CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Universidad Carlos III de Madrid e-Archivo

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES

SEGMENTACIÓN DE CAMPO DE TENIS EN SECUENCIAS DE VIDEO

Alberto Sánchez González

Tutor: Jesús Cid Sueiro

Indice

INTRODUCCIÓN

En los últimos años las retransmisiones deportivas han despertado un creciente interés, con audiencias de millones de espectadores en todo el mundo. El aumento de la audiencia conlleva también un aumento en la demanda de información por parte de los espectadores ya sea antes, durante o después de la retransmisión. Esta demanda de información está fomentando la investigación de métodos de reconocimiento automático de eventos, ya sea para utilizarlos en tiempo real, cómo para su posterior análisis y clasificación.

En este trabajo pretendemos diseñar un software de detección y segmentación de campos de tenis en secuencias de video de partidos de tenis reales, que pueda servir de base para futuros desarrollos en el tratamiento de imágenes en tiempo real o el posterior análisis de las mismas.

Para ello comenzaremos analizando el estado del arte en lo que a detección y segmentación de imágenes se refiere, centrándonos en modelos de detección de eventos para videos de tenis.

Posteriormente marcaremos unos objetivos para nuestro sistema y desarrollaremos el mismo, implementando las mejoras necesarias para minimizar los errores que puedan surgir a lo largo de nuestro desarrollo.

Y finalmente analizaremos nuestro sistema, estudiaremos los resultados obtenidos y se plantearán nuevas líneas de investigación a tenor de las conclusiones alcanzadas.

ESTADO DEL ARTE

Muchas líneas de investigación actuales tienden a la detección de eventos, a la extracción y al análisis de características semánticas en videos deportivos

La segmentación del campo de juego en videos deportivos está cobrando una importancia relevante. Esta segmentación sirve como soporte inicial para la extracción de características de medio y alto nivel.

Hay varias técnicas de segmentación aplicadas a diferentes deportes (14)(15) donde el problema principal es identificar la zona del campo en la imagen para obtener la información semántica necesaria.

Se propone realizar esta segmentación analizando los histogramas de color, detectar el color dominante y en torno al valor de este color establecer un rango que corresponda al campo a segmentar. También se propone detectar el color de las líneas y estimar dichas líneas.

Figura I. Imágenes de varios deportes donde es aplicable la segmentación del campo de juego

Centrándonos en el deporte del tenis, el sistema informático más importante que se aplica durante el desarrollo del juego es la tecnología de el ojo de halcón (Hawk-eye) (3) (4). Consiste en un sistema que genera la trayectoria que recorre la pelota y puede ser utilizado por los jueces para resolver jugadas dudosas donde no se ha podido determinar a simple vista si la pelota bota dentro de los límites del campo. Se basa en cálculos de triangulación a partir de las imágenes visuales y mediciones proporcionados por cámaras de video de alta velocidad.

Se necesitan al menos 4 cámaras colocadas estratégicamente alrededor de la zona de juego para registrar las imágenes, las cuales serán transmitidas a un procesador de alta velocidad que reconoce las imágenes y calcula su trayectoria. Se necesita además un modelo del área de juego, la posición de las cámaras y el lugar de enfoque. Utilizando las leyes físicas, se interpola las posiciones calculadas para recrear la trayectoria de la pelota y generar la imagen de la trayectoria junto a su interacción con la zona de juego (figura II). La generación de la imagen tarda unos segundos, con lo que la interrupción del juego es bastante corta.

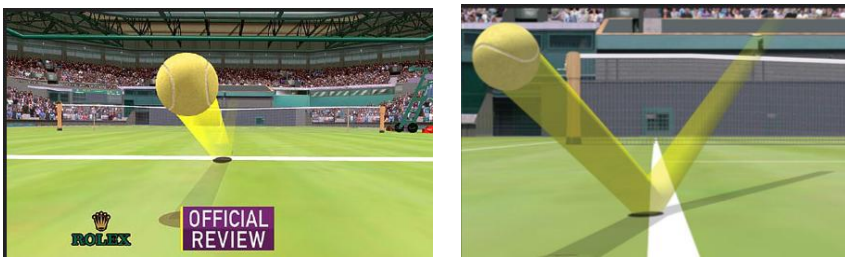


Figura II Simulación de la trayectoria creada con el ojo de halcón

Además de este sistema, hemos encontrado otros modelos de detección de eventos para partidos de tenis. En (5) se muestra un modelo que reconoce automáticamente que ha ocurrido durante los puntos jugados y detecta qué jugador ha ganado el punto, contando automáticamente los tantos. Para detectar estos eventos de alto nivel, inicialmente nos proponen un pre-procesado de las imágenes, un análisis espacio-temporal entre los frames y el reconocimiento del campo de juego.

En otro artículo (12) nos presentan un método para reconocer las acciones básicas de los jugadores en retransmisiones de video de tenis. A diferencia de otros, este método se basa en el análisis del movimiento y considera la relación entre los movimientos de las diferentes partes del cuerpo y de las regiones en el plano de la imagen. Indica además que el reconocimiento y clasificación de las acciones de los jugadores es una tarea difícil debido a la bajea resolución de los jugadores en los fotogramas, ya que estos se representan en una mínima sección de pixeles respecto de la imagen.

Vemos que en los sistemas de detección de eventos estudiados, se basan en un procesado primario, detección y segmentación de la pista, para a partir de ahí desarrollar el software de detección de eventos de alto nivel. En (13) se plantea también la detección de eventos basados en el análisis de las trayectorias de los jugadores, donde se utiliza un filtro de Kalman adaptativo para mejorar la precisión en el seguimiento de los jugadores. Además de realizar un análisis temporal de los fotogramas para eliminar aquellos de menor importancia que no muestren un evento significativo.

Como se ha visto, en la detección de eventos de alto nivel en eventos deportivos es necesario realizar una correcta segmentación del campo de juego. Se plantean varias formas para realizar la segmentación, de las cuales obtenemos ideas que podemos aplicar en el desarrollo de nuestro sistema.

Pues bien, en nuestro caso la detección de color dominante en una imagen puede resultar de cierta utilidad para realizar la segmentación en videos de tenis. No podremos establecer un patrón fijo de color, ya que las diferentes superficies y pistas donde se disputan los grandes partidos de tenis varían según el lugar o el torneo. Por lo que debemos desarrollar un sistema que sea versátil, pero no por ello deje de ser robusto.

Además se plantea la detección de las líneas que limitan el campo y la estimación de dichas líneas por medio de algún algoritmo. En este caso, deberemos plantear un algoritmo que nos permita detectar los bordes del campo de juego.

Por tanto avanzaremos y estudiaremos ambos conceptos para poder realizar una correcta segmentación del campo, de una forma versátil y robusta, e intentando ser lo más eficiente posible.

Objetivos y punto de partida

Las diversas tecnologías, sistemas y aplicaciones están orientados a la detección de eventos durante la disputa del juego en partidos de tenis o al posterior tratamiento de las imágenes para la detección de dichos eventos. La segmentación del campo de juego o la detección de un modelo del área de juego aparecen como algo básico y necesario para implementar cualquiera de los sistemas mencionados.

Por tanto, nuestro punto de partida será desarrollar un sistema robusto y fiable de segmentación de campo de tenis a partir de secuencias de video, que sirva como base para futuros desarrollos de detección de eventos de alto nivel.

Este sistema deberá acercarse en la medida de lo posible y con los medios a nuestro alcance, a realizar la segmentación de los límites del campo de juego en tiempos de cómputo similares a los que podría ofrecer un sistema de análisis en tiempo real.

Nuestro desarrollo se dividirá en dos análisis distintos, la extracción de bordes y la detección del color dominante. Por tanto intentaremos demostrar si la combinación de ambas técnicas puede ofrecernos garantías de alcanzar los objetivos deseados.

Debido a la complejidad que podría suponer analizar los diferentes tipos de planos y secuencias en la retransmisión de un partido de tenis, nuestro análisis se centrará en el plano general y más utilizado en las retransmisiones de este tipo de de eventos.

DESARROLLO, APLICACIÓN Y RESULTADOS

El objetivo es implementar un sistema que realice una segmentación del campo de tenis en tiempo real. Generalmente, en las retransmisiones de partidos de tenis durante el tiempo que un punto está en juego, la cámara muestra un plano general de todo el campo de juego, donde se puede cubrir con una cámara prácticamente estática los movimientos de ambos jugadores y la trayectoria que recorre la bola en cada golpeo.

Por ello, nuestro análisis se va a centrar en la segmentación del campo de juego para el plano general más empleado en las retransmisiones de partidos de tenis.

Figura III. Muestra del plano general en la retransmisión de encuentros de tenis

Partiremos de dos análisis bien diferenciados, la extracción de la máscara del campo por medio de la detección del color dominante y la detección de líneas. Se espera que con la combinación de ambas técnicas obtengamos óptimos resultados.

Emplearemos dos bases de videos, una base de entrenamiento para desarrollar e implementar las mejoras de nuestro sistema y otra base de test para determinar la robustez y eficiencia de nuestro sistema.

Las bases de videos utilizadas se han obtenido mediante descarga por el sistema de intercambio de archivos “emule”, con sistema P2P utilizando el protocolo eDonkey 2000 y la red Kad, publicado como software libre para sistemas Microsoft Windows.

Para facilitar el procesado de las secuencias en Matlab, se ha unificado el tipo de archivo en tipo “AVI Video file” (.avi) para todos los clips empleados de nuestra base de videos.

Cada base consta de 8 secuencias de video, con una duración comprendida entre 5 y 15 segundos. Se eligieron secuencias cortas para evitar ralentizar el sistema en el momento de cargar las secuencias en Matlab.

Estos clips de video no tienen la misma resolución, lo que hará que nuestro sistema tenga que ser adaptable y versátil frente a las diferentes señales de entrada, ya sea definición estándar o alta definición. Además podemos determinar cuan de optimo es nuestro sistema para las distintas resoluciones analizadas.

