizará, como por ejemplo para tener acceso a las rutas de los vídeos guardados en subcarpetas (Figura 29.1), acceso a la información del vídeo donde podemos obtener una serie de variables necesarias para los cálculos (Figura 29.3) y la inicialización de otro conjunto de ellas para después utilizarlas en la interfaz como por ejemplo velocidad máxima, aceleración máxima, la masa del cuerpo (figura 29.2)… Además se crea los objetos de sistema para detección de primer plano (Figura 30.1) y análisis de blob (Figura 30.2).

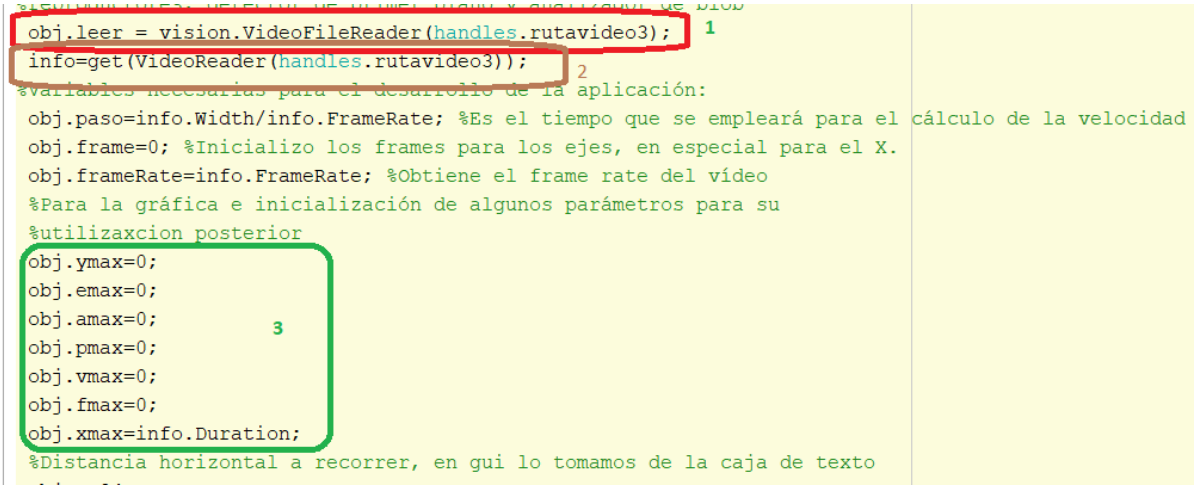


Figura 29: Primera parte del bloque de objetos.

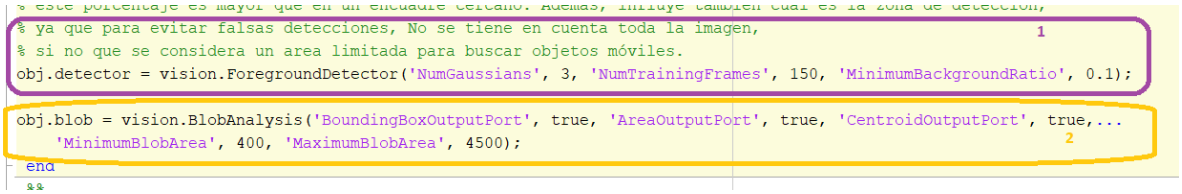


Figura 30: Segunda parte del bloque de objetos.

Cuyos valores usados se explican a continuación:

Para la sustracción del fondo o detección de primer plano se realiza el método explicado anteriormente de la mezcla de Gaussianas que se implementa para distinguir móviles respecto del fondo. Para esto hay que tener en cuenta que por defecto, vision.ForegroundDetector usa 5 gaussianas. Estas gaussianas corresponden a "características" que busca en los pixeles. Solo se busca pixeles que pertenezcan a 3 categorías:

- Un background inmóvil (no cambia ni su iluminación ni su color a lo largo de la toma).

- Un background móvil (por ejemplo el background que cambia de color por efecto del avance del día, o las sombras que se proyectan sobre él).

- Un foreground u objeto móvil de interés.

A pesar de que por defecto se usa 5, en el algoritmo se va a utilizar 3 gaussianas (se experimentó con más gaussianas, pero eso significaba subdividir las 3 categorías mencionadas y los resultados eran más ruidosos, es decir un mayor número de detecciones no deseadas o que no corresponden con lo que se busca).

El número de training frames fue puesto en 150 frames (lo que Matlab nos proporciona por defecto para la función "ForegroundDetector"), ya que el objeto detector necesita una cantidad de muestras del video en que no haya objetos móviles para definir las características del background inmóvil. En este caso, se obtiene en los videos un espacio de 150 frames libres en la escena y se emplea como training (el momento corresponde a los instantes iniciales del video).

El minimum background ratio, controla qué porcentaje mínimo de la escena se espera que sea background. En un encuadre grande donde los objetos móviles se ven pequeños, este porcentaje es mayor que en un encuadre cercano. Además, influye también cuál es la zona de detección, ya que para evitar falsas detecciones, No se tiene en cuenta toda la imagen, si no que se considera un área limitada para buscar objetos móviles.

Por otro lado, para el análisis de blob los 3 OutputPort están puestos en "true" para obtener como salida: las coordenadas de la caja del blob detectado, el centroide de dicha caja y el área del blob encontrado.

Además, con el centroide y las coordenadas de la caja podemos diferenciar detecciones de móviles que estén muy cercanos entre sí (como por ejemplo 2 personas que se mueven a la par sobre una misma calle de atletismo).

El MinimumBlobArea y MaximumBlobArea está definido en 400 y 4500 pixeles respetivamente debido a que en las pruebas realizadas este valor da la mejor tasa de verdaderas detecciones positivas permitiendo cubrir la región de interés y los diversos tamaños del objeto u objetos en movimiento.

También, y así evitar continuas llamadas por el bucle de principal, se decide por colocar aquí los ejes con los títulos de las gráficas para que solo se tenga que realizar una vez este dibujo y ahorrar tiempo de ejecución (Figura 31).

```
title(handles.vt3,'Velocidad en función del tiempo');Título de la gráfica
        xlabel(handles.vt3,'Tiempo(seg)'); Eje x
        ylabel(handles.vt3,'Velocidad(m/s)'); Eje y
        grid(handles.vt3,'on'); Para la malla de fondo
        hold(handles.vt3,'on'); Dibujar
title(handles.vp3,'Velocidad en función de la posicion');
        xlabel(handles.vp3,'Posicion(m)');
        ylabel(handles.vp3,'Velocidad(m/s)');
        grid (handles.vp3,'on')
        hold (handles.vp3,'on')
```

Figura 31: Llamada a ejes de gráficas.

5.5.2- TRACKS:

Se crea una estructura array (tracks) vacía con propiedades del objeto al que se le hará seguimiento en el video para su posterior uso en lo que corresponde a registros estadísticos. Contiene todas las personas que han sido detectadas en el video; si una persona desaparece por mucho tiempo, se elimina en funciones posteriores.

```
tracks = struct(...
        'id', {}, ...  1
        'boxx', {}, ...  2
        'kalmanFilter', {}, ...  3
        'posicion', {}, ...  4
        'posicionReal', {}, ...  5
        'posicionAnt', {},...  5
        'distancia',{},...  6
        'totalVisibleCount', {}, ...  7
        'consecutiveInvisibleCount', {},...  7
        'velocidad', {},...
        'aceleracion', {},...
        'fuerza', {},...
        'potencia', {},...
        'energia', {},...
        'veloAnt',{},...
        'vmax',{},...                          8
        'fmax',{},...
        'acelAnt',{},...
        'amax',{},...
        'pmax',{},...
        'eAnt',{},...
        'emax',{},...
        'fAnt',{},...
        'pAnt',{},...
  9     'iniX',{},...
  10    'centroid', {},...
        'centroidAnt', {});
```

Figura 32: Estructura de tracks.

1- id: id asociado al track dentro de esta estructura.

2- box: la caja que contiene al objeto en movimiento, podrá verse al reproducirse el vídeo.

3- Kalmanfilter: que es el objeto de Kalman para el algoritmo, lo explicare posteriormente.

4- posición: es el "contador", es decir el número en la etiqueta que aparece en la caja en la reproducción.

5- posicionReal y posicionAnt: la ubicación del objeto en movimiento dentro del vídeo, actual y la anterior a la misma.

6- distancia: Utilizada para el eje X cuando representa la velocidad en función de la distancia total recorrida.

7- totalVisibleCount y consecutiveInvisibleCount como su propio nombre indica es el contador según las detecciones verdaderas o falsas se producen durante la duración del vídeo.

8- Ahora viene el bloque de las variables a calcular como la velocidad, aceleración, etc… También la de los instantes anteriores para su uso en diferentes partes del código como la obtención de la gráfica y por último las máximas que aparecerán en la interfaz.

9- iniX: Es el inicio del atleta en pixeles, es decir trasladado la realidad a pixeles la posición donde empieza a correr.

10- centroide y centroidAnt: Es la posición central del objeto en movimiento detectado en pixeles (filas y columna).

5.5.3- LECTURA DE FRAMES:

La función claramente "más simple" de este algoritmo, se va a encargar de leer frame a frame del vídeo con la función que realiza step.

```
function frame = readFrame()
    %Lectura de frames
    obj.frame=obj.frame+1;
    frame = obj.leer.step();
```

Figura 33: Lectura de frames paso a paso.

5.5.4- FUNCIÓN DETECCIÓN:

Se va a aplicar operaciones morfológicas para remover las distorsiones y rellenar huecos, además se encarga de procesar y separar background de foreground, es decir separar el primer plano del fondo.

```
%Discriminación de objetos de primer plano
 fondo = obj.detector.step(frame);
%strel: Elemento estructural

 fondo = imerode(fondo, strel('rectangle', [3,3]));

 fondo = imclose(fondo, strel('rectangle', [15, 15]))

 fondo = imfill(fondo, 'holes');

 [~, centroide, box] = obj.blob.step(fondo);
end
```

1

2

Figura 34: Operaciones morfológicas y aparición del análisis de blob.

Las operaciones morfológicas (Figura 34.1) van a simplificar imágenes y conservar las principales características del contorno del objeto en movimiento. Las que se van a emplear en este algoritmo: (en ambos las medidas de la caja, en este caso el rectángulo de base y altura se dan entre corchetes y son diferentes para obtener una mejor precisión en las mejoras aportadas por las operaciones morfológicas):

- Imerode o Erosión: Elimina píxeles. Reduce bordes. Separa objetos próximos.

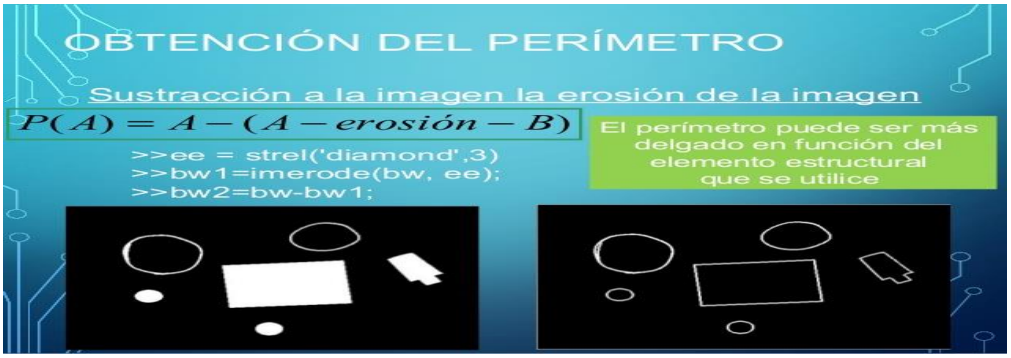


Figura 35: Imerode.

