Anexo 1: Programa de cambios de plano

Como se ha comentado en la memoria, antes de comenzar con el programa de clasificación hemos tenido que desarrollar un pequeño programilla para detectar donde se producen los cambios de plano a lo largo del vídeo, ya que algunos algoritmos que vamos a utilizar se basan en la evolución de las imágenes a lo largo del tiempo y necesitamos establecer puntos de referencia donde comenzar a aplicarlos.

El programa que hemos desarrollado funciona recorriendo el video completamente y guardando en un fichero de texto el frame en el que se produce cada cambio de plano que detecte.

Creación del fichero de texto

En primer lugar necesitaremos crear el fichero en el que vamos a escribir los datos de los cambios de plano. Para ello se ha elegido un fichero de texto .txt de manera que también se pueda leer fácilmente fuera del programa para buscar datos si se necesitan.

Para trabajar con el fichero utilizaremos la clase ofstream de C++, que nos permite acceder a un fichero como si fuese un stream de datos y escribir en él. El nombre que se le dará al fichero creado será igual que el nombre que tenga el vídeo que analizamos y se encontrará ubicado en la misma carpeta que éste. Para ello solamente tenemos que cambiar la extensión .mp4 por .txt en el string que contiene la ruta del video.

Elección de espacio de color

Para el análisis de los cambios de plano, utilizaremos el espacio de color YUV, extrayendo la componente de luminancia (Y) de las imágenes. Este espacio de color es el utilizado en la representación del video analógico y consta de tres componentes. La componente Y se corresponde a la luminancia, que contiene la información de luminosidad de la imagen, mientras que las otras dos componentes contienen la información de color.

La imagen de la componente de luminancia será una imagen similar a una imagen en blanco y negro y la elección de esta componente se debe a que contiene la mayoría de la información relevante de la imagen, así podremos trabajar solamente con una componente.

La transformación desde componentes RGB al espacio de color YUV es una simple transformación lineal, por lo que apenas supone coste computacional.

Desarrollo del programa

Según obtengamos los frames desde el vídeo los iremos transformando al espacio de color YUV y separaremos sus componentes para quedarnos solamente con la Y. Una vez hecho esto procederemos al análisis del video.

Basaremos el funcionamiento del programa en analizar cómo evolucionan los valores de luminancia a lo largo del vídeo y cuando se produzca una variación muy grande de estos valores consideraremos que se ha producido un cambio de plano.

Al principio de cada plano necesitamos observar cómo se comporta la componente de luminancia durante los primeros planos, por eso supondremos que no se produce ningún cambio de plano durante los primeros 20 frames de cada plano. Este valor es bastante coherente, porque estamos analizando programas retransmitidos en televisión y los 20 frames aparecen en menos de un segundo trabajando con una tasa de 25 imágenes por segundo, por lo que es muy poco probable que se produzca un cambio de plano en ese intervalo.

Una vez tenemos los 20 frames calculamos la media de los valores de luminancia que hay en la imagen y la desviación típica de éstos con respecto a esta media. Estos valores se irán actualizando según se avance en el plano y se considere que aún no se ha producido un cambio de plano.

Utilizando el valor de la media y la desviación típica estableceremos un umbral a cada lado del valor medio que, cuando sea rebasado se considerará que se ha producido un cambio de plano.

Para definir los umbrales que podemos ver en la figura 19 tenemos que tener en cuenta que las imágenes de televisión son imágenes con un nivel de ruido bastante alto. Por ello como umbral utilizaremos un valor grande que será 4 veces la desviación típica de la luminancia. Cuando el valor del frame analizado supere este margen por encima o por debajo del valor medio consideraremos que se ha producido el cambio de plano.

Como se observa en la figura 19 el umbral se recalculará cada vez que una imagen no suponga un cambio de plano. De esta manera mantendremos un valor actualizado de la media y la desviación típica mejorando el funcionamiento del algoritmo.