También los videos incluyen partidos disputados en diferentes superficies para intentar que el sistema pueda ser robusto frente a distintos tipos de pista, por lo que se incluyen partidos en todos los tipos de superficies empleadas en el circuito mundial (pista dura, arcilla, hierba y moqueta).

Jugadores y pista	Resolución	Fotogramas por segundo	Alta definición
Djokovic - Nadal – Open Australia	360x480	29fps	NO
Tipsarevic - Del Potro - Delray Beach	720x1080	50fps	SI
Federer - Verdasco - Us open (Nueva York)	720x1080	50fps	SI
Nadal vs Murray - Japón	720x1080	50fps	SI
Nadal vs Djokovic - Madrid	720x1080	50fps	SI
Federer vs Nadal – Wimbledon (Londres)	270x480	25fps	NO
Federer vs Roddick - Melbourne	270x480	25fps	NO
Djokovic - Nadal - Rolang Garros	720x1080	50fps	SI

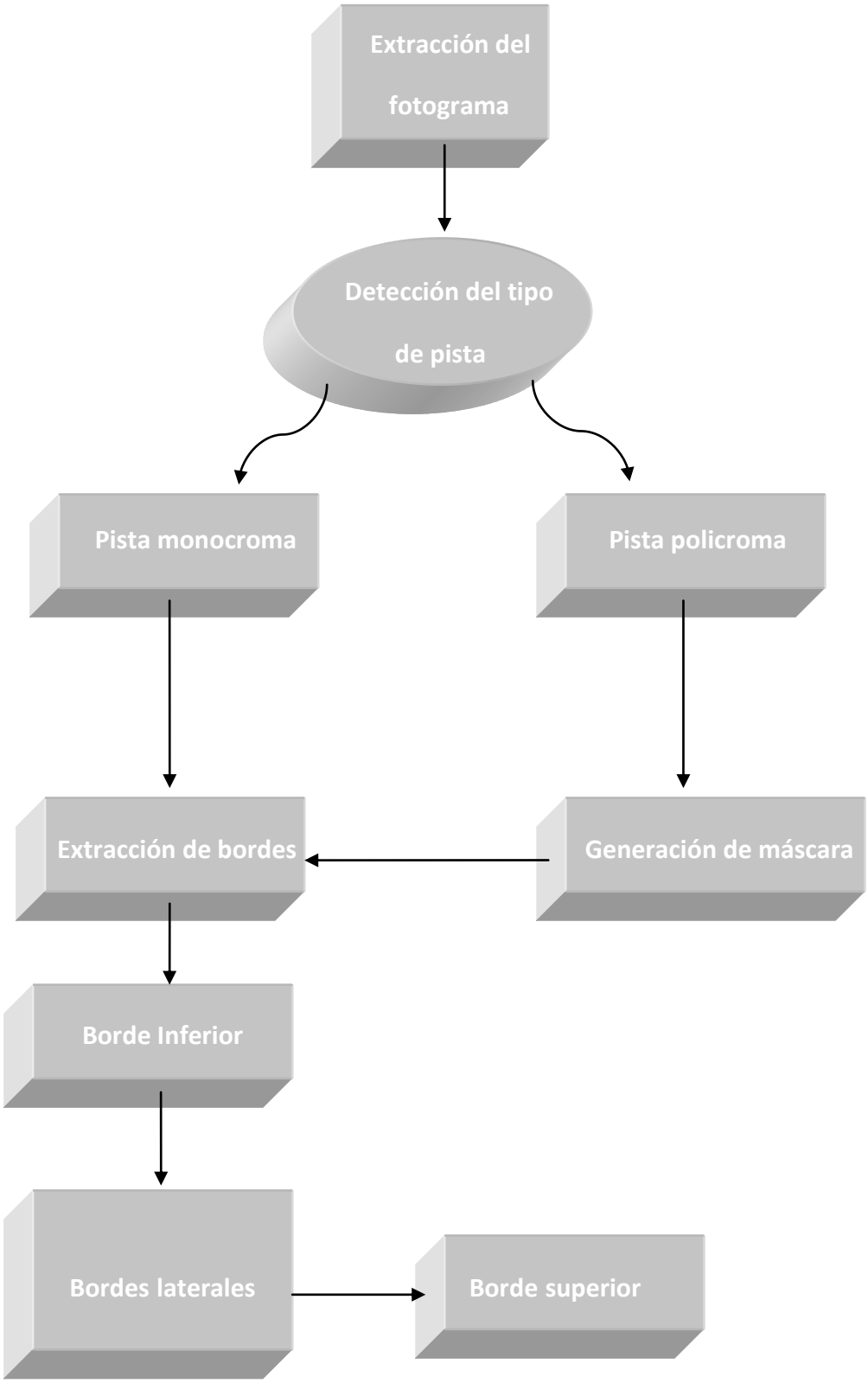
Tabla con la base de videos empleados para el test

Se ha utilizado el software Matlab R2009a (versión 7.8.0.347) para 32 bits (win32), instalado en un ordenador portátil Acer (modelo AO751h) con procesador Intel Atom CPU Z520 1.33GHz con memoria RAM de 2 Gb, sistema operativo Windows Vista Home Basic.

Para los test de rendimiento del sistema se ha utilizado un equipo más potente. Se ha utilizado un pc de sobremesa Packard Bell (modelo S3810) con procesador Intel Core i3 CPU 540 3.07GHz con memoria RAM de 4 Gb, sistema operativo Windows 7 Home Premium.

El sistema a implementar básicamente se compondrá de la extracción del fotograma a analizar. Se detectará el tipo de pista y en caso de que esté compuesta por más de un color se generará la máscara por el color dominante, sino se omitirá este paso. Después de realizará la extracción de los bordes y generaremos la máscara definitiva.

Diagrama del sistema a implementar



Extracción de los fotogramas

Para obtener los fotogramas de las secuencias de video, cargamos la secuencia de video en Matlab mediante la función *mmreader* incluida en el *Image Processing Toolbox*. Esta función es muy completa, ya que únicamente introduciendo como valor de entrada el nombre del clip de video seguido de la extensión, construye un objeto multimedia el cual almacena distintos parámetros de la secuencia de video, tales como la duración en segundos, número de bits por pixel, número de fotogramas por segundo o las dimensiones del clip de video analizado. Estos parámetros podrían ofrecer cierta información de utilidad a lo largo del desarrollo de nuestro sistema.

Duration	7.0600	7.0600	7.0600
Name	'FedererHD.avi'		
Path	'C:\Users\Alberto\De...		
Tag	''		
Type	'mmreader'		
UserData	[ ]		
BitsPerPixel	24	24	24
FrameRate	50	50	50
Height	720	720	720
NumberOfFrames	353	353	353
VideoFormat	'RGB24'		
Width	1280	1280	1280

Figura IV. Valores que extrae la función mmreader

Empleamos el método *read* de la clase *mmreader* para extraer los fotogramas de la secuencia de video. Introduciendo cómo valor de entrada el objeto construido anteriormente, el método extraerá todos los fotogramas que contenga la secuencia.

Esto computacionalmente puede ser muy costoso, por lo que también el método ofrece la posibilidad de extraer un fotograma. Para ello se debe añadir cómo valor de entrada un número entero positivo menor o igual que el número de fotogramas totales de la secuencia de video, el cual indicará el fotograma que se extraerá.

Para facilitar el desarrollo de nuestro sistema vamos a tratar inicialmente con fotogramas, lo que nos permitirá ajustar las correcciones y mejoras para el correcto análisis de las secuencias.

Una vez extraído el fotograma, realizamos una conversión del fotograma con la función *im2double* del *Image Processing Toolbox*, que cómo se explica en (9) resulta necesaria para el tratamiento de los fotogramas. Los valores de tipo uint8 (1byte por pixel) requieren menos recursos para almacenarse y representarse que una imagen de tipo double (8 bytes por elemento). Sin embargo con los valores de tipo entero (como

uint8), Matlab no permite aplicar directamente casi ninguna operación. Por ello realizamos la conversión con esta función prediseñada de Matlab.

Extracción máscara del campo mediante componente de color

Tras la conversión realizada, procedemos a la detección del color dominante de la imagen, que cómo se ha explicado en el comienzo de este capítulo, por el tipo de planos que se emplean en los videos a analizar (figura III), la tonalidad dominante en la imagen coincidirá en una alta probabilidad con el color del interior del campo a segmentar.

Este método puede resultar muy eficaz dependiendo del tipo de video que nos dispongamos a tratar.

En el tenis, muchas de las superficies en las que se disputan los grandes torneos tiene el color de una sola tonalidad y las líneas que delimitan el campo están pintadas sin que se produzca una variación cromática en la superficie de juego del interior de las líneas limítrofes respecto al exterior (figura V). Esto puede suponer un problema, ya que al intentar detectar los límites del campo por medio del color dominante de la imagen, el sistema no tendrá resultados óptimos porque se detectará cómo campo tanto hacia el interior de las líneas cómo hacia el exterior de las mismas.

Figura V. Ejemplo de pistas donde no existe variación cromática entre el interior y el exterior del campo de juego

Aunque suponga un problema para terrenos de juego monocromos, esta técnica puede ser de gran ayuda en terrenos de juego policromos (figura VI), donde el cambio de tonalidad entre el interior y el exterior de la pista es más que evidente, por lo que avanzaremos y desarrollaremos la idea inicialmente planteada.

Figura VI: Ejemplo de pistas donde sí existe variación cromática entre el interior y el exterior del campo de juego

Este método puede ser muy efectivo para detectar los límites de la pista cuando se esté disputando encuentros de dobles, ya que se emplea la totalidad de la pista para disputar el juego. Sin embargo, para partidos individuales las dos zonas laterales de la pista no se consideran campo, por lo que el método puede no ser tan efectivo frente a estos casos.

Lo primero que vamos a hacer para extraer la componente de color del campo de tenis es crear el histograma de la imagen para obtener el color predominante.