- Imclose o Cierre: Es la realización de una dilatación seguida de una erosión, utilizando el mismo elemento estructural en ambas operaciones. Rellena detalles, conectando objetos que están próximos entre sí. Suaviza contornos de los objetos. Rellena vacíos en el contorno. Elimina pequeños huecos. Aumentar la definición de formas.

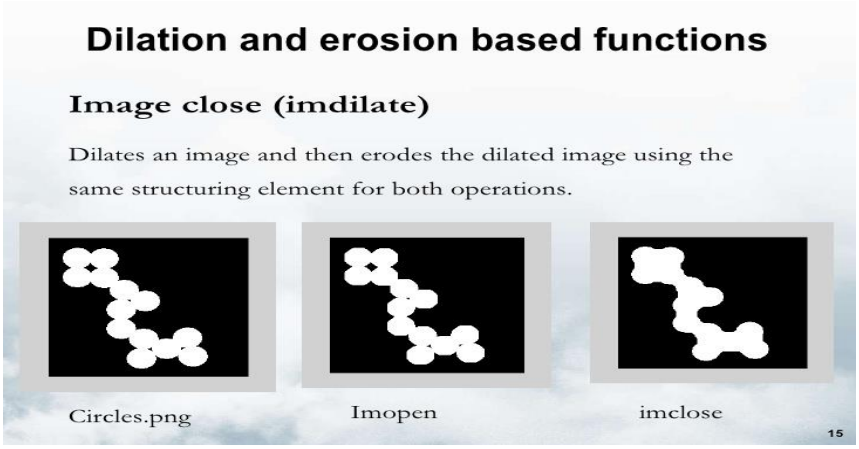


Imagen 36: Imclose

- Imfill: Se utiliza para evitar huecos en negro dentro de los principales y así no puedan ser interpretados como otros objetos. Rellena estos huecos de los objetos principales.

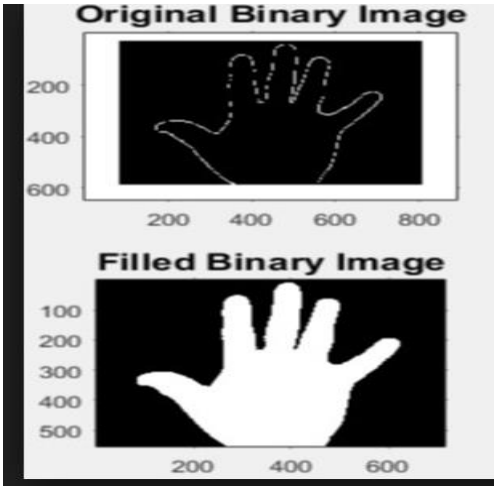


Figura 37: Imfill

Por último, realiza el análisis de blob (Figura 34.2) para encontrar los componentes conectados. Devolviendo los centroides y las cajas delimitadores de las detecciones.

5.5.5- FUNCIÓN NUEVA UBICACIÓN:

Hace una estimación de la posición donde estará el siguiente rectángulo de detección. Esto es fundamental para el seguimiento, ya que se comparará el rectángulo de detección siguiente con el que se obtuvo en la predicción. Por lo que es la primera vez que aparecerá el filtro del Kalman, realizando el primer paso del proceso de retroalimentación que se habló en el punto 5.2.1. En la figura 38.1.

```
    for x = 1:length(tracks)
    boxx = tracks(x).boxx;
%Predice la ubicación actual del track.                                    1
    nuevo_centroide = predict(tracks(x).kalmanFilter); %Predice coordenadas del centroide del blob.

    nuevo_centroide = int32(nuevo_centroide) - boxx(3:4) / 2;
    tracks(x).boxx = [nuevo_centroide, boxx(3:4)];
    end                                                                    2
end
```

Figura 38: Localización de la nueva ubicación.

También se va a encargar del movimiento la caja de detección para que este en el centro de la ubicación predicha. La parte del código boxx(3:4) apunta a los elementos 3 y 4 del vector boxx. Este vector tiene como elementos [x inicial, y inicial, altura, ancho] que corresponden a la caja. Entonces se produce una diferencia al centroide del blob el valor boxx(3:4) / 2 (que es la mitad del alto y ancho de la boxx), significa encontrar la posición del (x,y) superior izquierdo de la caja. En la figura 38.2.

Con esto se reubica la misma para que siga al móvil correspondiente y se actualiza la info de posición con la línea: tracks(i).boxx = [nuevo_centroide, boxx(3:4)].

5.5.6- FUNCIÓN DE SEGUIMIENTO:

Esta función se basa en principios estadísticos para definir si una detección merece ser asignada a un track, o si en cambio tiene grandes posibilidades de tratarse de un falso positivo. En este caso se le asigna una penalización de "20" para todas aquellas detecciones que no queden asignadas a un track. El valor es empírico y por defecto de la aplicación encontrada en la web de Matlab.

```
    num_detec = size(centroide, 1);
    costo = zeros(num_tracks, num_detec);
%Calcula el costo de asignar cada detección a cada track
    for i = 1:num_tracks
    costo(i, :) = distance(tracks(i).kalmanFilter, centroide);
    end
%Resuelve el problema de asignacion

    costo_no_asignados = 20;
    [tracks_asignados, tracks_no_asignados, detec_no_asignadas] = assignDetectionsToTracks(costo, costo_no_asignados);
end
%%
```

Figura 39: Función seguimiento.

Se va a encargar de calcular el costo de asignar cada detección a cada track en términos de su cambio de posición por medio de la distancia euclidiana ("distance") entre lo que se predice y la detección (Figura 39.1). Se almacena en una matriz MxN, donde M es el número de pistas, y N es el número de detecciones.

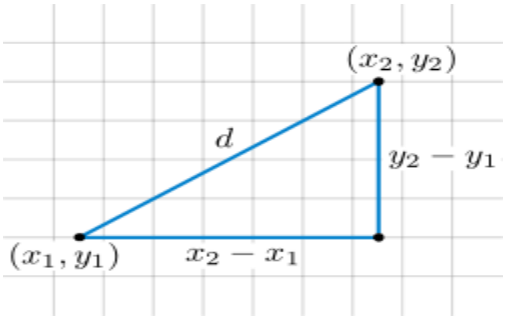


Figura 40: Distancia euclidiana, d: Su definición coincide con el concepto más común de distancia, es decir:

$$d = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \quad (14)$$

También resuelve el problema de asignación si se establece demasiado bajo, aumenta la probabilidad de crear una pista nueva y puede producir una fragmentación de la pista. El ajuste demasiado alto puede dar como resultado una sola pista que corresponde a una serie de objetos móviles separados (Figura 39.2).

Como resultado nos va a dar una matriz de Mx2 que va a contener los índices correspondientes a las pistas asignadas y detecciones. También devuelve los índices de pistas y detecciones que quedaron sin asignar. Con estos índices obtenidos, se realizan varios procesos en los siguientes 4 puntos.

5.5.7- FUNCIÓN ACTUALIZA TRACKS ASIGNADOS:

Es la función más importante del algoritmo, ya que su misión es actualizar las variables de seguimiento definidas en tracks.

En el primer bucle está recorriendo el array de asignación que es el que contiene todas las detecciones presentes en el frame de video que se está analizando. Por esto recorre desde la primera detección, hasta la última (que se corresponde con el contador de "numAssignedTracks") (Figura 41.1)

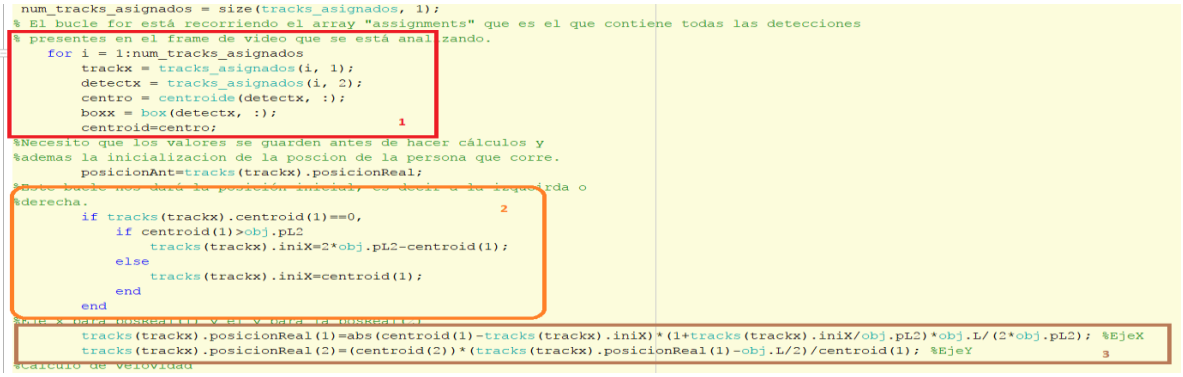


Figura 41: Primera parte de la actualización.

En la Figura 41.2 se puede observar el if que te indicará si el corredor empieza en la izquierda o derecha de la pantalla y Figura 41.3 te indica la correspondencia de la posición real con respecto a la posición en pixel. (Ver Anexo 1 para poder entender esta correspondencia y como se ha calculado).

```
%ademas de la aceleración:
        energia=abs(0.5*obj.masa*(velocidad^2-tracks(trackx).veloAnt^2))/1000;
        potencia=energia*obj.paso;
        fuerza=potencia/velocidad;
        aceleracion=fuerza/obj.masa;
```

Figura 42: Fórmulas.

En este punto se van a realizar los cálculos físicos de las diferentes informaciones de interés para un atleta como son velocidad, aceleración, fuerza, potencia y energía cinética, a partir de las siguientes fórmulas:

$$v = \frac{posición\ acual - posición\ inicial}{tiempo} \qquad (15)$$

Donde posición actual es para mí posiciónReal y la posición inicial es posiciónAnt y el tiempo es un factor de paso que está definido en la función 6.4.1:

$$paso = \frac{pixeles\ horinzontales}{frameRate\ de\ la\ grabación, en\ este\ caso\ a\ 30fps} \qquad (16)$$

Las demás fórmulas van encadenadas a partir de esta (15):

$$Energía\ cinética = \frac{1}{2} * masa * \Delta velocidad\ (15)^2 \qquad (17)$$

Es decir una variación de la velocidad que viene dada de la diferencia e la velocidad actual en el frame actual menos la velocidadAnt guardada. Continuando con la potencia:

$$Potencia = \frac{Energía\ cinética\ (17)}{tiempo} \qquad (18)$$

Donde con el tiempo vuelve a aparecer el factor paso visto en el (16). Por último:

$$Fuerza = \frac{Potencia\ (18)}{Velocidad\ (15)} \qquad (19)$$

$$Aceleración = \frac{Fuerza\ (19)}{masa\ del\ corredor} \qquad (20)$$

A continuación se va a producir la segunda llamada al filtro Kalman realizando el segundo paso, y último, del proceso de retroalimentación que se habló en 6.1.

```
        aceleracion=fuerza/obj.masa;

 %Corrige de forma estimada la ubicación de persona usando la detección nueva.
    correct(tracks(trackx).kalmanFilter, centro);
 %Reemplaza el rectángulo de predicción con el rectángulo predicho.
    tracks(trackx).boxx = boxx;
 %Actualiza la posicion del track.
```

Figura 43: Uso de la función de Matlab correct.