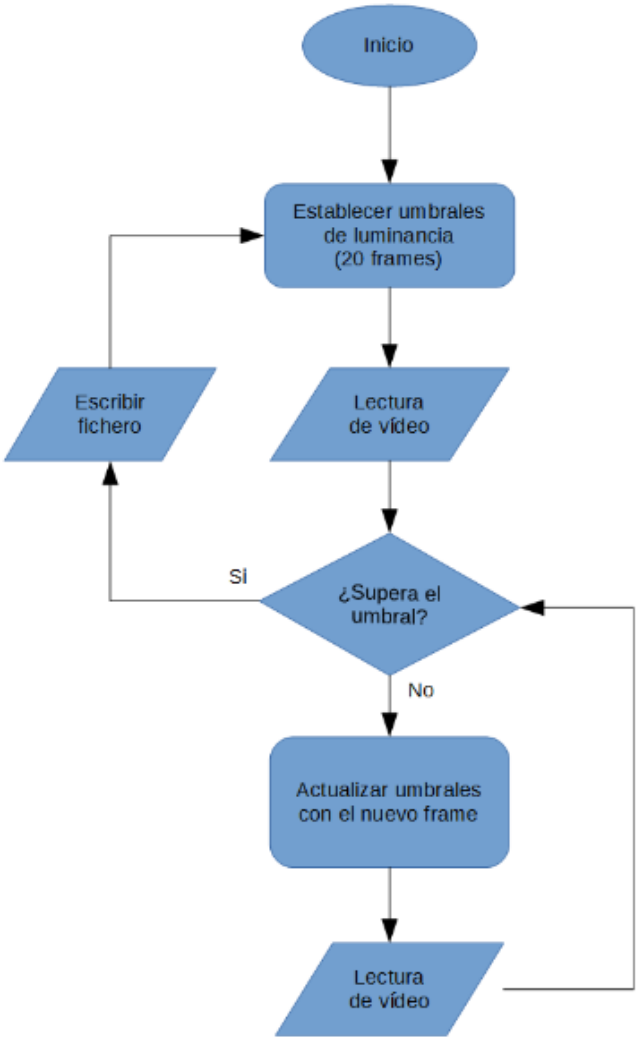

Figura 19: Diagrama del programa de separación de planos

Anexo 2: Métodos del programa

Para entender mejor los métodos que hemos creado en nuestro programa estudiaremos un poco más a fondo el funcionamiento de éste. Basándonos en el árbol de decisiones planteado anteriormente podemos establecer dos métodos de análisis distintos que serán aplicados cada uno a una categoría de video diferente.

Análisis de primeros planos

En primer lugar tomaremos el primer frame de cada plano y realizaremos la búsqueda de caras para determinar qué escenas consideraremos primeros planos y cuáles no. Como hemos comentado antes, utilizaremos un umbral para filtrar las caras que aparecen en la imagen. En este caso elegimos que solamente se queden las caras cuya área es mayor que el área total de la imagen dividida por 25. Después, con las caras que tenemos haremos un recuento y comprobaremos que no contamos con más de tres de ellas, además de calcular el movimiento a lo largo de varios frames y comprobar que es mínimo. En este caso consideraremos que la escena con la que estamos tratando es un primer plano.

Según vayamos detectando los primeros planos iremos almacenando los datos correspondientes hasta que se haya recorrido una cantidad de video suficiente para que consideremos que se ha alcanzado el estado estacionario y el valor que queremos analizar se mantiene prácticamente constante. Los valores que guardaremos serán la duración media de tanto primeros planos como de los que no lo son, y también almacenaremos la proporción de primeros planos que hay con respecto al total a lo largo del vídeo.

Cuando se alcance el estado estacionario, evaluaremos estos valores obtenidos y, si la duración media de los primeros planos es superior a 200 frames, tal y como se observa en la figura 8, y la relación de primeros planos con respecto al resto está entre el 15 y el 25%, consideraremos que el programa que estamos analizando es un informativo.