Para ello antes se realiza la conversión de imagen al espacio HSV, donde la imagen en RGB se transforma en una imagen con 3 componentes; la primera contiene la información del color y las otras dos la información de la saturación de color y del brillo de la imagen. Para ello se emplea la función de Matlab *rgb2hsv* (*ver Anexo El modelo de color HSV)*.

Extraer la información de color resulta muy útil, porque no hay variaciones de brillo o saturación, y podemos obtener el color predominante en la imagen. Con la componente de color extraída, se crea el histograma de la componente H donde se puede obtener el color predominante en la imagen con una probabilidad de acierto muy alta. El histograma se crea mediante la función *imhist*, la cual a partir de una imagen en escala de grises crea el histograma en un rango de 256 valores, por lo que la resolución de H será de 0,004 o lo que es igual de 1,44°.

Cómo se ha comentado anteriormente, este análisis de la componente de color va a ser muy útil para detectar los límites del campo en superficies que tengan un color distinto para el interior y el exterior del campo de juego. Sin embargo para superficies monocromas este análisis no nos va ofrecer información de interés, ya que aunque detectemos correctamente el color del campo, identificará como campo toda la superficie alrededor del mismo, incrementando significativamente la tasa de error.

Además el histograma nos puede ofrecer información relevante sobre la cual se puede desarrollar un método para que el sistema decida si nos encontramos ante una pista monocroma o una pista con más de un color dominante. Viendo las características estadísticas de la señal se podría determinar ante qué tipo de pista nos encontramos. Este método lo desarrollaremos más y ampliaremos en el apartado de futuros desarrollos.

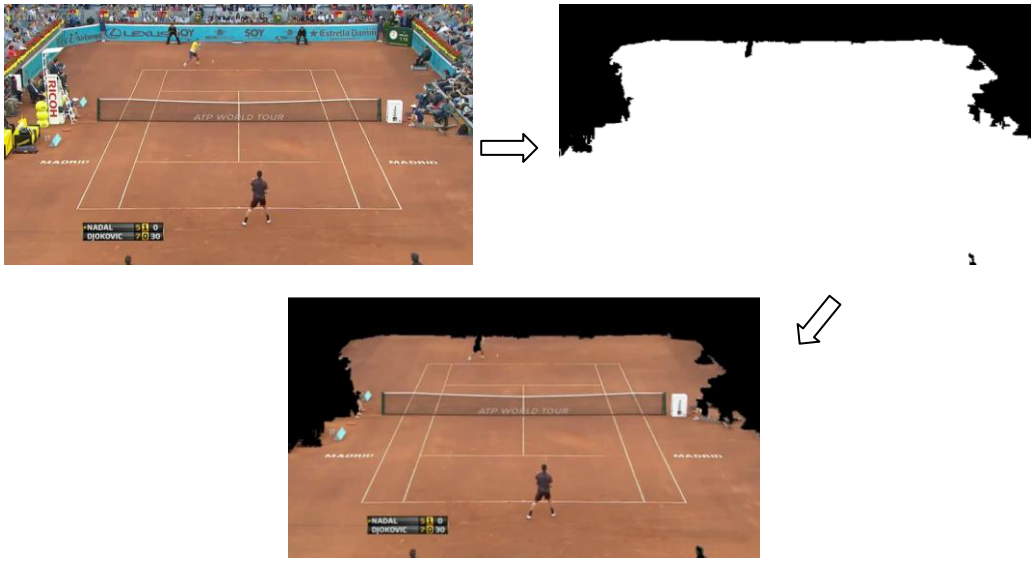


Figura VII. Imagen original de una superficie monocromática. Máscara generada ante imagen monocromática. Salida del sistema ante imagen monocromática.

Una vez se ha creado el histograma, buscamos el valor más repetido por medio de la función *mode*, la cual nos dará información sobre la tonalidad más repetida en los pixeles de la imagen. Pero aunque el valor coincida con pixeles que representan la imagen del campo a segmentar, el color de dicho campo puede no ser homogéneo y presentar ligeras variaciones en la tonalidad. Por tanto se determina un rango en torno al valor que cubra los diferentes tonos que puede presentar la pista. Después de realizar varias simulaciones se escoge que el rango más adecuado está en ± 0,04.

Por tanto el rango elegido abarcará 0,09/1 de los valores de H (32´4° en la representación de H en el espacio HSV), lo que nos optimiza la detección de los pixeles de campo. Como se dijo anteriormente, el histograma se representa por 256 valores que son muestras de las distintas tonalidades, por lo que nuestro sistema con el rango elegido abarcará 22 tonalidades distintas más la tonalidad del color más repetido en el histograma. Haciendo un total de 23 tonalidades respecto a las 256 del rango total de colores.

Una vez se ha concretado el rango, se crea nuestra primera máscara del sistema, poniendo a “0” los pixeles que no se encuentren dentro del rango establecido, y a “1” los que están dentro del rango.

Podemos observar que con la máscara generada, aunque detectamos una tasa bastante alta de los pixeles da campo, también se identificarían cómo pixeles de campo una importante cantidad de pixeles erróneos, producidos por elementos externos al campo a segmentar (vallas, elementos del público, juez de silla, etc…).

También el sistema ha identificado dentro de los límites del campo a segmentar varios pixeles erróneamente, producidos por los jugadores, la red o las líneas interiores.

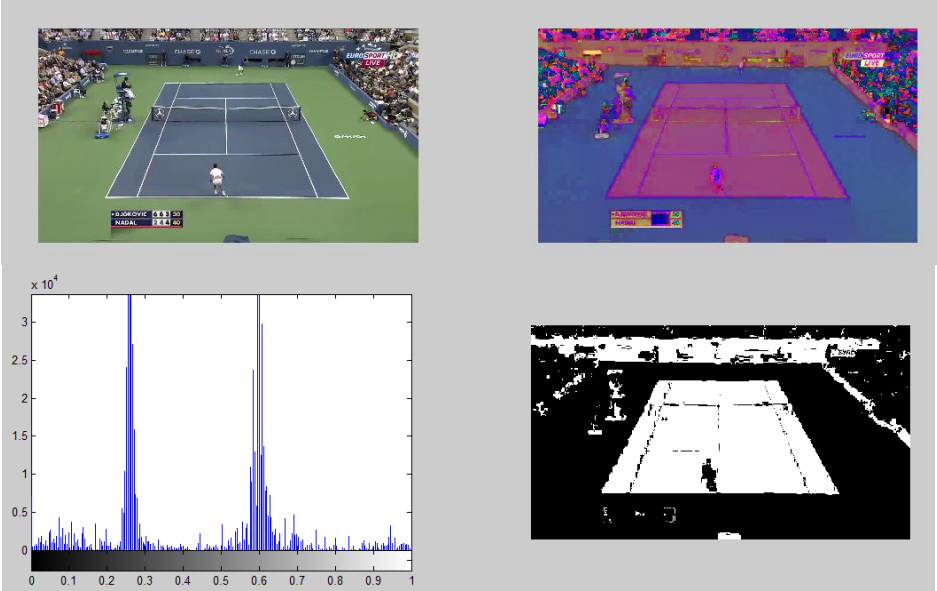


Figura VIII. Muestra del proceso implementado hasta generar la primera máscara

Inicialmente se planteó para eliminar los dos tipos de errores aparecidos realizar varias operaciones morfológicas sobre la máscara obtenida. Primero se erosionaba la máscara y se dilataba con el mismo patrón, con lo que se conseguía eliminar gran parte de los pixeles que se habían detectado cómo campo de tenis pero se encontraban fuera de los límites de la pista. Posteriormente se volvía a realizar el mismo proceso pero a la inversa, primero dilatando y después erosionando con el mismo patrón pero algo mayor que el anterior, con lo que se rellenan los pixeles que se encontraban dentro de los límites de la pista pero estaban en negro.

Aunque este sistema obtenía resultados satisfactorios para algunas imágenes, resultaba ser muy poco eficiente para la base de videos de entrenamiento, por lo que finalmente se descartó por otro método que fuera más efectivo.

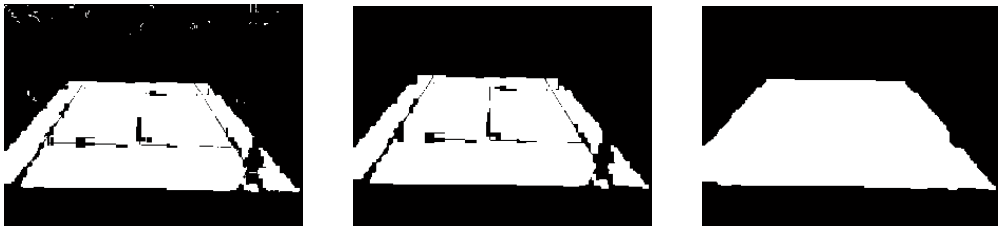


Figura IX (a). Ejemplo de mascara correcta al emplear operadores morfológicos erosión-dilatación

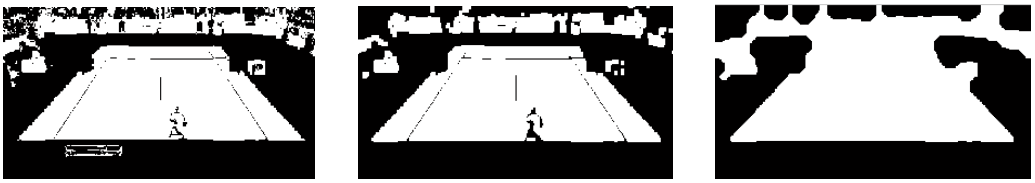


Figura IX (b). Ejemplo de mascara errónea al emplear operadores morfológicos erosión-dilatación

El método que finalmente se eligió para minimizar estos errores fue emplear la función *imfill,* que realiza la operación morfológica de rellenado en una imagen binaria. Esta operación aplicada a nuestra máscara nos rellena únicamente los pixeles negros que están rodeados por pixeles blancos. Por tanto, este rellenado nos elimina los pixeles mal interpretados como de campo, pero no soluciona el error de haber interpretado pixeles de campo que se encuentran fuera de los límites del propio campo. Para corregir esta mala interpretación podemos aplicar la función *imfill* a la máscara negada, con lo que conseguiríamos rellenar todos los pixeles blancos que se han identificado erróneamente cómo de campo. Pero además, eliminaríamos también los pixeles blancos del campo identificados correctamente, ya que la función *imfill* rellena los huecos negros rodeados de pixeles blancos salvo que estén tocando con los bordes de la imagen.