Seguidamente se procede a las gráficas y la obtención de cada valor máximo que serán visualizadas en la interfaz.

```
    tracks(trackx).potencia=potencia;
    tracks(trackx).energia=energia;
 %Graficar velocidad
    if ~(10*tracks(trackx).veloAnt<tracks(trackx).velocidad) || velocidad<obj.L,
        tracks(trackx).veloAnt=tracks(trackx).velocidad;
        if velocidad>obj.ymax,
            axis(handles.vt3,[-0.1 obj.xmax -0.1 1.1*velocidad]); %Ejes
            obj.ymax=velocidad;
            tracks(trackx).vmax=velocidad;
            texto=[ num2str(velocidad) 'm/s' ];
            set(handles.vmax3,'String', texto);
        end
        plot(handles.vt3,obj.frame/obj.framerate,velocidad,'.r');%Respecto al tiempo.
%Calculo de la distancia recorrida
        tracks(trackx).distancia=tracks(trackx).distancia+...
        abs(tracks(trackx).posicionReal(1)-tracks(trackx).posicionAnt(1));

        if velocidad>=obj.ymax,
            axis(handles.vp3,[-0.5 obj.L*1.1 -0.5 1.1*velocidad]); %Ejes
            obj.ymax=velocidad;
        end
        plot(handles.vp3,tracks(trackx).posicionReal(1),velocidad,'.b');%Respecto al espacio.
        plot(handles.fp3,tracks(trackx).distancia,velocidad,'.g');%respecto a la distancia recorrida
```

Figura 44: Gráfica velocidad y distancia recorrida en cada instante.

```
%Gráfica de aceleración
        tracks(trackx).acelAnt=tracks(trackx).aceleracion;
        if aceleracion>obj.amax,
            axis(handles.ap3,[-0.5 obj.L*1.1 0 1.1*aceleracion]); %Ejes
            obj.amax=aceleracion;
            tracks(trackx).amax=aceleracion;
            texto=[ num2str(aceleracion) 'm/s^2' ];
            set(handles.amax3,'String', texto);
        end
        plot(handles.ap3,tracks(trackx).posicionReal(1),aceleracion,'.c');


%Gráfica de energía
        tracks(trackx).eAnt=tracks(trackx).energia;
        if energia>obj.emax,
            axis(handles.ep3,[-0.5 obj.L*1.1 -0.5 1.1*energia]); %Ejes
            obj.emax=energia;
            tracks(trackx).emax=energia;
            texto=[ num2str(energia) 'kJ' ];
            set(handles.emax3,'String', texto);
        end
        plot(handles.ep3,tracks(trackx).posicionReal(1),energia,'.k');
```

Figura 44: Gráfica aceleración y energía.

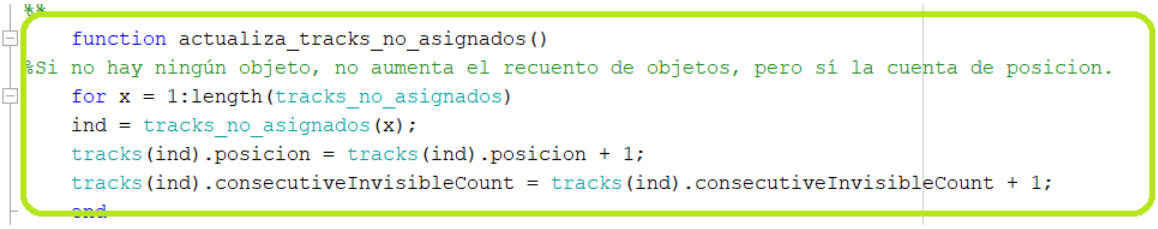
```
%Gráfica de fuerza:
        tracks(trackx).fAnt=tracks(trackx).fuerza;
        if fuerza>obj.fmax,
            obj.fmax=fuerza;
            tracks(trackx).fmax=fuerza;
            texto=[ num2str(fuerza) 'kN' ];
            set(handles.fmax3,'String', texto);
        end

%Gráfica de potencia:
        tracks(trackx).pAnt=tracks(trackx).potencia;
        if potencia>obj.pmax,
            axis(handles.pp3,[-0.5 obj.L*1.1 -0.5 1.1*potencia]); %Ejes
            obj.pmax=potencia;
            tracks(trackx).pmax=potencia;
            texto=[ num2str(potencia) 'kW' ];
            set(handles.pmax3,'String', texto);
        end
        plot(handles.pp3,tracks(trackx).posicionReal(1),potencia,'.y');
```

Figura 45: Gráfica potencia y máximo de velocidad, no hay gráfica ya que me parecía más interesante la variación de la velocidad con respecto a la distancia.

5.5.8- FUNCIÓN ACTUALIZA TRACKS NO ASIGNADOS:

Su función es tan simple como si una persona de un frame pasado no aparece en el frame actual, se marca 'invisible' por lo que aumentará la cuenta de consecutiveInvisibleCount definida anteriormente.

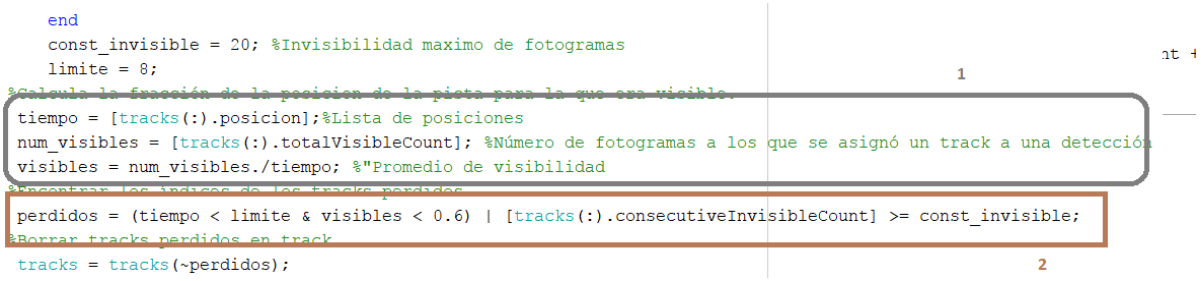


```
**
    function actualiza_tracks_no_asignados()
%Si no hay ningún objeto, no aumenta el recuento de objetos, pero sí la cuenta de posicion.
    for x = 1:length(tracks_no_asignados)
    ind = tracks_no_asignados(x);
    tracks(ind).posicion = tracks(ind).posicion + 1;
    tracks(ind).consecutiveInvisibleCount = tracks(ind).consecutiveInvisibleCount + 1;
    end
```

Figura 46: Función de aumento en la cuenta de tracks no visibles.

5.5.9- FUNCIÓN DE BORRADO DE TRACKS:

Cuando el objeto lleva muchos frames 'invisible' como indica la const_invisible, ya no está en el sistema de detección y se borrará su información, para ello se establece unas constantes para saber si un track debe eliminarse (Figura 47.2).



```
    end
    const_invisible = 20; %Invisibilidad maximo de fotogramas
    limite = 8;
%Calcula la fracción de la posicion de la pista para la que era visible.
tiempo = [tracks(:).posicion];%Lista de posiciones
num_visibles = [tracks(:).totalVisibleCount]; %Número de fotogramas a los que se asignó un track a una detección
visibles = num_visibles./tiempo; %"Promedio de visibilidad
%Encontrar los índices de los tracks perdidos
perdidos = (tiempo < limite & visibles < 0.6) | [tracks(:).consecutiveInvisibleCount] >= const_invisible;
%Borrar tracks perdidos en track
tracks = tracks(~perdidos);
```

Figura 47: Borrado de tracks.

Sólo considere las pistas que han sido visibles durante más de un número mínimo de fotogramas para el seguimiento, ya que también pueden aparecer las detecciones ruidosas pueden crear de pistas falsas (Figura 47.1).

5.5.10- FUNCIÓN NUEVOS TRACKS:

Esta función se resume en que si se detecta una persona nueva en el frame, se añade a la estructura del track, ya que aparece la misma estructura que en 6.4.4 para estas nuevas detecciones que se van añadiendo a continuación, con lo que irá aumentando de tamaño esta estructura de tracks.

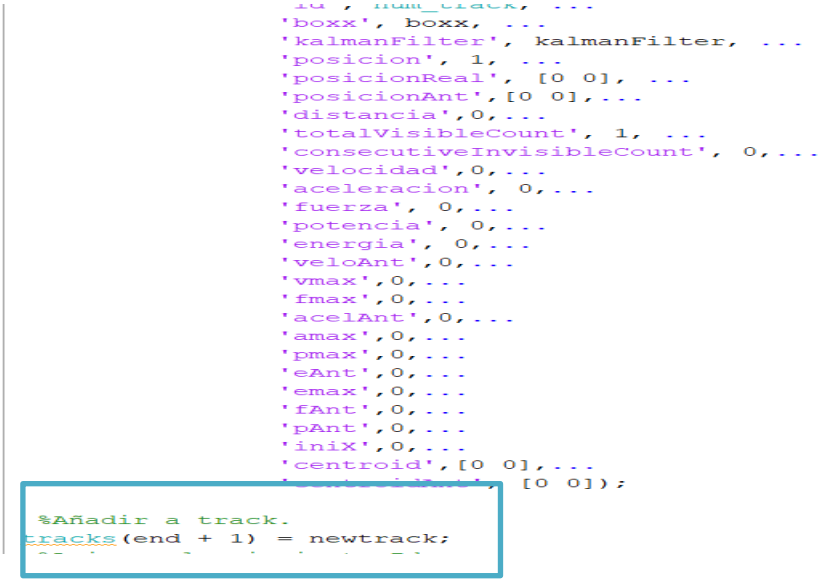


Figura 48: Aumento de la estructura tracks.

Se va a crear el objeto del filtro Kalman en el que los datos que aparecen indicarán:

```
    for i = 1:size(centroide, 1);
    centro=centroide(i,:);
    boxx=box(i, :);
% Crear objeto de Filtro de Kalman:
kalmanFilter = configureKalmanFilter('ConstantVelocity', centro, [200, 50], [100, 25], 1000);
```

Figura 49: Filtro Kalman creado.

Con este comando se utiliza para el seguimiento de un objeto físico en un sistema de coordenadas cartesianas. El objeto seguido puede moverse con velocidad constante o aceleración constante. Cuando el atleta es detectado por primera vez, el algoritmo crea un filtro de Kalman. Cuando se determina la localización del mismo, si se detecta, el filtro de Kalman predice primero su estado en el marco de video actual, por eso aparece el bucle for de los centroides.