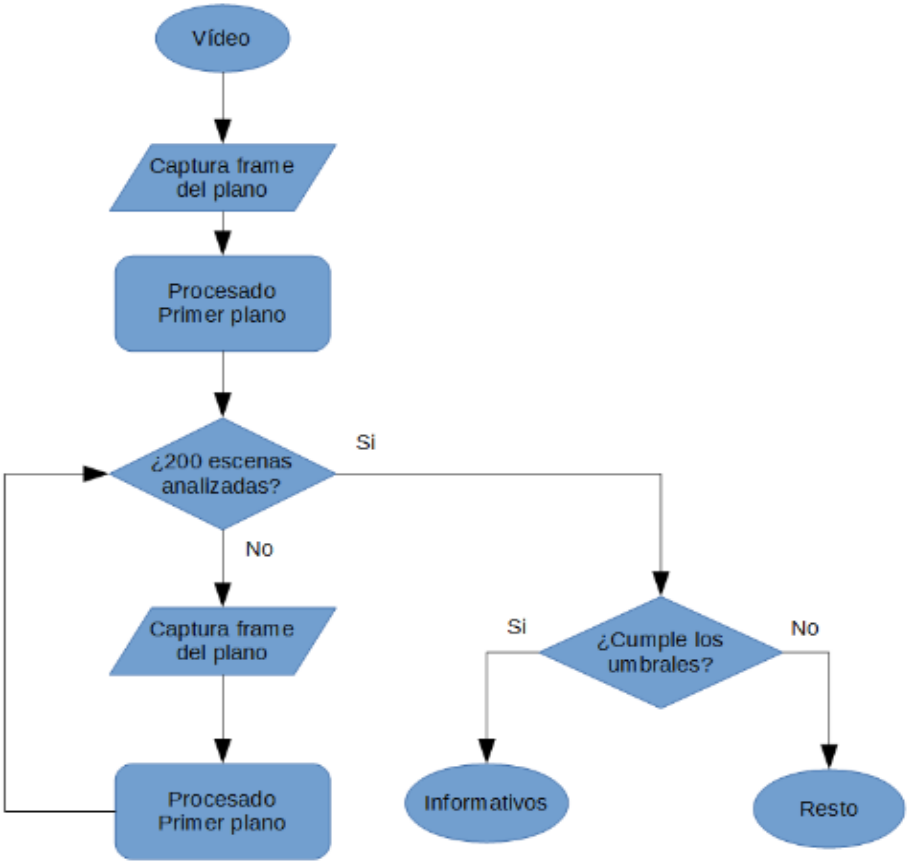

Figura 20: Diagrama del análisis de estructura basado en primeros planos

En la figura 20 queda plasmado el funcionamiento que acabamos de explicar. En él hemos considerado que el estado estacionario de los primeros planos se alcanza tras 200 cambios de plano.

Tras probar este método sobre varios programas, hemos observado que a veces se observan falsos positivos con programas en los que se entrevista a gente, porque el algoritmo detecta primeros planos y algunos pueden llegar a cumplir los requisitos de estructura. Sin embargo, como hemos nombrado anteriormente en el apartado 3.3.3, la duración de los programas de la categoría informativos también es un factor decisivo en el análisis. Por ello añadiremos una capa más en los umbrales donde se analizará la duración del programa entero y se restringirá a que ésta se encuentre entre 45 y 60 minutos.

Análisis del movimiento

En apartados anteriores se ha explicado como analizar el movimiento de una imagen y evaluarlo para decidir si las escenas que vemos pertenecen a un deporte o a otro. Ahora veremos cómo seleccionar esas escenas para que las escenas sean lo más propicias posible y que el algoritmo sea robusto. Para

realizar esto ordenaremos, en primer lugar, las escenas de mayor a menor duración.

Cuando tengamos ordenadas las escenas según su duración, aprovecharemos las propiedades de cada deporte para definir tres bloques de análisis. Estos bloques comenzarán en la duración máxima de las escenas, donde se centran las escenas más características de futbol, 750 frames, que se corresponde con 30 segundos y 375 frames para 15 segundos.

Dentro de cada escena extraeremos los vectores de movimiento y definiremos los rectángulos de los jugadores. Para asignar una escena a un deporte en especial contaremos el número de jugadores y analizaremos su distribución en la imagen. El color del campo también afectará a nuestra decisión quedando el decisor de la siguiente manera.

- Fútbol: De 2 a 20 jugadores distribuidos por toda la imagen con un campo de juego que toma valores en la componente de HUE de 38º a 55º.
- Baloncesto: De 2 a 10 jugadores distribuidos por toda la imagen con un color predominante del campo entre 9 y 21.
- Tenis: 1 o 2 jugadores situados en la parte central de la imagen, estando uno en la mitad superior y otro en la mitad inferior. El color del campo varía en función del material de la pista, por lo que no es un factor decisivo.
- El resto de escenas que no cumplan estos requisitos se contarán como pertenecientes a la categoría de otros.