Para subsanar esto, antes de aplicar la función *imfill* a la máscara negada, se añade una línea horizontal de pixeles negros desde la mitad del lateral izquierdo de la imagen hasta el lateral derecho de la imagen pasando por el centro de la imagen. Con esto conseguiremos que los pixeles que componen el campo no se eliminen al ejecutar la función de rellenado. Además antes de añadir esta línea horizontal, se pondrán a 1 todos los pixeles que componen los bordes de la imagen en la máscara negada. ¿Por qué hacemos esto? Pues bien, hemos encontrado algunos casos en que algún que elemento identificado erróneamente como pixeles de campo se encontraba pegado a los bordes de la imagen, por tanto al realizar el rellenado de la máscara negada no conseguíamos eliminarlo.

Después de aplicar lo expuesto anteriormente invertimos de nuevo la máscara resultante y obtenemos nuestra máscara definitiva.

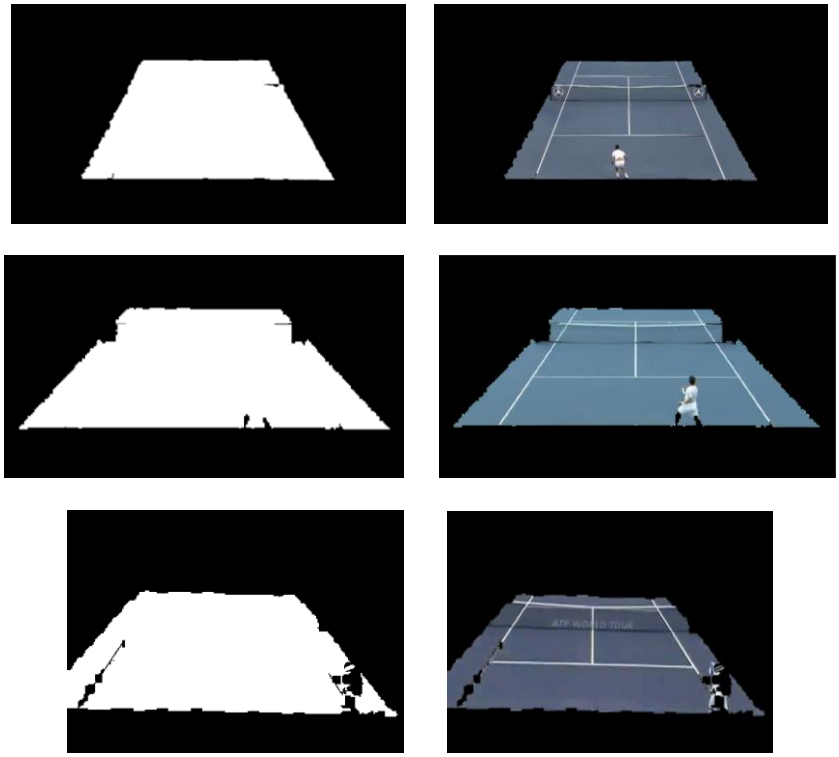


Figura X. Ejemplos de las máscaras obtenidas al finalizar el proceso

Mejoras

Una vez creado nuestro sistema para la extracción de la máscara por medio de la componente de color, lo hemos probado con una base de varios videos de prueba para poder descubrir los errores más comunes y poder implementar algunas mejoras que nos ayudara a conseguir una tasa de acierto mayor.

- Nos han aparecido ciertos casos que el valor máximo del histograma de color o valor más repetido no se correspondía con el color del interior del campo. Esto nos ocurría cuando el plano de la imagen es demasiado lejano. Ya que aunque la totalidad del campo se encontraba dentro de la imagen, otro color está más repetido en la imagen y el sistema detectaba los pixeles con este color cómo campo. En la figura XI se puede ver la máscara que se genera al detectar el color de la pista exterior cómo el predominante en la imagen.

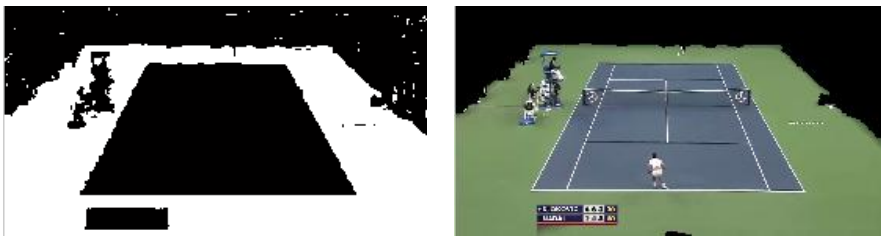


Figura XI. Mascara errónea e imagen de salida del sistema

Para ello en lugar de elegir el valor más repetido cómo referencia del rango a segmentar, se extrae la matriz central de la imagen de dimensión 41x41 pixeles y hallamos el valor medio de la tonalidad (H) de la matriz. Este valor será el valor central del rango de valores que determinará la primera máscara del sistema, lo que hace el sistema más robusto ante los errores acarreados por los planos más alejados de la retransmisión.

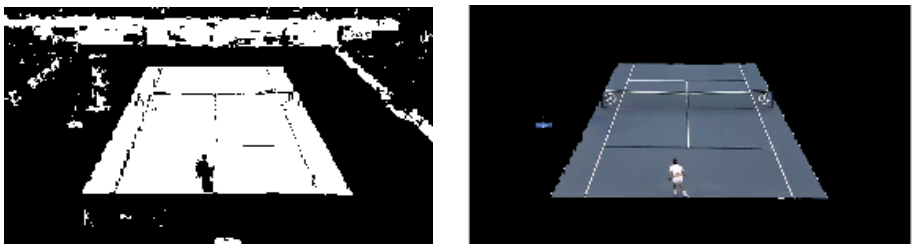


Figura XII. Máscara generada tras incluir la mejora e imagen de salida del sistema

- Cuando aplicábamos el 2° rellenado con la función *imfill* en la máscara negada y añadíamos una línea horizontal de pixeles negros para que no se eliminara nuestro campo al aplicar la operación morfológica, encontramos ciertos casos donde había elementos que coincidían en el espacio con dicha línea y no se eliminaban al aplicar el rellenado. Por lo que se planteó aplicar un doble rellenado a la máscara inversa para conseguir borrar los elementos erróneos de nuestra máscara final.

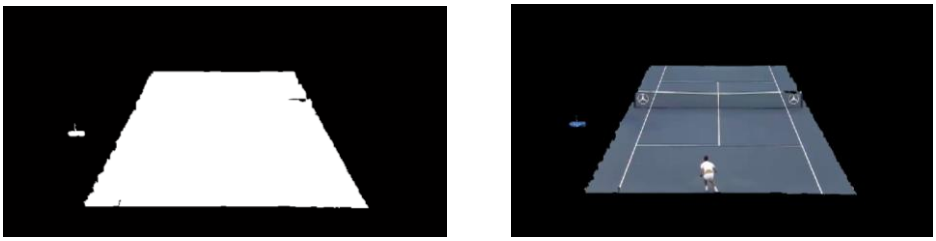


Figura XIII. Ejemplo del error producido al coincidir un elemento externo del campo con la línea añadida

Este método consiste en aplicar el rellenado a la máscara inversa, pero en lugar de añadir la línea que divida la imagen en dos mitades, se suma una línea horizontal de pixeles negros desde el lateral izquierdo al centro de la imagen. Se aplica el rellenado y se consigue una máscara. De nuevo añadimos a la máscara negada original otra línea horizontal que parta del centro de la imagen al borde lateral derecho de la imagen, aplicamos el rellenado y obtenemos otra máscara.

Multiplicamos entre sí las dos máscaras obtenidas, por lo que eliminamos los elementos que solo aparezcan en una de las dos máscaras (incluidas los pixeles no deseados de las líneas añadidas en este método), invertimos la máscara obtenida y obtenemos la máscara definitiva.

Extracción bordes

En la extracción de bordes vamos a partir de dos análisis diferenciados. Por un lado estudiaremos la extracción de bordes aplicado a las máscaras obtenidas en la sección anterior. Y por otro lado estudiaremos las imágenes que no hemos podido extraer la máscara adecuadamente por ser pista monocromática. A priori se espera obtener mejores resultados en la extracción de bordes sobre máscaras obtenidas por la componente de color.

Para la extracción de bordes se utiliza la función *edge* del *Image Processing Toolbar* con la aproximación a la derivada Sobel. Como se muestra en (10) esta función encuentra los bordes de una imagen de distintos niveles de intensidad. El resultado es una imagen binaria del mismo tamaño que la imagen original en la cual, "1" significa que ha detectado un borde y "0" es que no lo ha detectado.

El parámetro *thresh* indica el umbral de binarización. Si se elige el umbral de binarización, hay que ser consciente de que la función edge normaliza la imagen antes de procesarla, llevándola al intervalo [0,1]. También divide las máscaras empleadas en los filtros por un factor, siendo este valor 8 para Sobel.

Después se aplica la transformada de Hough para detectar líneas en nuestra imagen con bordes. La transformada Hough es un algoritmo empleado en reconocimiento de patrones de una imagen, la cual permite encontrar formas como círculos, líneas, entre otras, dentro de la imagen. La versión más sencilla consiste en encontrar líneas, pero de acuerdo a la imagen y problema que se tenga se puede modificar para encontrar otro tipo de formas.