- ConstantVelocity: Para los casos en los que se está estimando la posición de un objeto que se desplaza con la misma velocidad o aparentemente la misma se puede asumir un modelo en que la velocidad sea constante (modelo de velocidad constante), además es ventajoso en un modelo estacionario, es más prolongado para una mejor encaje con situaciones de la vida real. También puede ayudar a un mejor cálculo de la etapa de predicción.

- Centro: Indica la posición inicial donde va a trabajar.

- El [200,50] define el sitio en el frame en que se crea el objeto inicial para el filtro.

- El [100 25] son la mitad del primero, y corresponde a la incertidumbre del punto inicial del filtro, esto permite que la estimación del Kalman tenga más libertad de movimiento y pueda encontrar los móviles más rápido, con la problema de ser un poco más inestable.

- El "100" del final, corresponde al parámetro MeasurementNoise y simplemente indica qué tan lejos puede estar el punto inicial de la primera estimación debido al ruido.

5.5.11- FUNCIÓN DISPLAY:

Lo primero que hace es coger los frames y transformarlos a clase uint8, es decir un entero representado en 8 bits. Esto nos da 2^8=256 valores que se distribuyen en el rango de [0 255] para cada pixel. (Figura 50.1)

```
%Convertir frame y mascara a uint8 RGB.
frame = im2uint8(frame); 1
min_visible = 8;

 if ~isempty(tracks)%Determina si el array está vacío
 % Solo muestra tracks que han sido visibles por encima de un número
 % mínimo de frames, en este caso 8.
 num_tracks_reales = [tracks(:).totalVisibleCount] > min_visible;
 tracks_reales = tracks(num_tracks_reales);  2
 %Muestra los objetos. Si un objeto no ha sido detectado en este
 %frame, muestra su rectángulo predicho.
 if ~isempty(tracks_reales)
 %Obtiene rectángulos de detección e ids.
 box = cat(1, tracks_reales.boxx);
 %Va a representar un entero
 numero = int32([tracks_reales(:).id]);
  %Crea etiquetas para objetos indicando los que
  %si se usa del rectangulo predicho o el detectado.
etiqueta = cellstr(int2str(numero'));
prediccion = [tracks_reales(:).consecutiveInvisibleCount] > 0;
                prediccion1 = cell(size(etiqueta));
                prediccion1(prediccion) = {'prediccion'};
                etiqueta = strcat(etiqueta, prediccion1);        3
%Dibujar objetos en el frame.
 frame = insertObjectAnnotation(frame, 'rectangle', box, etiqueta);
 end
 end
%Para la reproducción sobre la interfaz gráfica.
%'Parent': Para construir una interfaz de usuario que
%le da control de la figura y las propiedades de los ejes.
    imshow(frame,'Parent',handles.video3);                    4
```

Figura 50: Algoritmo para la reproducción.

Se va encargar de mostrar los tracks por encima de un umbral en cuanto a la cuenta de frames se refiere (Figura 50.2). Posteriormente se va a crear la asignación de etiquetas para las detecciones y los movimientos donde se predice que va a ir el deportista (Figura 50.3).

Por ultimo va a devolver el vídeo, con los objetos detectados insertando tanto la etiqueta como la caja. (Figura 50.3) El resultado final puede verse en la figura 20.

5.6- INTERFAZ GRÁFICA DE USUARIO DE MATLAB:

Posteriormente, se ha creado una interfaz gráfica de usuario donde se incluye opciones interesantes como controles tales como menús, barras de herramientas, botones y controles deslizantes, donde se va a poder representar las dos partes fundamentales del TFG: detección y obtención de estadísticas interesantes para el deportista en forma de gráficas y valores máximos. Como punto de partida en este punto:

Al llamar en Matlab al comando Guide y aparecer la plantilla de una interfaz donde he introducido varios elementos necesarios para lo mencionado en el párrafo anterior:

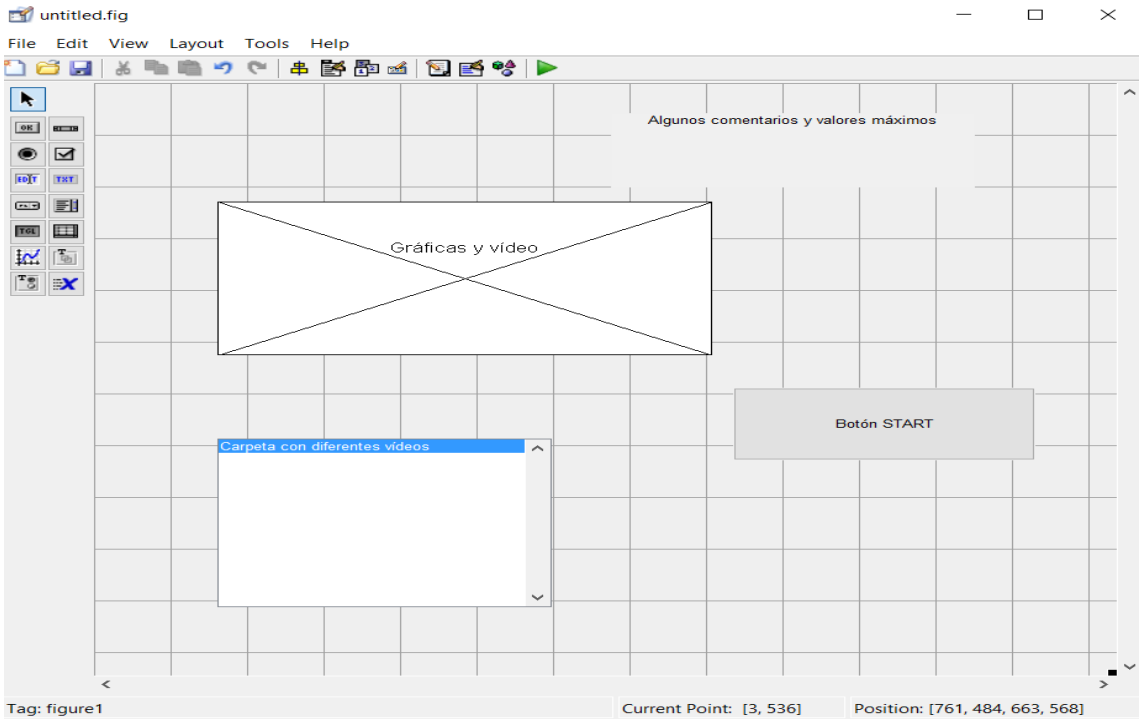


Figura 51: Plantilla y elementos de la guide empleados.

Los elementos empleados y algunos conceptos necesarios en mi algoritmo son los siguientes:

- Handles: Cada objeto (o figure, o axes, o button, etc.) está representado por una variable handles. Es la variable raíz de la interfaz gráfica y para unir interfaz y código se realiza por la fórmula siguiente:
  HANDLES.TAG

  Por ejemplo:

```
%'Parent': Para construir una interfaz de usuario que
%le da control de la figura y las propiedades de los eje
    imshow(frame,'Parent',handles.video3);
    drawnow();
end
end
```

Figura 52: Asociar el vídeo al axis video3 a través de handles.

- Callback: Una vez que los elementos están en posición y Al hacer click derecho se editan las funciones de llamada (View Callbacks) de cada uno de ellos, abre el archivo .m. asociado a nuestro diseño y nos posiciona en la parte del programa que corresponde a la subrutina donde se puede escribir el código de MATLAB que se ejecutará cuando el control sea utilizado. En este caso, como ejemplo todo el control de la aplicación se encuentra en el Callback de push button llamado START.

- Tag: Cada elemento creado en la interfaz tiene un identificador único (nombre del Tag) pulsando sobre el elemento dos veces sobre el izquierdo aparece una ventana donde identificaremos el TAG y pondremos el nombre que después usaremos en el código. Por ejemplo:

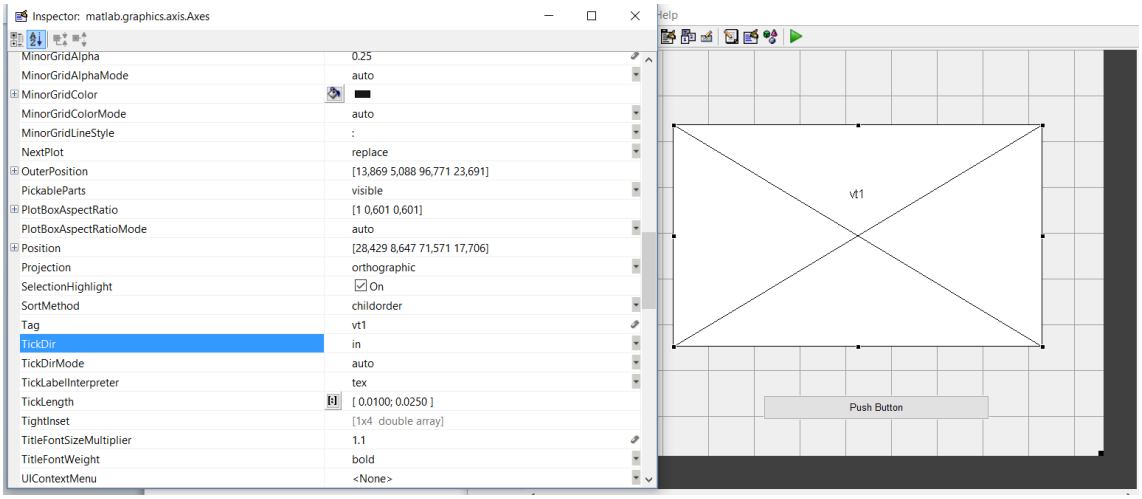


Figura 53: Tag del axis vt1.

Con el cual nos refieres al vt1 del código en mi caso es la gráfica velocidad tiempo del corredor 1:

```
plot(handles.vt1,obj.frame/obj.frameRate,velocidad,'.r');%Respecto al tiempo.
title(handles.vt1,'Velocidad en función del tiempo');
```

Figura 54: Referencia del código.

- La asignación y la obtención de valores de los componentes se realiza mediante las sentencias set y get, la primera tiene la capacidad de asignar valares y la segunda nos proporciona el valor que queremos enseñar.

- Texto estático o static text (comentarios y valores máximos) puede mostrar símbolos, mensajes o valores numéricos de una GUI, y puede colocarse en lugar deseado. No tiene cadenas de invocación (Cuando se interactúa con un control). Para el caso de mi código:

```
tracks(trackx).vmax=velocidad;
texto=[ num2str(velocidad) 'm/s' ];
set(handles.vmax1,'String', texto);
end
```

Figura 54: Establecer en vmax1 la velocidad máxima del corredor 1.

- Push button (botón de START) generan una acción cuando das clik con el puntero del ratón sobre ellos, este botón va a realizar toda la función tracking al completo vista a lo largo del 6.4.