A pesar de dividir en bloques el análisis para que los frames característicos de cada deporte aparezcan fácilmente, seguiremos teniendo intercalados una gran cantidad de planos que no nos interesan y estorbarán en el decisor. Por ello, para implementar el decisor se ha elegido utilizar un sistema de votación que analice varias escenas de cada bloque tal y como se observa en la figura 21 y almacene el número de escenas que pertenezcan a cada categoría. De esta manera, haciendo un recuento tras analizar los tres bloques podremos decidir a qué categoría pertenece el video.

Para la votación establecemos cuatro categorías que se corresponden con cada uno de los deportes analizados y el resto de videos. También, para asegurarnos que analizamos un número suficiente de escenas analizaremos las 100 primeras de cada bloque e iremos añadiendo el voto correspondiente de cada una a la categoría que pertenezca.

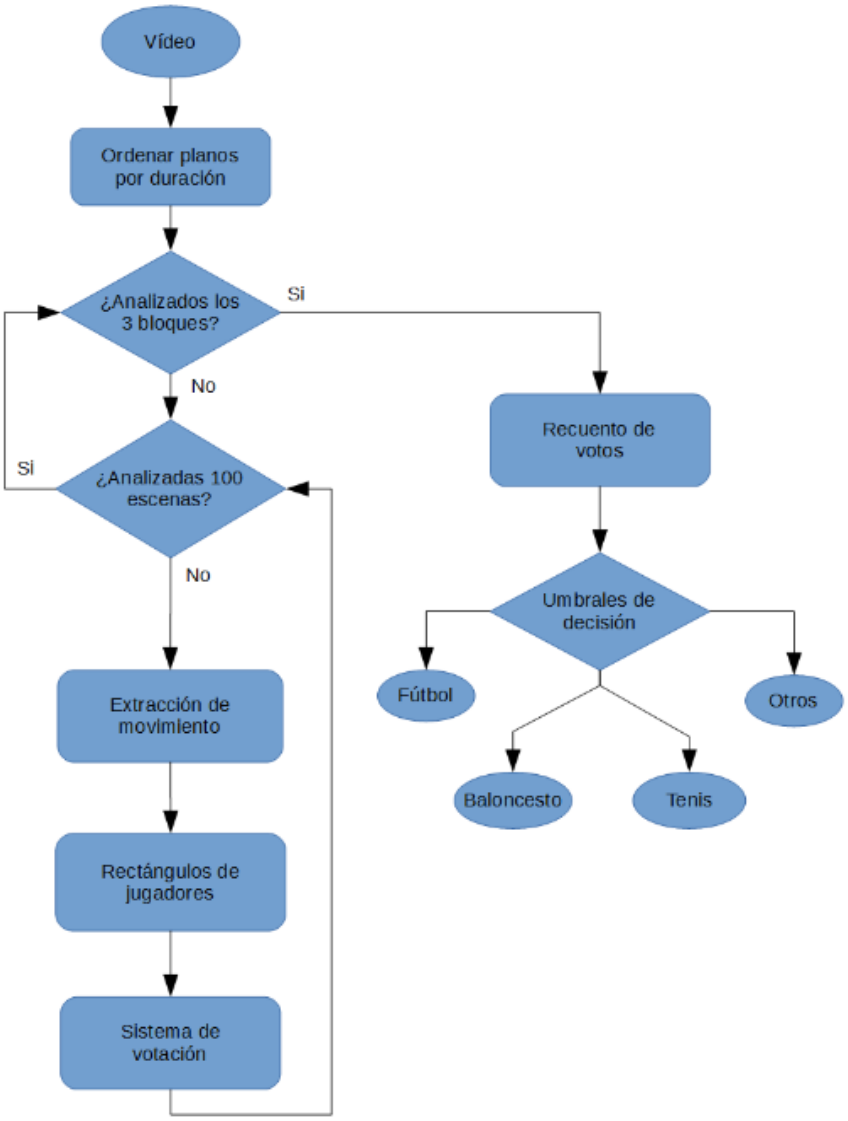

Figura 21: Diagrama del sistema de análisis de movimiento con votación.