La trasformada Hough utiliza dentro de su funcionamiento una representación paramétricas de forma geométrica, es decir, que si se tiene una recta, esta se representaría con los parámetros r y θ, donde r es la distancia entre a línea y el origen, y θ es el ángulo del vector desde el origen al punto más cercano.

$$r = x\cos\theta + y\sin\theta$$

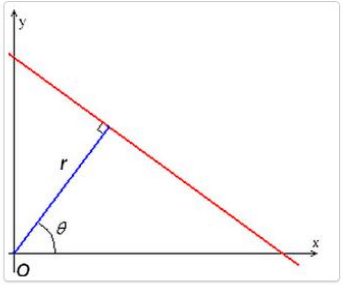


Figura XIV. Parametrización de la recta

Una de las características que posee la trasformada es que si se representan en un plano cartesiano la recta quedaría representada mediante las coordenadas (r ,θ), y el punto se representaría como una función senoidal.

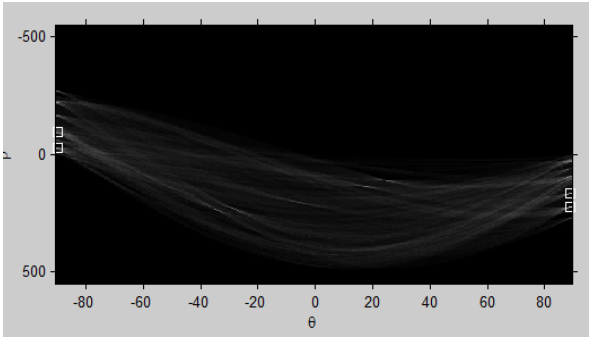


Figura XV. Representación senoidal de la TFH

Extracción de bordes sobre máscara obtenida por la tonalidad dominante en el fotograma

Como se ha dicho anteriormente emplearemos la función Sobel para detectar los bordes de la máscara generada anteriormente. Esta función detecta cualquier irregularidad en los bordes de la máscara, lo que nos supone un pequeño problema, ya que es demasiado precisa y puede que para ciertos casos el uso de la transformada de Hough no sea tan óptimo como se espera. En la figura XVI se muestran el resultado de aplicar en varias máscaras obtenidas la función Sobel.

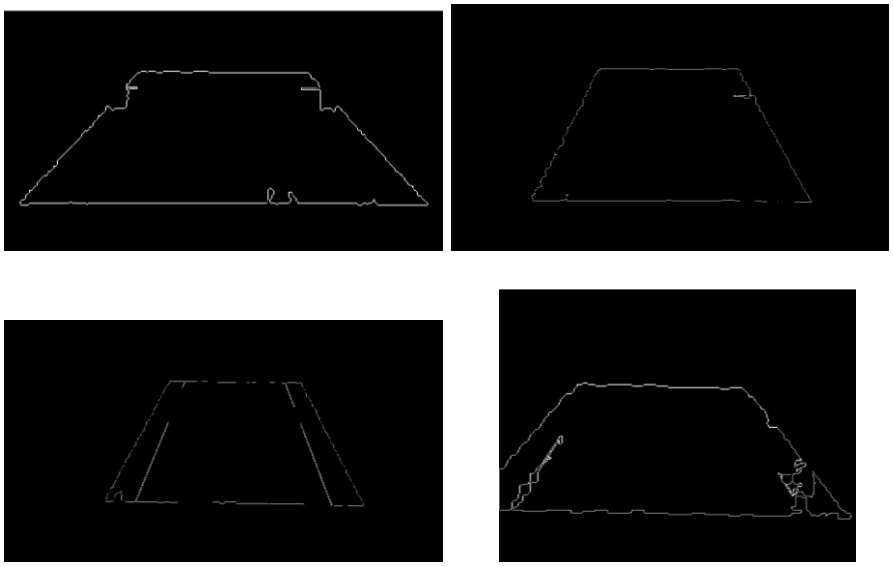


Figura XVI. Ejemplos de la detección de bordes en máscaras obtenidas

Una vez obtenidos las máscaras de salida de la función sobel aplicaremos las técnicas que se detallan en el siguiente apartado para extraer los bordes del campo a segmentar. Posteriormente decidiremos si en verdad mejora el rendimiento del sistema realizar una primera segmentación por medio del color dominante.

Extracción de bordes sobre fotogramas

Como se comentó en capítulos anteriores, cuando nuestra imagen de entrada al sistema se trata de una pista con superficie monocroma que no presenta un cambio brusco en la tonalidad de la pista, no resulta nada útil obtener la máscara por detección del color predominante en la imagen. Es por esto que cuando se presente este caso, el sistema omitirá este paso y comenzará el análisis directamente con la detección de bordes.

Lo primero operación que haremos con el fotograma una vez extraído y transformado a tipo "double" es pasar la imagen a escala de grises para poder detectar los bordes.

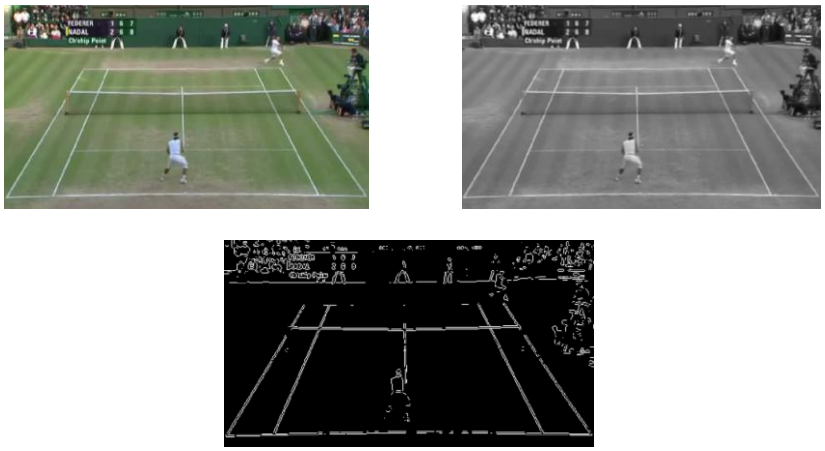


Figura XVII. Fotograma original, fotograma en escala de grises y fotograma tras pasar por el filtro Sobel.

Observamos que la función es bastante eficiente detectando los bordes de la pista, aunque también identifica otros elementos que pueden llevar el sistema a errores.

Una vez se obtenga la imagen analizada con la función Sobel, vamos a dividir la detección de bordes en cuatro partes diferenciadas. Comenzaremos intentando identificar el borde inferior de la pista, después el borde lateral derecho y el borde lateral izquierdo, finalizando con el borde superior que se presupone el más complicado de identificar. Este reparto en la detección de bordes, conllevará un aumento en el tiempo de cómputo y de procesado, ya que aplicaremos varias veces similares procesados así como la transformada de Hough, pero conseguiremos incrementar la probabilidad de detectar con éxito los límites de la pista.

Detección borde inferior

Para facilitar la detección del borde inferior, antes de aplicar la transformada de Hough, ponemos todos los píxeles de la mitad superior de la imagen a 0. Con esto evitaremos que se creen conflictos a la hora de decidir cuál es el borde inferior de la pista, ya que eliminamos la opción de detectar líneas generadas por vallas publicitarias en la parte superior de la pista o la propia red. Además en el plano general más usado en retransmisiones de partidos de tenis el borde inferior de la pista suele presentarse en la mitad inferior de la pantalla. En la figura XVIII podemos ver un ejemplo de la mejora que se produce al realizar la operación de negar la mitad superior de la imagen, y de no ser así la red podría considerarse candidata a identificarse cómo borde inferior.

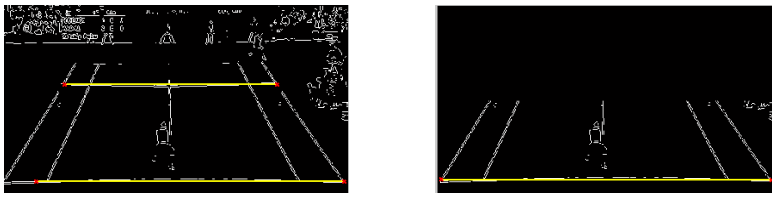


Figura XVIII. Ejemplo de detección de borde inferior

Aquí es importante decidir que parámetros debemos elegir para la función *houghpeaks* cómo para la función *houghlines,* ya que unos parámetros adecuados nos permitirá afinar mucho más en la detección del borde inferior. Para este caso y tras las pruebas realizadas, se establece para *houghpeaks* un máximo de tres picos y un umbral de 0,5 · max(H), mientras que para *houghlines* el "Fillgap" se establece en 50 píxeles y para "Minlength" en 200 píxeles.

Después de aplicar la transformada de Hough, decidiremos que el borde inferior de la pista se corresponderá con el segmento más largo entre los candidatos obtenidos, ya que para las pruebas realizadas este coincidía en un alto porcentaje de casos con el borde deseado.

Una vez se haya detectado el borde inferior, se calcula la pendiente de la recta y se calcula una línea con la misma pendiente que la recta hallada, pero con comienzo y final en los límites de la imagen, para posteriormente crear una máscara en la que todos los pixeles debajo de la línea sean cero. Esto nos permitirá eliminar los pixeles que aparecen por debajo del borde inferior producto de aplicar la función Sobel, por ejemplo fruto de posibles rótulos que se encuentren en la pista. Está máscara la generamos con la función *poly2mask*, que nos permite a partir de unos puntos dados generar un polígono con vértices en dichos puntos. Los cuatro vértices del polígono serán los dos pixeles de los dos vértices inferiores de la imagen, y los dos puntos donde corta la línea calculada con los límites laterales de la imagen, a los que les sumaremos un pixel en la dirección del eje "y" para no ocultar las líneas cuando se superponga la

máscara. Cómo podemos ver en (8), si *y* es una función lineal de *x*, entonces el coeficiente de *x* es la pendiente de la recta. Por lo tanto, si la ecuación está dada de la siguiente manera

$$y = mx + b$$

m es la pendiente. En esta ecuación, el valor de b puede ser interpretado como el punto donde la recta se interseca con el eje Y, es decir, el valor de y cuando $x = 0$. Este valor también es llamado coordenada de origen.