- Cajas de lista o listbox (Carpeta con diferentes vídeos): Este elemento muestra una lista de componentes y permite a usuarios seleccionar unos o más artículos. En mi caso toda la colección de vídeos grabados. En el código:

```
54
55      % Choose default command line output for interfaztrackinguno
56 -    handles.output = hObject;
57 -    handles.d3=dir('corredor1/');
58 -    set(handles.videoscorredor3,'String',{handles.d3.name});
59      % Update handles structure
60 -    guidata(hObject, handles);
```

Figura 55: Establece el directorio, la carpeta de vídeos y que tenga acceso a los nombres de los vídeos.

```
78    function pushbutton1_Callback(hObject, eventdata, handles)
79    % hObject    handle to pushbutton1 (see GCBO)
80    % eventdata  reserved - to be defined in a future version of MATLAB
81    % handles    structure with handles and user data (see GUIDATA)
82 -  posvideo=get(handles.videoscorredor3,'Value');
83 -  handles.rutavideo3=[ 'corredor1/' handles.d3(posvideo).name ];
84 -  tracking3(handles);
85
86    % --- Executes on selection change in videoscorredor3.
```

Figura 56: Relaciona el vídeo que seleccionemos con la reproducción del mismo en la función base de tracking a través de la rutavideo.

- Axes o multiejes (Graficador y reproductor del vídeo): Es muy flexible y ofrece múltiples opciones a los programadores. Este comando axes abre un eje en un punto especificado dentro de una ventana de figura.

7- RESULTADOS:

Después de la explicación de todo el desarrollo y los pasos seguidos, llega la hora de mostrar los resultados conseguidos, para ellos he desarrollado dos ideas diferentes. Para ello en la interfaz gráfica he intentado dar la misma importancia tanto al vídeo parara su detección como a la extracción de datos:

7.1- PARA UNA PERSONA:

Lo primero que se verá al ejecutar el archivo .m de la interfaz gráfica, llamado: interfaztrackinguno.m será esto:

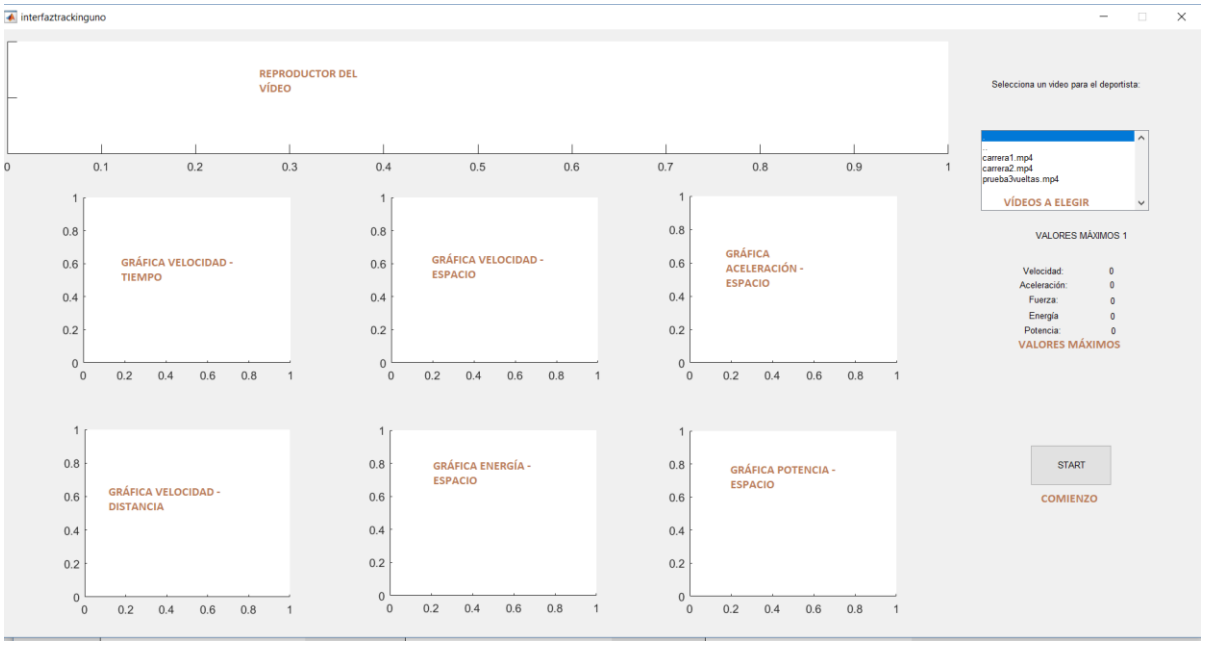


Figura 56: Modelo una persona

Para ponerlo en marcha seleccionamos un vídeo de la lista de vídeos ofrecida, para esta explicación con solo tres es suficiente, en el primero y el segundo solo será una carrera de una vuelta empezando desde un lado de la pista y el otro, en cambio el tercero son 3 vueltas para mostrar la diferencia entre gráficas sobre todo la gráfica de velocidad-distancia. Quizás algún resultado no sea el adecuado, pero el equipo de grabación no era el mejor y mínimos fallos a la hora de procesar la imagen, puede dar algún resultado irregular.

Es interesante conocer la diferencia entre las gráfica de velocidad sobre todo la de espacio y tiempo, ya que en la primera únicamente (y como todas las que tengan como referencia en el eje X el espacio) se observará los diferentes picos de máximos y mínimos y entorno a que valores medios el atleta se va a mover, en cambio la segunda se podrá ver como el corredor evoluciona en cada vuelta que ve, esto se podrá observar sobre todo en la prueba tres que se realizará, ya que las dos primeras solo son a una vuelta como se mencionó en el párrafo anterior.

Durante el proceso de reproducción, esto es lo que se va a ir viendo:

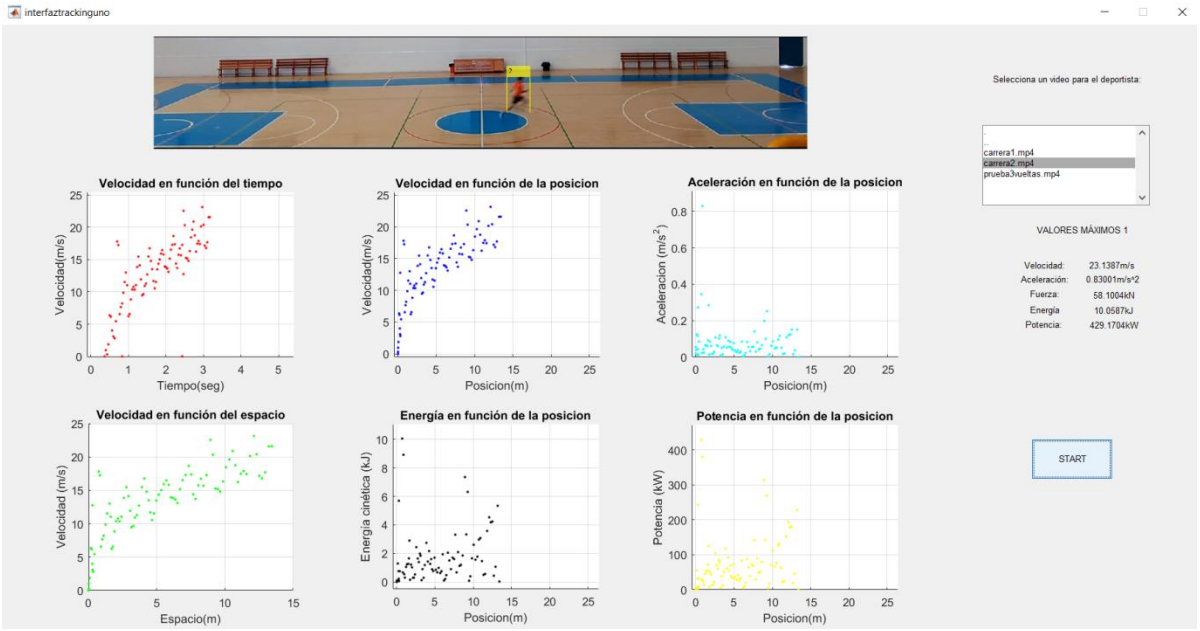


Figura 57: Reproducción de derecha a izquierda.

Lo más interesante es como las 4 gráficas de la derecha que dependen de la posición del corredor se mueven en sentido inverso de la siguiente reproducción en la Figura 60, la gráfica velocidad – espacio es útil para la reproducción de más de una vuelta y siempre se mueven en esa dirección ya que el tiempo y el espacio recorrido siempre serán positivos, el segundo porque en el código:

```
%Calculo de la distancia recorrida
        tracks(trackx).distancia=tracks(trackx).distancia+...
        abs(tracks(trackx).posicionReal(1)-tracks(trackx).posicionAnt(1));
```

Figura 58: Distancia recorrida, trabajando en valor absoluto.

Otro aspecto señalable es como es capaz de empezar en (0,0) y acabar en el (tiempo máximo o espacio máximo, 0). Como resultado de esta primera prueba se obtiene:

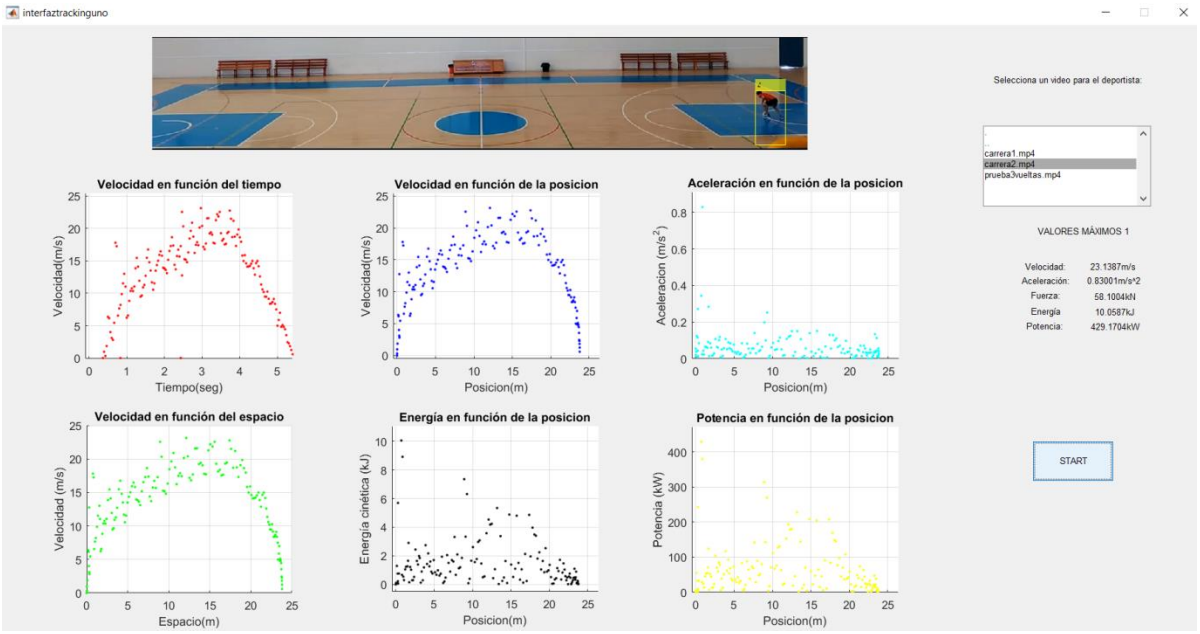


Figura 59: Resultado primera prueba.