Como el número de escenas no identificadas para uno de los deportes será normalmente mayor que el de las que sí que pertenecen a un deporte, evaluaremos si el número de escenas correspondiente a otros tipos de programas supera un valor que calcularemos como el doble de la suma de los otros tres valores. Si se cumple esta condición consideraremos que el video analizado se encuentra en la categoría de otros, mientras que si no se cumple, el deporte que tenga el máximo número de votos será el elegido.

En la figura 21 se puede observar el proceso entero de análisis de las escenas en cada bloque así como la aplicación del sistema de votación.

Métodos desarrollados

Ahora se explicará con detalle cada uno de los métodos programados para realizar los dos análisis que acabamos de explicar.

Análisis de estructura y primeros planos

LeeCambioPlano: Esta función se encarga de recorrer el fichero de texto que contiene los cambios de plano y devuelve el número de frame en el que comienza la siguiente escena dentro del vídeo.

CuentaCambiosPlano: Método utilizado para determinar la longitud del fichero de texto con los cambios de plano que devuelve el número de líneas de éste y, por consiguiente el número de cambios de plano que hay en el video.

CalculaMov: Función que evalúa de manera básica la cantidad de movimiento que ha habido entre dos imágenes comparando cada pixel con su correspondiente en la imagen siguiente. Si éste varía mucho de una imagen a la siguiente se incrementa un contador que acabará representando la cantidad de pixeles donde ha habido movimiento.

DetectFace: Función que aplica de manera sencilla todos los parámetros necesarios para ejecutar el clasificador en cascada sobre una imagen y devuelve un vector de rectángulos que contienen cada una de las caras que se ha detectado en la imagen.

IsPrimerPlano: Método que aplica los criterios comentados anteriormente para filtrar las caras sobrantes de una imagen y decidir si la imagen analizada se trata de un primer plano.

CuentaPrimerosPlanos: Función que se encarga del recuento de primeros planos para actualizar los valores de duración media y la relación de primeros planos con respecto al video completo. Es llamada cada vez que se analiza una imagen para actualizar los valores comentados.

Análisis de color y movimiento

HSVHist: Método que calcula el histograma de un frame en el espacio de color HSV.

IsUniforme: Función que utiliza el histograma calculado por la función anterior para decidir si existe una superficie de color uniforme en ese plano. Esta función se encuentra sobrecargada de manera que en función de los parámetros que le introduzcamos devuelva unos valores u otros dependiendo de lo que nos interese.

MaskHistHUE: Función que aprovecha la componente de HUE de una imagen y su histograma para crear una máscara que tenga a uno los píxeles

donde el valor del color se encuentra dentro del rango establecido y a cero todos los demás.

AbsOptFlowMap: Método que evalúa la matriz del flujo óptico entre dos imágenes y define una máscara en la que se muestran los píxeles cuyo movimiento es probable que pertenezca a un jugador en un deporte.

MotionDetectDense: Método que aplica el algoritmo de Farneback para calcular el movimiento que se produce entre dos imágenes para cada uno de los píxeles de éstas. Además utiliza el anterior método para definir los posibles jugadores y devuelve los arrays de puntos que representan los contornos de cada una de las siluetas de los jugadores.

Resize: Función que modifica la posición de los cambios de plano para unir dos o más si es necesario y se ha detectado que los cambios de plano no se producían de verdad.

OrdenaCambioPlano: Método que analiza la imagen previa a un cambio de plano y la siguiente y las compara para comprobar que de verdad el cambio de plano detectado en ese punto es un cambio de plano. Si detecta que se ha colado algún cambio de plano donde no debería llama a la función **resize** para corregir el fallo.

VotaciónDeporte: Función que evalúa una escena y decide si esta forma parte de un partido de fútbol, baloncesto, tenis o ninguno de estos, "depositando" el voto correspondiente en la categoría correcta.

RecuentoVotos: Método que lee los datos de la votación y devuelve qué tipo de programa es el que estamos analizando.

ProcesaMovimiento: Función que aplica todos los métodos anteriores para obtener los datos de movimiento necesarios de una escena y llama a **VotacionDeporte** para que aplique el voto a una de las categorías.

ProcesaVideo: Método principal de la clase del clasificador. Éste es llamado una vez se han inicializado todos atributos de la clase **DecisorProg** y aplica de manera automática todos los métodos y algoritmos necesarios para realizar el análisis del vídeo y clasificarlo en una de las categorías.