Si la pendiente *m* de una recta y el punto (x0, y0) de la recta son conocidos, entonces la ecuación de la recta puede ser encontrada usando:

$$y - y_0 = m(x - x_0)$$

Figura XIX. Ejemplo de fotograma con la máscara del borde inferior

Detección borde lateral derecho

Hallada la máscara para el borde inferior, el siguiente paso consiste en detectar el borde lateral derecho. Similar de como se ha hecho para detectar el borde inferior, antes de aplicar la transformada de Hough, ponemos todos los píxeles de la mitad izquierda de la imagen a 0. Aquí el principal problema que se nos va a presentar es el de indicar al sistema que línea decidir cuál es el límite de la pista, ya que el patrón de escoger el segmento más largo puede no ser tan bueno como para el borde inferior. Con este criterio el sistema podrá detectar tanto la línea que delimita el campo de juego en partidos de dobles como la que delimita la pista en partidos individuales. Esto nos ofrece la posibilidad de poder segmentar la pista dependiendo del tipo de partido que se esté disputando.

Por tanto se ha elegido cómo parámetros para la función *houghpeaks* un máximo de tres picos y un umbral de 0,5 · max(H), mientras que para *houghlines* el "Fillgap" se establece en 50 píxeles y para "Minlength" en 200 píxeles, similar a los valores utilizados para el borde inferior.

Una vez se ha aplicado la transformada de Hough, se debe decidir qué línea debe ser la que marque el límite del campo dependiendo del tipo de partido que se esté disputando. Por tanto si el partido es de dobles, el sistema deberá elegir la línea más exterior hacia el borde derecho de la imagen. Sin embargo si el partido es de individuales, el sistema establecerá el borde del campo en la línea más interna de la imagen. Finalmente este tipo de decisor no se incluyó en la versión final, ya que se primó reducir el tiempo de cómputo de la aplicación, aunque se considera una buena utilidad si se pusiera en práctica en un sistema y escenario real.

Una vez se haya elegido la línea adecuada, similarmente que en el apartado anterior se calcula la pendiente de la recta y se calcula los puntos de la recta para crear una máscara con los píxeles que quedan a la derecha de dicha línea. Esto nos permitirá eliminar los pixeles que se encuentren a la derecha del borde lateral. La máscara se genera también con la función *poly2mask*.

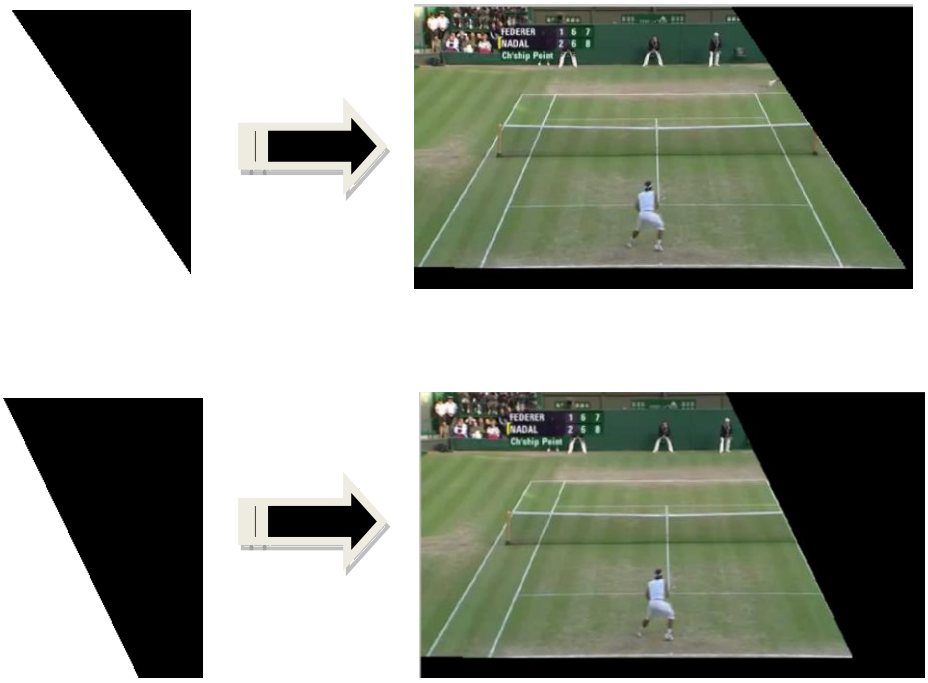


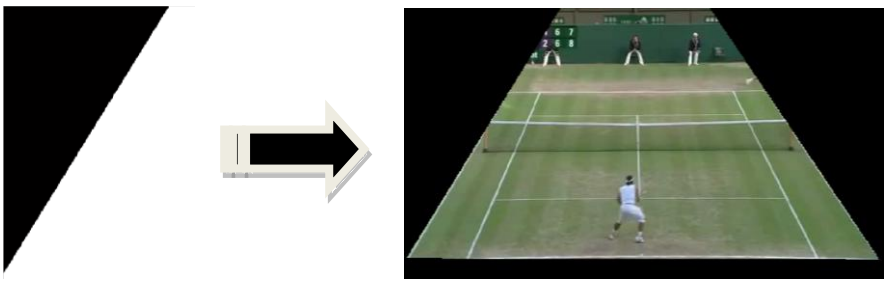
Figura XX. Ejemplo de las máscaras generadas por la detección de borde lateral derecho. 1. Muestra de máscara en partido de dobles. 2. Muestra de máscara en partido de individuales

Detección del borde lateral izquierdo

Hallada la máscara para el borde lateral derecho, seguiremos los mismos pasos para detectar el borde lateral izquierdo pero variando los parámetros para el lado opuesto. En este caso se ponen todos los píxeles de la mitad derecha de la imagen a 0.

Los parámetros para la función *houghpeaks* y *houghlines* serán similares a los valores utilizados para el borde opuesto.

Para este caso, cuando se decida la línea adecuada, se calcula la pendiente de la recta y los puntos de la recta para formar la máscara con la función *poly2mask* de los píxeles que quedan a la izquierda de dicha línea.



Detección del borde superior

La detección del borde superior es con diferencia el caso más complejo de los cuatro distintos tipos de bordes a identificar. Las razones de la dificultad en la detección de esta línea del campo pueden atribuirse a que por la perspectiva de la imagen es el segmento más corto de los que debemos identificar. Además, debido a que es la línea más alejada desde el punto en que se filman las imágenes, hace que además el grosor de la línea en la imagen sea menor, y por tanto, ocupe menos pixeles en el fotograma. Este problema se acentúa más en las imágenes con definición estándar (SD) respecto a las secuencias en alta definición (HD), ya que las imágenes se representan en menos pixeles y dificulta la identificación de las líneas del campo. Además por perspectiva la red aparece cerca del borde superior y tiende a poder confundir al sistema.

Antes de aplicar la transformada de Hough, como en los apartados anteriores, hemos puesto a cero los pixeles de la mitad opuesta de la pantalla donde presumiblemente se va a encontrar el borde a identificar. Tras realizar varias pruebas y simulaciones hemos comprobado que si al sistema se le indica los criterios de decisión implementados hasta ahora la detección del borde superior no es eficiente.

Indicar al sistema que elija el segmento más largo no resulta útil, porque en la mayoría de casos por la perspectiva de la filmación, la red se presenta como una línea más larga que el borde superior. Además tampoco se le puede ordenar que elija el segmento más cercano al borde superior del fotograma, porque puede confundirse con líneas generadas por vallas publicitarias o el propio público.

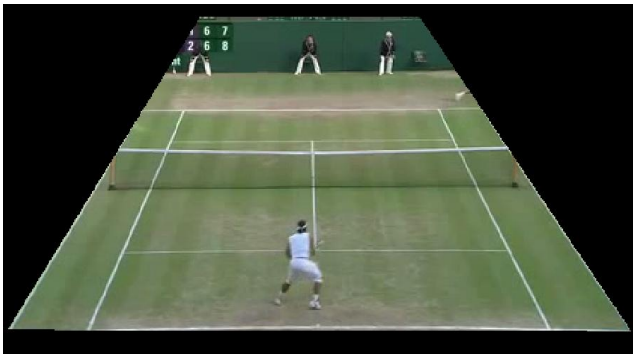


Figura XXI. Ejemplo de detección incorrecta del borde superior

Analizando las diferentes secuencias hemos obtenido un método que nos aumenta la probabilidad de acierto considerablemente. Para ello, obtendremos la imagen original con la máscara obtenida tras la detección de los bordes anteriores (Figura XXI). Reducimos los pixeles sobre los que actuará la transformada de Hough, intentando evitar que se detecte la red. Para ello aumentamos las filas que ponemos a cero en nuestra imagen.

En lugar desde el centro de la imagen hasta el borde inferior de la imagen, se ponen a cero las 3/5 partes inferiores de las filas totales de la imagen. Con esto evitamos detectar la red en nuestra transformada para prácticamente la totalidad de las secuencias analizadas.