En la segunda prueba, muy similar a la primera, comenzaría tal que así:

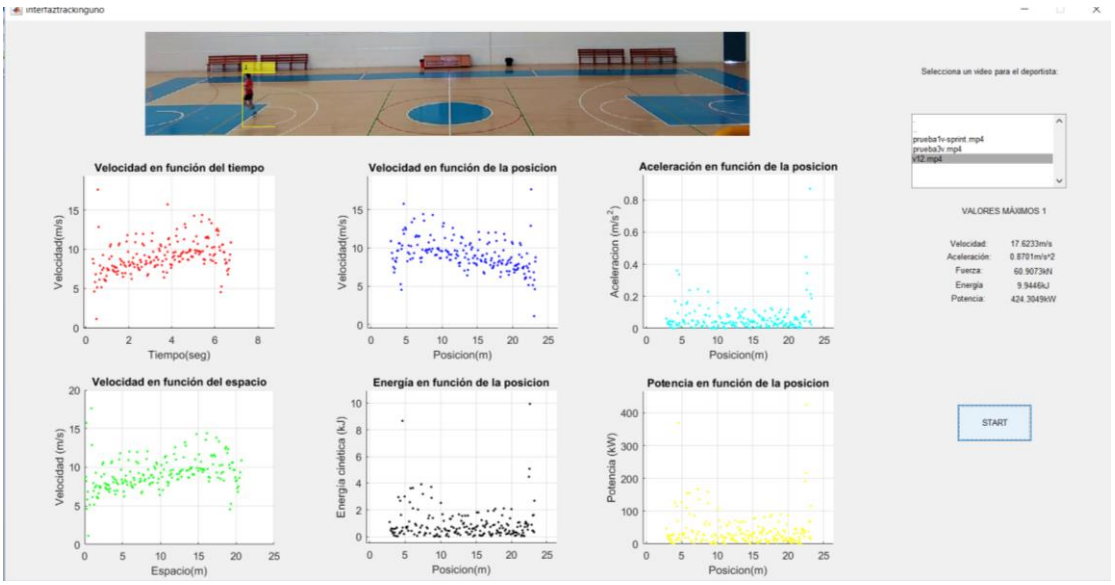


Figura 60: Segunda prueba de sprint máximo.

Como se aprecia en la gráfica velocidad-espacio, el corredor se mueve en sentido contrario, los resultados varían, ya que el desplazamiento ahora es de derecha a izquierda, empezando y acabando en las mismas coordenadas que antes.

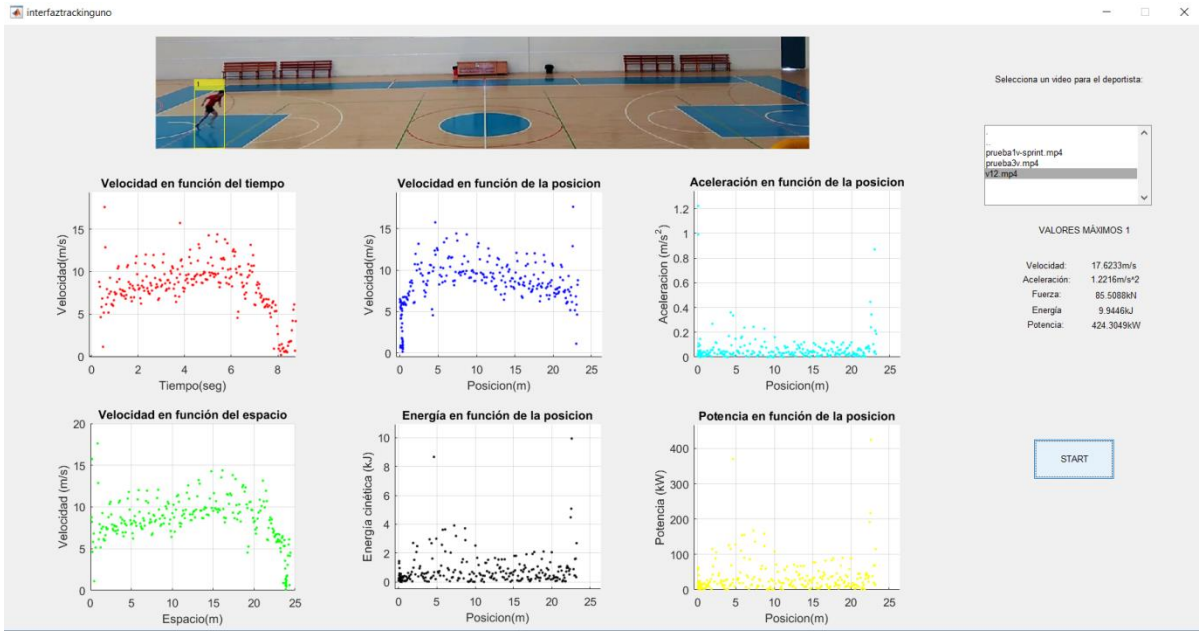


Figura 61: Resultado segunda prueba.

También hay variantes en el tema de la velocidad ya que esta segunda ha sido un sprint al máximo donde la velocidad ha variado mostrando picos mayores, los demás valores se han mantenido medianamente constantes, con pequeñas diferencias.

La última prueba en solitario constará de 3 vueltas, después de iniciarlo, y durante su transcurso:

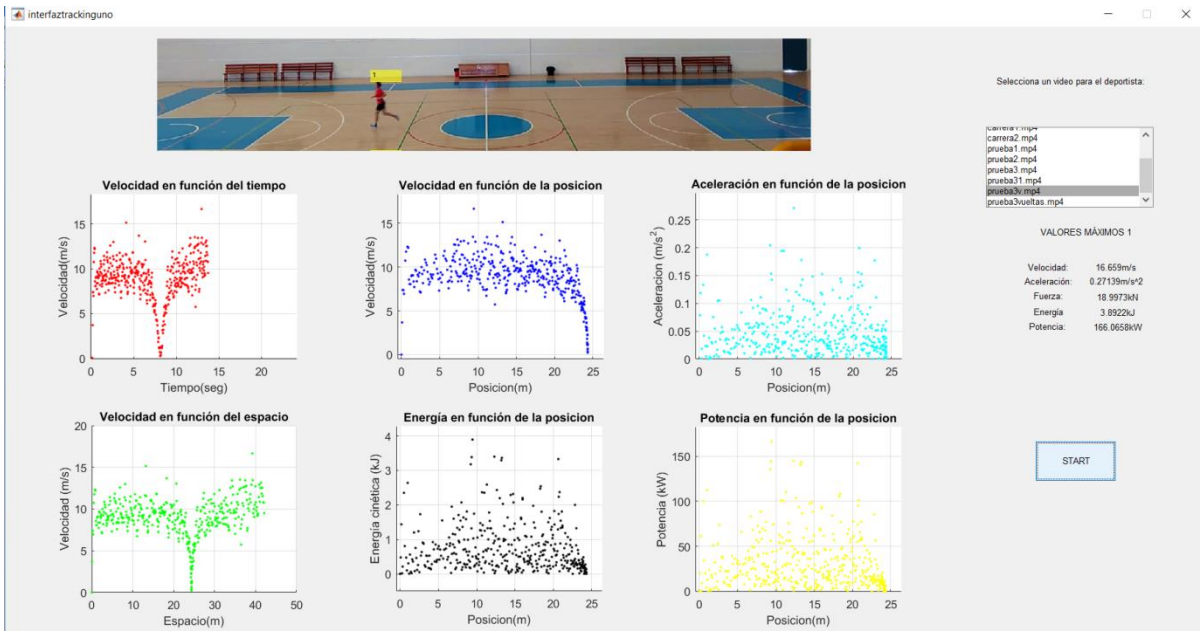


Figura 62: Tercera prueba

Como dato interesante se puede observar en las gráficas de velocidad-tiempo y velocidad-espacio que se va formando esa especie de curva entre vuelta y vuelta ya que se produce una frenada cuando llega al final de cada una de ellas. Finalmente llegamos al siguiente resultado:

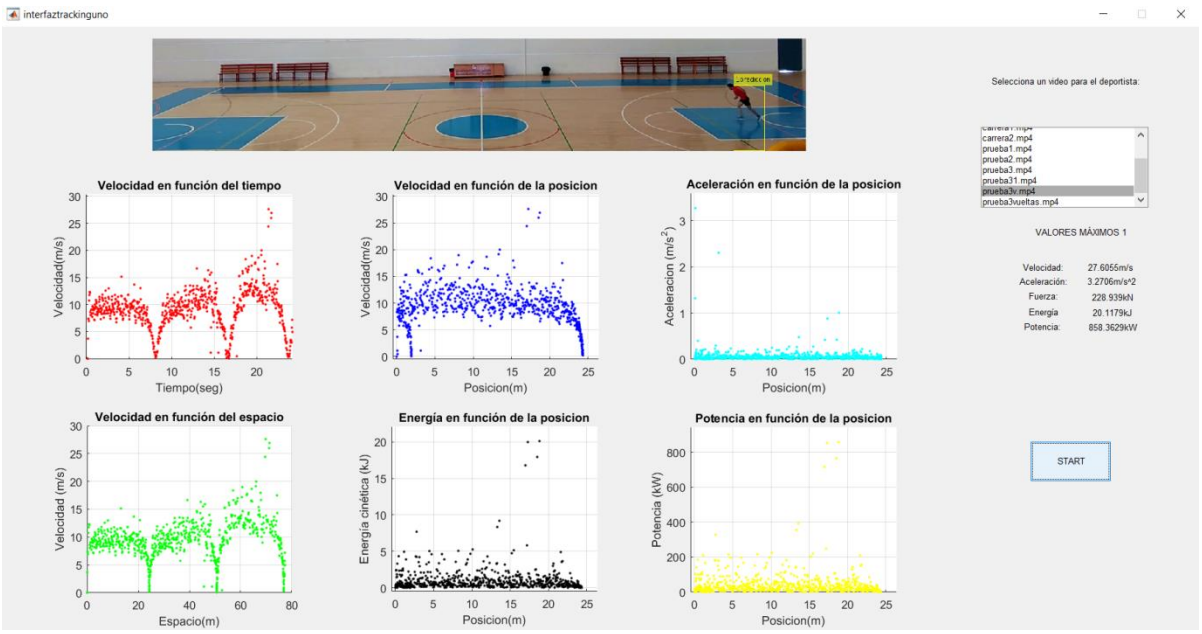


Figura 63: Final tercera prueba

Se puede apreciar en la gráfica velocidad-espacio también como ha variado la velocidad, donde la tercera vuelta ha sido en la que el deportista ha sido más rápido.

7.2- PARA DOS PERSONAS:

Para añadir algo más, he desarrollado una interfaz de comparación entre dos deportistas, el código varía solamente en la ruta de vídeos que en este caso serán dos y dos funciones de tracking idénticas que cada uno tendrá un nombre diferente, tal que así:

```
handles.output = hObject;
handles.d3=dir('corredor1/');
set(handles.videoscorredor3,'String',{handles.d3.name});
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes interfaztrackinguno wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = interfaztrackinguno_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
posvideo=get(handles.videoscorredor3,'Value');
handles.rutavideo3=[ 'corredor1/' handles.d3(posvideo).name ];
tracking3(handles);
```

```
% --- Executes just before trackinginterfaz is made visible.
function trackinginterfaz_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to trackinginterfaz (see VARARGIN)

% Choose default command line output for trackinginterfaz
handles.output = hObject;
handles.d1=dir('corredor1/');
set(handles.videoscorredor1,'String',{handles.d1.name});
handles.d2=dir('corredor2/');
set(handles.videoscorredor2,'String',{handles.d2.name});
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
posvideo1=get(handles.videoscorredor1,'Value');
handles.rutavideo1=[ 'corredor1/' handles.d1(posvideo1).name ];
% tracking(handles);
posvideo2=get(handles.videoscorredor2,'Value');
handles.rutavideo2=[ 'corredor2/' handles.d2(posvideo2).name ];
trackingdoble(handles);
```

Figura 64: Cambios de una carpeta de vídeos a dos.

La interfaz desde donde se va a trabajar sería en este sería tal que así:

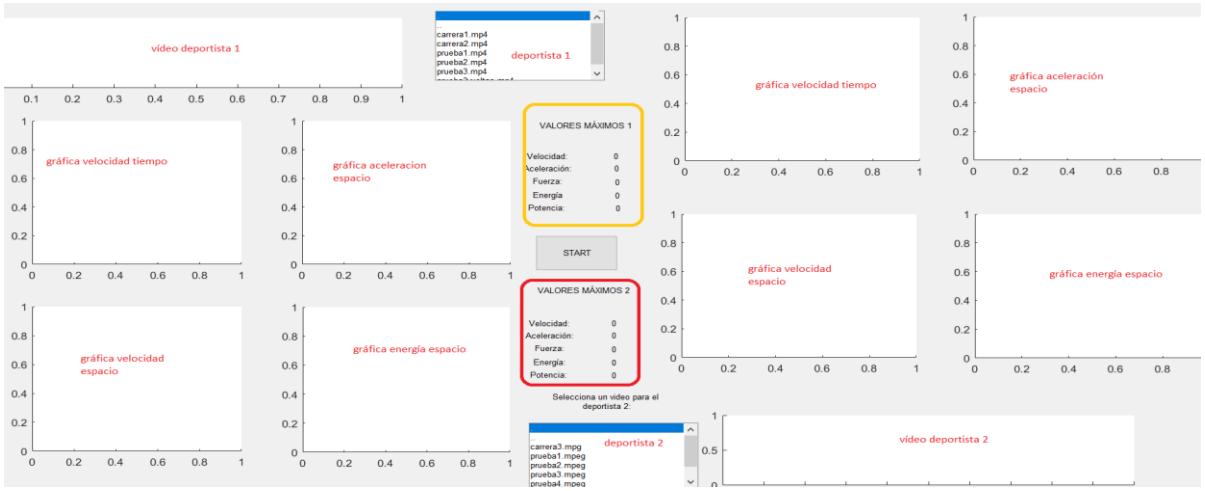


Figura 65: Interfaz doble

Muy similar a la anterior interfaz gráfica, pero se ha reducido el número de gráficas por falta de espacio, ya que se ha ocupado todo lo que podía ofrecer la interfaz para su buena visualización. Se selecciona un vídeo de cada uno, pulsamos posteriormente START y como resultado:

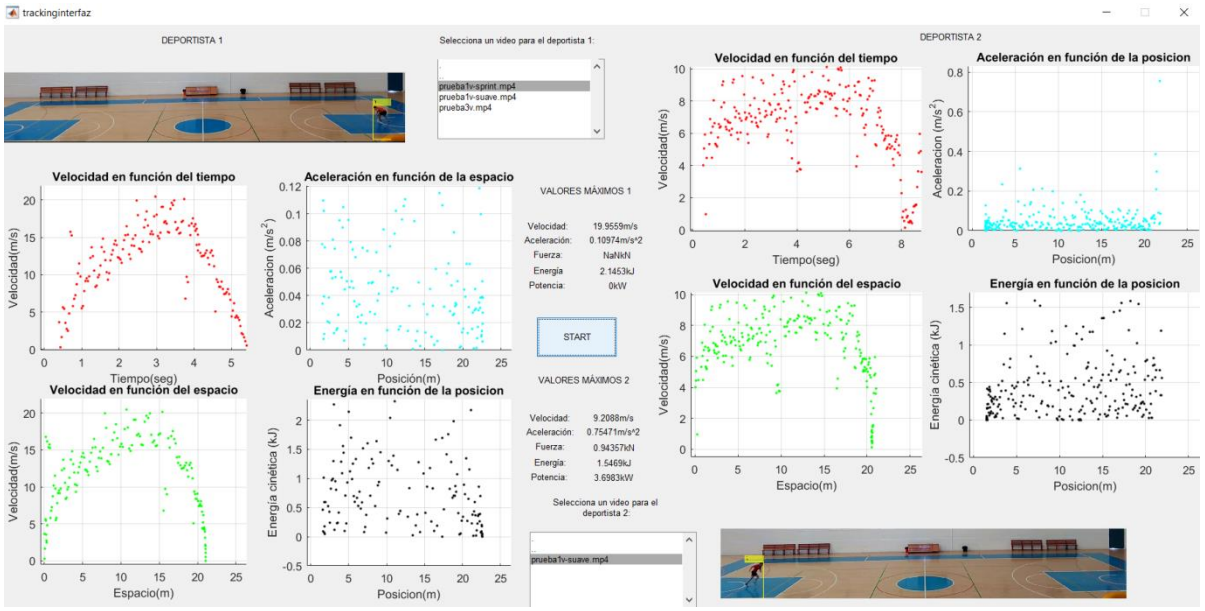


Figura 66: Resultados del tracking doble.

Comparando los resultados se puede ver como el deportista dos, estando en sprint, ha podido superar al deportista uno en velocidad también destacar como el ritmo de subida de velocidad de este último es más constante que el segundo, generando una mayor energía y potencia.

8- TRABAJO FUTURO:

Como este trabajo se ha realizado en interior, animo a realizar una aplicación en un entorno más dinámico donde la propia naturaleza o el mismo público pueden entrar en escena y poder afectar a la hora de la detección, creo que la base está hecha con este trabajo para esto mencionado:

Figura 67: Carrera amateur al aire libre.

Tras el algoritmo desarrollado se puede completar con diversas mejoras pensadas a lo largo de su realización, sobre todo en el tema del lenguaje que una buena opción por ejemplo sería en lenguaje C++ o también a través ordenadores más potentes o con más prestaciones con el que se ha realizado, se puede ahorrar tiempo de procesamiento, e incluso añadir nuevos seguimientos, como por ejemplo en un vídeo con varios corredores tipo finales olímpicas de 100m, 200m, etc… Es decir de las diferentes pruebas de atletismo que existen. Como por ejemplo se ve en la imagen 40, debería realizarse la detección de cada uno de ellos y realizar el seguimiento tipo el que se ha realizado además de los cálculos estadísticos.

Figura 68: Final mundial 100m 2017.

Otras de las ideas que pueden aparecer en un futuro es la adaptación de una

aplicación a la realidad virtual, ya que está en pleno auge de crecimiento y con una mayor demanda estos últimos años. Como aprendizaje a la hora de realizar entrenamientos o visualizar tus propios entrenamientos para conseguir mejoras de rendimiento puede ser sus puntos favorables.

Figura 69: Posibilidad de gafas de entrenamiento en realidad virtual

Por último, se podría extender a otros deportes, creo que un ejemplo muy bueno y práctico sería la natación, con vídeos de cámaras estáticas conocidas como spidercam o flycam donde se puede observar perfectamente toda la piscina desde la altura y los competidores y los cálculos serían muchos más exactos, por ejemplo:

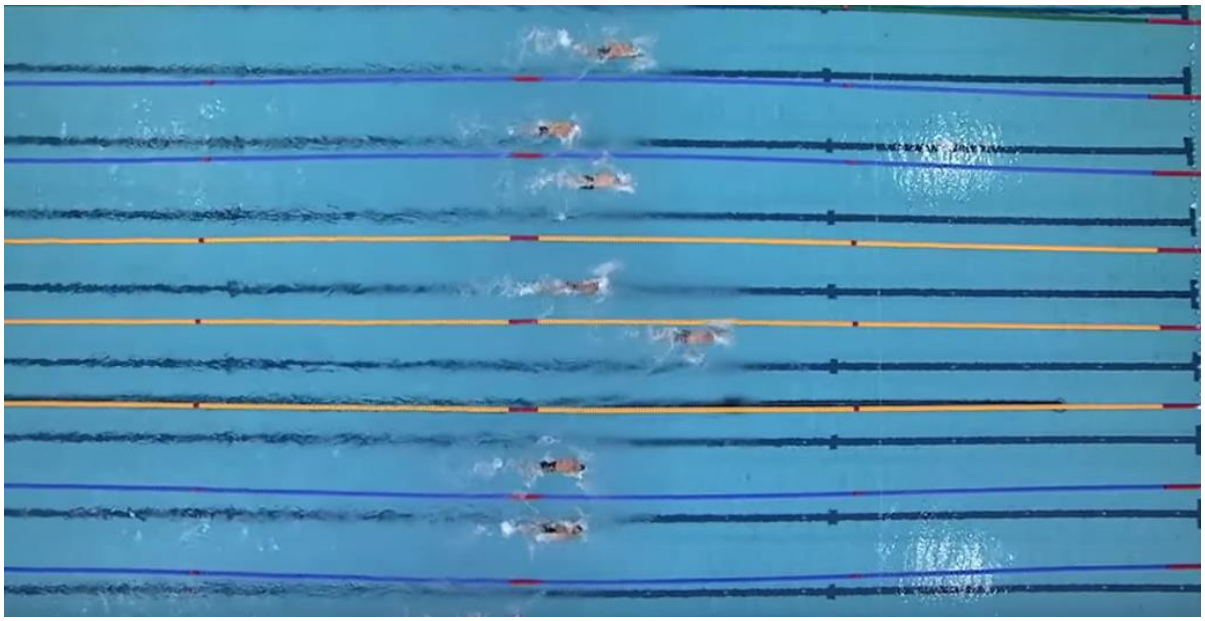


Figura 70: Final natación Budapest.

9- CONCLUSIÓN:

Como ya se ha mencionado varias veces a lo largo del proyecto la misión de este trabajo han sido dos: La capacidad de localizar y realizar el tracking a algún objeto y a partir de los frames obtener un conjunto de medidas que sean adecuadas para el cálculo de las estadísticas del ejercicio realizado por el atleta.

Para lograr estos dos objetivos se ha desarrollado una aplicación en lenguaje Matlab apoyándose en un código que ya estaba realizado que se encuentra dividido en diferentes subrutinas al cuál se le han aplicado las diferentes modificaciones pertinentes para llegar a obtener los datos ya mencionados como son: velocidad, aceleración, etc… que están combinadas con cálculos de sus máximos y gráficas. A parte de la grabación propia de diferentes vídeos que pueden verse en la carpeta adjunta al código, ya que los vídeos proporcionados por Internet no se han encontrado los adecuados que se ajustasen al proyecto.