Ya solo tenemos que elegir los parámetros correctos para las funciones *houghlines y houghpeaks* e indicar al sistema que elija el segmento más largo de la imagen como el borde superior de la pista. Hemos determinado para la función *houghpeaks* los mismos valores empleados en el borde inferior, sin embargo para la función *houghlines* establecemos estos valores en 100 pixeles tanto para el *Fillgap* como para *Minlength.*

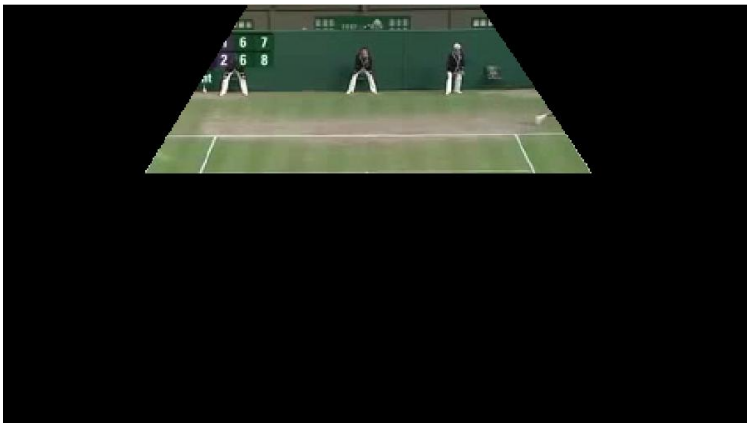


Figura XXII Muestra de la zona de imagen para determinar el borde superior

Una vez obtenida la posición del borde superior, operamos como en los casos anteriores. Detectamos la pendiente de la recta, calculamos los puntos de la misma y se crea la máscara para el borde superior, que añadida a la máscara de los bordes anteriores, formaran la máscara definitiva de nuestro sistema.

Figura XXIII. Máscara definitiva de la extracción de bordes

Mejoras

- Después de varias pruebas se ha detectado un error que se produce en varias imágenes al detectar el borde lateral. En varias ocasiones se detectaban las líneas que limitan el campo lateralmente tanto para partidos de dobles como para partidos individuales, pero también se identificaba el borde inferior, lo que nos introducía errores en nuestro modelo de decisión. Inicialmente se redujo el número de picos máximos de la función *houghpeaks* de tres a dos, pero esto solo solucionaba el problema en algunos casos. Por tanto, se añadió la máscara obtenida de detectar el borde inferior a la imagen antes de aplicar la transformada de Hough, con lo que se minimiza los errores producidos por detectar el borde inferior(ver figura CV).

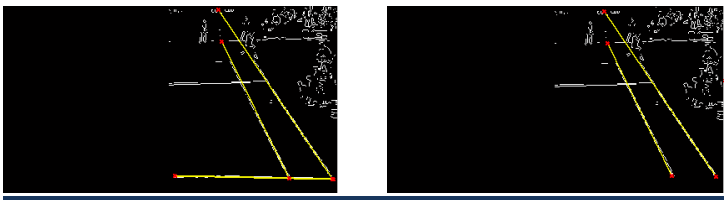


Figura XXIV. Muestra de líneas detectadas antes y después de implementar la mejora

CONCLUSIONES Y FUTUROS DESARROLLOS

Conclusiones

Una vez finalizado el desarrollo de nuestro sistema debemos sacar conclusiones respecto a la eficiencia del mismo. Hemos realizado los test con nuestra base de videos de test. Hemos realizado el test para secuencias de 1 segundo, por lo que dependiendo de los fotogramas por segundo de cada video, se habrán analizado más o menos fotogramas. Se ha establecido que en las secuencias de 1 segundo, se debería de obtener como mínimo un 80% de acierto en la detección de bordes para considerar que se ha podido detectar correctamente. Por lo que si la secuencia va a 50 fotogramas por segundo, se necesitaría detectar el borde en al menos 40 fotogramas.

Los resultados se definen en las siguientes tablas:

En esta tabla mostramos el número de video con el partido correspondiente.

Video	Jugadores y pista
Video 1	Djokovic - Nadal - Australia
Video 2	Tipsarevic - Del Potro - Delray Beach
Video 3	Federer - Verdasco - Us open (Nueva York)
Video 4	Nadal vs Murray - Japón
Video 5	Nadal vs Djokovic - Madrid
Video 6	Federer vs Nadal – Wimbledon (Londres)
Video 7	Federer vs Roddick - Melbourne
Video 8	Djokovic - Nadal - Rolang Garros

En la siguiente tabla se muestra el número de video junto a si se le ha realizado la extracción del color dominante. Además se indica para qué videos se ha identificado correctamente cada uno de los bordes y finalmente el porcentaje de acierto para cada uno de esos bordes.

Video	Extracción de color dominante	Detección borde inferior	Detección borde lateral derecho	Detección borde lateral izquierdo	Detección borde superior
Video 1	SI	SI	SI	SI	SI
Video 2	NO	SI	NO	NO	NO
Video 3	SI	SI	SI	SI	SI
Video 4	SI	SI	SI	SI	SI
Video 5	NO	SI	SI	NO	NO
Video 6	NO	SI	SI	SI	SI
Video 7	NO	SI	SI	SI	NO
Video 8	NO	SI	SI	SI	SI
Media		100%	87.5%	75%	62.5%

De los resultados obtenidos en la tabla podemos afirmar que se ha obtenido un 62,5% de éxito en la segmentación del campo completa.

Aunque se observa que el hecho de que al video se le haya tratado anteriormente para la extracción del color dominante, nos proporciona óptimos resultados. Ya que para el 100% de las secuencias tratadas se han detectado los cuatro bordes.

Además, independientemente de que el video haya sido tratado o no, el borde inferior también se ha detectado satisfactoriamente en todos los videos de nuestra base de test, a diferencia de lo que ocurría con nuestra base de entrenamiento donde esta cifra no era del 100 %.

Por tanto los resultados pueden considerarse aceptables a tenor de los medios disponibles, aunque con una base de entrenamiento más amplia estos porcentajes podrían ser más altos.

A continuación se muestra la tabla con el tiempo de cómputo de cada secuencia, lo que nos podrá indicar si el sistema podría emplearse en un escenario real. En esta tabla se distingue también si el video ha sido monitorizado o no, es decir, si se han mostrado en pantalla las imágenes de la secuencia cada vez que se analizaba un fotograma.

Video	Extracción de color dominante	Tiempo monitorizado (seg)	Tiempo no monitorizado (seg)
Video 1	SI	8,3	2,2
Video 2	NO	9,7	2,8
Video 3	SI	12,7	4,3
Video 4	SI	11,3	4,2
Video 5	NO	7,8	2,5
Video 6	NO	5,6	2,2
Video 7	NO	4,4	1,9
Video 8	NO	6,9	3,2
Promedio		8,3375	2,9125

Observando los tiempos de cómputo de nuestro sistema, podemos decir que los resultados no son tan buenos como cabría esperar. Por un lado los tiempos obtenidos incluyendo el monitoreado son demasiado largos cómo para plantear ejecutarlo en un sistema de análisis en tiempo real. Sin embargo el tiempo de cómputo sin incluir el monitoreado desciende bastante. Lo que indica que nuestro software aplicado a un sistema real debería separar la ejecución del tratamiento de imágenes respecto al monitoreado, ejecutándose ambos en paralelo para reducir el tiempo de cómputo.

Por otro lado, la cámara con la que se filmaron las secuencias utilizadas permanece prácticamente fija. Únicamente se detectan ligeros movimientos horizontales o verticales de la cámara, y un leve uso aislado del zoom. La sensación que percibe el espectador es de que se trata de una cámara fija, de hecho en ciertas secuencias así es. Por lo que, aunque nuestro sistema no haya obtenido resultados óptimos en lo que a tiempo de cómputo se refiere, la diferencia de movimiento en la posición del campo de un fotograma al siguiente puede tender a cero o ser cero. Esto nos hace pensar que el número de fotogramas por segundo que entran en nuestro sistema podría verse reducido sin alterar los resultados finales. O lo que es lo mismo, podríamos descender el número de fotogramas a analizar, lo que reduciría el tiempo de cómputo sin afectar a la eficiencia de nuestro software.

Además se debe tener en cuenta que ante un escenario real se dispondría de un hardware mucho más potente que descendería considerablemente los tiempos de cómputo, acercándolo a valores de eficiencia necesaria para trabajar con las secuencias en tiempo real.

Podemos concluir argumentando que los resultados obtenidos respecto a la segmentación de los bordes de la pista no son tan óptimos como se esperaban, pero sí esperanzadores. Ya que en un escenario real, con una base de entrenamiento mucho más amplia para poder detectar los errores más habituales y equipos más eficientes y potentes, la tasa de acierto aumentaría con toda seguridad.

Por tanto, en un escenario real, podría emplearse para realizar una segmentación de la pista de los fotogramas en tiempo real. Además de poder emplearse como base para la detección de eventos de alto nivel, tales como el movimiento de los jugadores y el seguimiento de la trayectoria de la pelota en cada golpe, tal y como mostraban las diferentes técnicas y métodos estudiados en el momento de comenzar este trabajo.

Futuros desarrollos

Una de las futuras líneas de desarrollo sugeridas podría ser la de implementar un sistema de análisis tanto espacial y temporal entre los distintos fotogramas para reducir el tiempo y el coste computacional del sistema implementado. Para ello se plantea analizar la relación entre los fotogramas vecinos, empleando las técnicas de vector de movimiento y desplazamiento temporal.

Además también debido a no ver aumentado en demasía la carga computacional del sistema no se desarrolló un método automático de clasificación del tipo de video que entraba en la aplicación.

Por ello se propone el desarrollo de un método de decisión de qué tipo de campo se va a analizar, ya sea de un solo color o tenga varios colores. Para ello se puede analizar la información del histograma e implementar un método que permita al sistema de forma autónoma clasificar si se trata de una señal de entrada con una pista monocroma o policroma.

El sistema que hemos desarrollado confiamos en que podría emplearse para analizar secuencias de videos deportivos para otros deportes en los que se muestren planos generales de la pista o el campo de juego. Con ligeros ajustes y adaptándolo al tipo de señal de entrada podría resultar eficiente para analizar secuencias de video en deportes tales cómo voleibol, tenis de mesa, o baloncesto.

Además muchas de los conceptos e ideas planteadas podrían servir de guía para futuros desarrollos en videos deportivos en los no se muestra el campo integro en la imagen.