Durante el proceso de este trabajo en este período de tiempo, he aprendido numerosos conocimientos totalmente nuevos para mí, como las técnicas de tracking o manejar de forma básica, pero en mi opinión lo necesario, de la interfaz de Matlab de usuario. Además de conocimientos que ya se poseían de procesado de vídeo y programación en Matlab por el estudio realizado a lo largo de este grado de telecomunicaciones en su rama de sonido e imagen.

Personalmente, estoy bastante satisfecho con el trabajo realizado. Aunque no ha sido nada fácil, debido a la dificultad de realizar del seguimiento en un vídeo es un paso más sobre el procesamiento de imagen que he estudiado durante el grado; además la utilización de código y librerías para su adaptación, que supone un estudio más profundo de la misma en el entorno Matlab; y la utilización de imágenes reales, con la complejidad de su captura y procesamiento hace que finalmente y en mi opinión haya podido alcanzar el objetivo marcado. Además, un trabajo realizado de forma personal aporta siempre una mayor destreza con la mirada puesta en el futuro inmediato.

BIBLIOGRAFÍA:

[1] Portillo Portillo, José. *Detección de movimiento de objetos mediante secuencias de video*. Sección de estudios de postgrado e investigación. Escuela Superior de Ingeniería mecánica y eléctrica Unidad Zacatenco. México D.F. Mayo de 2012.

[2] Ayala-Sánchez, A., Zavaleta-Carrillo, P. y Pérez-Cruz, D. Seguimiento de objetos rígidos con movimiento libre en secuencias de imágenes. *Unacar Tecnociencia.* 2009. 24 - 34. 1316-6239.

[3] Herrera Hermosilla, Juan Carlos. *Breve historia del espionaje*. 1ª Edición. Madrid: Imprenta Fareso, 2012. ISBN-13: 978-84-9967-314-1.

[4] Legua Cruz, Carlos. *Seguimiento automático de objetos en sistemas con múltiples cámaras*. Proyecto fin de carrera. Universidad autónoma de Madrid. Madrid: Mayo 2013.

[5] Yilmaz, Alper. Javed, Omar. Mubarak, Shah. *Object Tracking: A Survey.* Image Processing and Computer Vision. Universidad Estatal de Ohio y Universidad Central de Florida. Estados Unidos. Diciembre 2006.

[6] Trocado Ferreira, Filipe. *Video Analysis in Indoor Soccer with a Quadcopter*. Proyecto fin de carrera. Facultad de ingeniería de la Universidad de Oporto. Oporto. Julio 2014.

[7] Blanco, Ilia. *El BIG DATA en el deporte de élite.* Disponible en: http://www.legaltoday.com/blogs/nuevas-tecnologias/blog-ecija-2-0/el-big-data-en-el-deporte-de-elite

[8] Rodríguez Moya, Álvaro. *Estudio del filtro de partículas aplicado al seguimiento de objetos en secuencia de imágenes.* Proyecto fin de carrera. Universidad Carlos III. Madrid. Octubre 2009.

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

[10] Pereira Ruiz, Sergio. *Localizacion de robots mediante filtro de Kalman.* Proyecto fin de carrera. Escuela Técnica Superior de Ingenieros Universidad de Sevilla. Junio 2010.

[11] Porras Martín, Raúl. *Seguimiento de personas en vídeo basado en detección.* Proyecto fin de carrera. Universidad autónoma de Madrid. Junio 2014.

[12] *Usando columnas de tipo BLOB.* Disponible en: https://firebird21.wordpress.com/2014/03/21/usando-columnas-de-tipo-blob/ Marzo 2014.

[13] Stauffer and Grimson. *Adaptive background mixture models for real-time tracking. Volumen 2.* Agosto 1999. Los Alamitos, CA, USA. IEEE Computer Society.

[14] Mateu García, Óscar. *Análisis y detección de objetos de primer plano en secuencias de video.* Proyecto fin de carrera. Universidad Politécnica de Cataluña. Junio 2009.

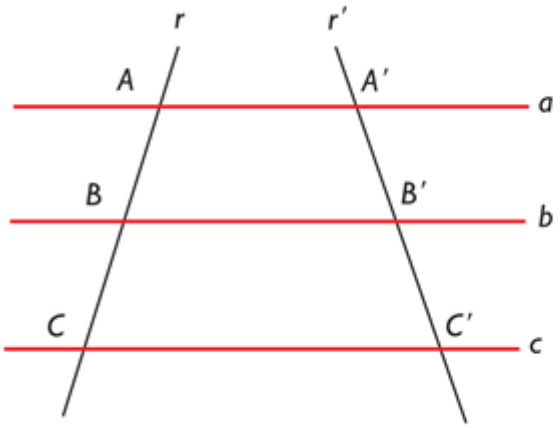
[15] Bustos Merino, Eduardo. *Operaciones Morfológicas*. Disponible en: https://sites.google.com/site/bustosmerino/home/operaciones-morfologicas

[16] Barragán Orlando, Diego. *Manual de interfaz gráfica de usuario en Matlab.* Disponible en https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf

[17] Corcuera, Pedro. *Creación de interfaces de usuario con MATLAB.* Dpto. Matemática Aplicada y Ciencias de la Computación. Universidad de Cantabria.

ANEXO 1: EQUIVALENCIA MEDIDA REAL EN MEDIDA PIXELS DEL VÍDEO:

Como es sabido, en un vídeo las medidas tomadas de la realidad no son las mismas que van a aparecer en el vídeo, por ello se ha desarrollado en el código posicionReal(1) para el eje X y posicionReal(2) para el eje Y a través del teorema de Thales:



$$\frac{AB}{A'B'} = \frac{BC}{B'C'} = \frac{AC}{A'C'}$$

Figura 1: Matemáticas del Teorema de Thales.

Estas son las dos líneas a las que me refiero del algoritmo, pero voy a realizar el cáculo del eje X:

```
tracks(trackx).posicionReal(1)=abs(centroid(1)-tracks(trackx).iniX)*(1+tracks(trackx).iniX/obj.pL2)*obj.L/(2*obj.pL2); %EjeX
tracks(trackx).posicionReal(2)=(centroid(2))*(tracks(trackx).posicionReal(1)-obj.L/2)/centroid(1); %EjeY
```

Figura 2: Cálculo de la posición del corredor en píxeles.

En el vídeo grabado, solo se conoce la longitud total que he corrido, exactamente 24 metros que corresponde con obj.L:

Figura 2: Frame cualquiera del vídeo.

Donde la aplicación del teorema de Thales se ha realizado de tal manera:

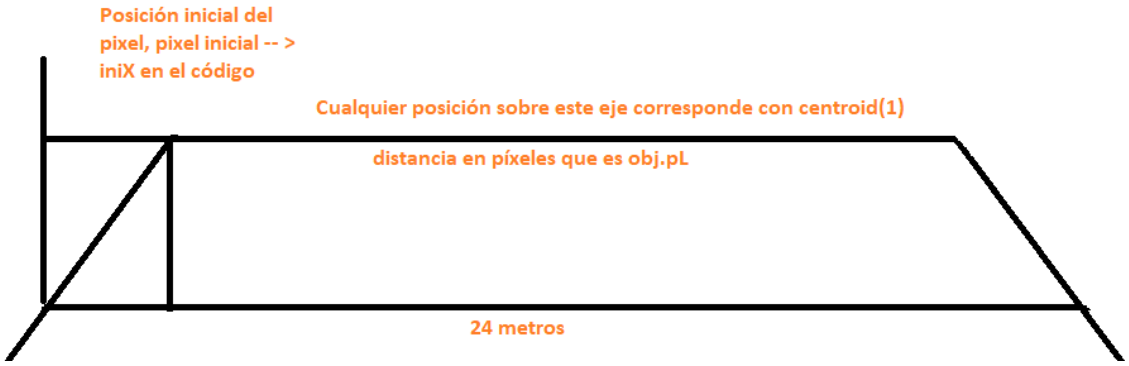


Figura 3: Punto de partido donde aplico el teorema de Thales.

Que en el vídeo corresponde a:

Figura 4: Correspondencia con figura 3.

Para calcular cualquier punto pixel de la equivalencia se va a realizar lo siguiente:

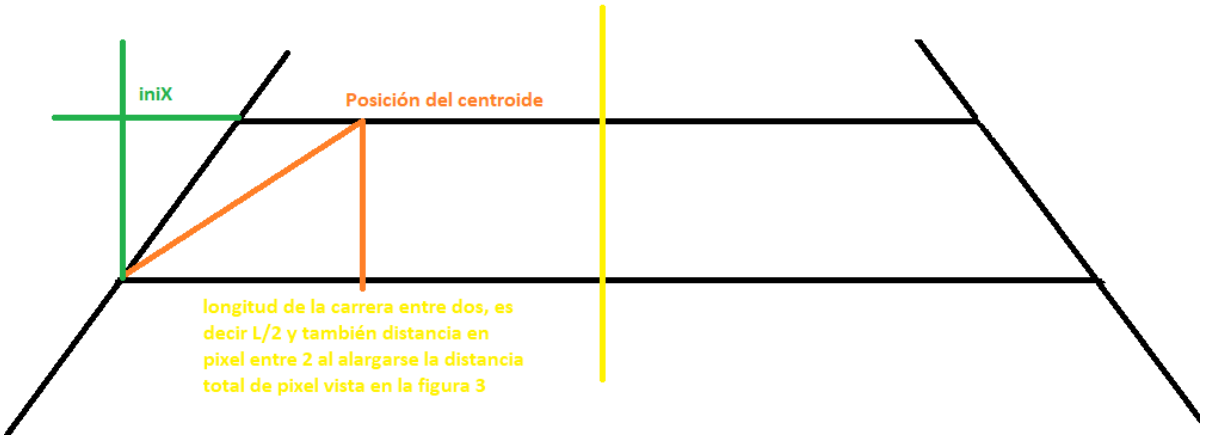


Figura 5: Dibujo para realizar los cálculos de las líneas del código antes mencionadas.

Por lo que la posiciónReal (1) va a tener el producto de los siguientes tres 3 factores:

11- El primero distancia en pixeles recorrida:

$$(\text{centroid}(1)\text{-tracks}(\text{trackx}).\text{iniX}) \quad (1)$$

12- Factor de corrección de la anchura de pixeles de la posición a la base:

$$1+\frac{tracks(tracksx).iniX}{obj.pL2} \quad (2)$$

13- Conversor de pixel a metros:

$$\frac{obj.L}{2*obj.pL2} \quad (3)$$

En conclusión por Thales la suma de que se produce al combinar (1) y (2) será:

$$\frac{(centroid(1)-tracks(trackx).iniX)+ tracks(trackx).iniX}{obj.pL2} = \frac{obj.L}{2*obj.pL2} \quad (4)$$

Donde los valores son:

14- iniX: Inicio de la carrera que se deberá prolongar para que tenga sentido la igualación.
15- centroid(1): Posición 1 del vector centroid donde se encuentra el corredor.
16- pL2: Distancia mitad en pixeles: Anchura del video partido de dos.
17- L: Longitud de la recta, 24 metros.