Ya que podrían ser de bastante utilidad los métodos empleados en nuestro sistema tanto para la detección de bordes como para la extracción del color dominante del campo de juego, en retransmisiones de deportes tales como el futbol o el balonmano.

ANEXOS

Presupuesto

El trabajo fin de grado realizado se ha presupuestado teniendo en cuenta el tiempo dedicado por el tutor y por el alumno, además del equipo empleado para la búsqueda de información, el desarrollo y la elaboración del TFG.

Se estima que el tiempo dedicado ha sido de unas 20 horas/semana durante un periodo de 4 meses. El tiempo empleado por persona/mes se fija en unas 80 horas.

Por tanto el presupuesto se ha estimado para un tiempo total de 344 horas de trabajo por parte del personal implicado.

En la tabla se muestra el desglose total sin incluir los impuestos indirectos.

Coste personal				
Categoría	**Nombre**	**Tiempo dedicado (mes)**	**Coste (persona/mes) (€)**	**Coste (€)**
Ingeniero Sénior	Jesús Cid Sueiro	0,3	3.663 €	**1.099 €**
Ingeniero	Alberto Sánchez González	4	2.147 €	**8.588 €**
Coste material				
Equipo empleado	**Coste equipo**	**Tiempo dedicado (mes)**	**Periodo Amortización (mes)**	**Coste (€)**
Ordenador portatil	900 €	3,5	72 €	**44 €**

En la siguiente tabla se muestra el desglose total incluyendo impuestos.

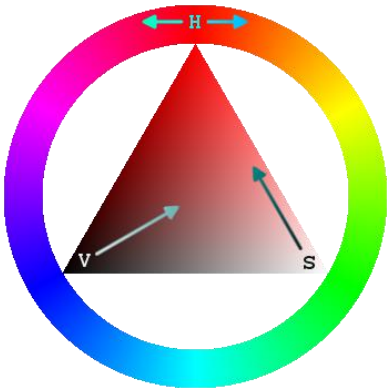
Coste personal	9.687 €
Coste equipos	44 €
Costes indirectos	2.044 €
Coste total	**11.775 €**

El presupuesto total del proyecto es de **11.775 €**

El modelo de color HSV

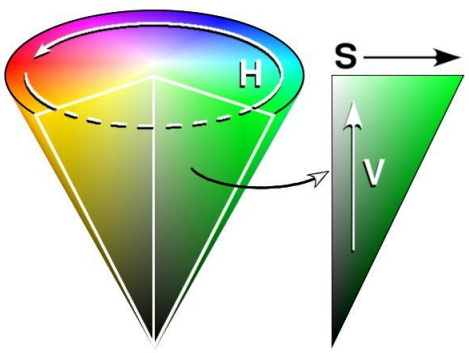
El modelo de color HSV es una transformación no lineal del espacio de color RGB. Esta transformación puede resultar muy útil cuando se desea escoger un color determinado.

Es un espacio cilíndrico, pero normalmente se le asocia a un cono, debido a que es un subconjunto visible del espacio original con valores válidos de RGB. El color se compone por medio de tres parámetros distintos.

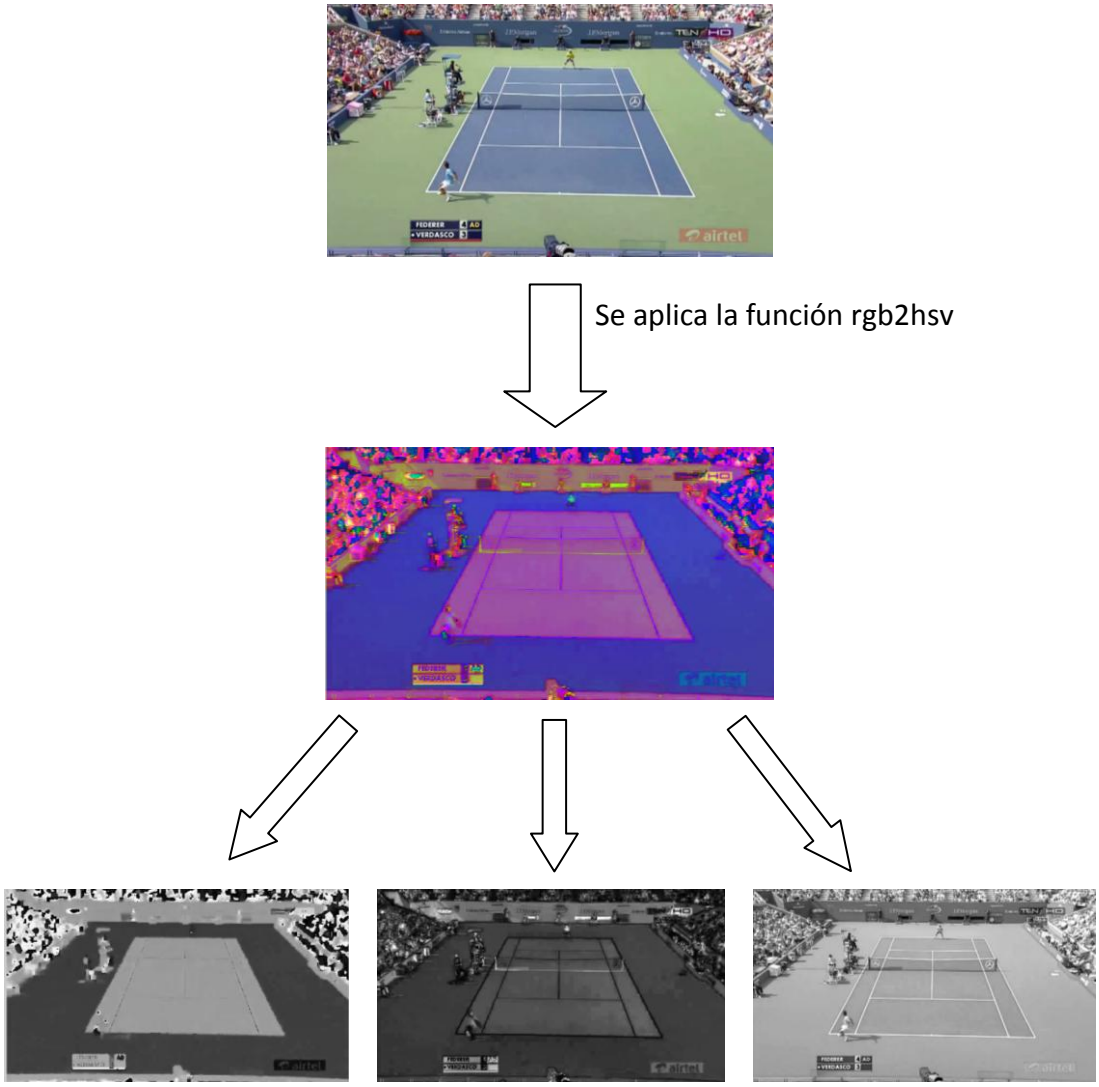


Tonalidad (Hue): Se refiere a la frecuencia dominante del color dentro del espectro visible. Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100%). Cada valor corresponde a un color e incrementa su valor mientras nos movemos de forma antihoraria en el cono. El rojo es encuentra en el angulo 0º, el verde en el 120º y el azul en el 240º.

Saturación (Saturation): Se refiere a la cantidad del color o a la distancia al eje de brillo negro-blanco. A este parámetro también se le suele llamar "pureza" por la analogía con la pureza de excitación y la pureza colorimétrica de la colorimetría. Los valores posibles van del 0 al 100%. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea tendrá y más decolorado se mostrará.



Valor (Value): Representa la altura en el eje blanco-negro., dicho de otra manera, es la intensidad de luz de un color o la cantidad de blanco o de negro que posee un color. Los posibles valores varían del 0 a 100. Para el color blanco se puede poner cualquier color y saturación, siempre que se establezca el valor (de luminosidad) máximo. Asimismo, para el color negro se puede poner cualquier color y saturación, siempre que se ponga el valor 0.



BIBLIOGRAFÍA

(1) http://red.pucp.edu.pe/rpo/pdfs/Curs_Farcy.pdf

(2) Tomas Svoboda, Jan Kybic and Vaclav Hlavac; *Image Processing, Analisys and Machine Vision.* International student edition, 2008.

(3) http://www.hawkeyeinnovations.co.uk/

(4) http://es.wikipedia.org/wiki/Ojo_de_halcon

(5) I. Kolonias, J. Kittler, W J Christmas, F Yan; *Improving the accuracy of automatic tennis video annotation by high level grammar*

(6) http://es.wikipedia.org

(7) es.wikipedia.org/wiki/Modelo_de_color_HSV

(8) http://sobrecolores.blogspot.com.es/2010/12/espacio-de-color-hsv.html

(9) http://es.wikipedia.org/wiki/Pendiente_recta

(10) http://arantxa.ii.uam.es/~jbescos/Docencia/TAPS/Practicas/TAPS09_10_Guion_P1.pdf

(11) http://www4.ujaen.es/~satorres/practicas/practica3_vc.pdf

(12) Guangyu Zhu, Changsheng Xu, Oingming Huang, Wen Gao and Liyuan Xing. *Proceeding Multimedia '06 Proceedings of the 14th anual ACM international conference on Multimedia. Pages 431-440*

(13) Chi-Kao Chang, Min-Yuan Fang, Chung-Ming Kuo, Nai-Chung Yang. Event Detection for Broadcast Tennis Videos Based on Trajectory Analysis

(14) A. Ekin and A.M.Tekalp. Robust dominant color region detection and color-based applications fos sport video. *International Conference on Image Processing*

(15) Jiang, Ye, Gao and Huang. A new method to segment playfield and its applications in match analysis in sports video. *Proceedings of the 12th annual ACM international conference of multimedia.*

(16) es.wikipedia.org/pista_de_tenis

(17) http://es.wikipedia.org/wiki/Transformada_de_Hough

(18) http://procesamientodigitalimagenes.wordpress.com/2012/11/02/transformada-hough/