UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

IDENTIFICACIÓN DE VIDEO TRANSMITIDO A TRAVÉS DE TELEVISIÓN DIGITAL USANDO UNA COMPACTA FIRMA BINARIA

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
ONOFRE ALVA MANUEL ALEJANDRO

TUTOR:
DR. NICOLÁS C. KEMPER VALVERDE
CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLÓGICO

MÉXICO, CDMX, FEBRERO 2017

Agradecimientos

Quisiera agradecer a mi madre Julieta Alva Olvera, a mi padre Alejandro Onofre Sánchez y a mi hermano Reymundo Antonio Onofre Alva por su apoyo incondicional durante todo este tiempo. Sin ustedes habría sido infinitamente más difícil llegar hasta aquí.

A la Universidad Nacional Autónoma de México (UNAM) por permitirme estar dentro de sus aulas una vez más.

Y al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico durante mis estudios de maestría.

Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea el resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

MANUEL ALEJANDRO ONOFRE ALVA. MÉXICO, CDMX, FEBRERO 2017

Resumen

El siguiente trabajo describe una metodología para la detección y reconocimiento de comerciales televisados a través de una señal digital. Debido a que trabajar directamente con los archivos de video implica un costo computacional importante es deseable obtener de ellos algún tipo de descriptor o descriptores que nos permitan reducir el costo computacional de buscar un fragmento de video o video clip en un archivo de video más grande o en una base de datos. La investigación está centrada hacia la búsqueda de una estructura compacta y relativamente simple, que describa el contenido del video de una forma eficiente. Teniendo en cuenta que un archivo de video es una sucesión de imágenes en el tiempo, el enfoque es encontrar primero una metodología que describa una imagen utilizando un tipo de estructura como la deseada y después buscar entre los procesos de codificación de los diferentes contenedores de video al contenedor que mejor encaje en el esquema seleccionado (MP4, MPG, FLV, AVI, etc.), es decir encontrar el formato adecuado para pasar la estructura compacta que describa imágenes a una estructura compacta descriptora de videos. Los resultados fueron satisfactorios, la combinación de firmas binarias para describir imágenes y la contenedores MPEG dan como resultado un descriptor de tamaño razonablemente pequeño que puede reducir el tiempo de procesamiento 12 veces (comparado con el uso de otros contenedores de video) para conocer el número de apariciones de un video clip, en la transmisión de un día completo en alguna televisora.

Índice de Figuras

Índice de Tablas

Índice general

Capítulo 1

Introducción

La publicidad es, notoriamente, un amplio campo para las diversas industrias e individuos que buscan dar a conocer una marca, imagen o producto. Si bien es cierto que la televisión no es el único medio de comunicación masiva, hoy en día junto con la publicidad por Internet y radio, es un punto de interés entre la población mundial, tanto para mantenerse informados de los sucesos nacionales e internacionales como para diversas frivolidades que en ella encuentran un espacio para llegar a los hogares de miles de millones de personas.

El siglo XXI y la llamada *era digital* trajeron consigo el *apagón analógico* a diversas partes de México y del mundo. Este no es un cambio que afecte directamente el contenido de las cadenas televisoras y los medios de comunicación, sino la calidad y el ancho de banda con la se transmiten los diversos contenidos desde las casas productoras hacia las casas de los televidentes y radio escuchas. Al ocurrir el *apagón analógico* una gran parte del ancho de banda que se utilizaba para transmitir la señal de televisión analógica quedó libre y con su liberación se abre un abanico de posibilidades para empezar a utilizar el ahora *espacio blanco*, que tienen que ver en su mayoría con el acceso a Internet para las comunidades, principalmente rurales, que aún no lo tienen y una mejor calidad en el servicio de quienes ya cuentan con este derecho.

Sin lugar a dudas, aún después del *apagón analógico*, la televisión es un punto de encuentro para las distintas marcas y sus clientes potenciales. Especialmente la televisión abierta es una mina de oro en oportunidades de comerciales y anuncios publicitarios de todo tipo, ahora con calidad digital. Empresas e instituciones públicas y privadas compiten por los horarios con mayor número de televidentes, en los cuales sus anuncios y comerciales pueden tener mayor impacto y penetración a su población objetivo y sus consumidores potenciales. Por lo cual, conocer si un espacio publicitario se ocupa en tiempo y forma para lo que se ha acordado es de gran interés para quien pagó por el transmitir el comercial, sus competidores, quienes realizan monitoreo de mercado junto con estadísticas y por su puesto para quien lo transmite.

1.1. Problemática

Se han realizado diversos trabajos e investigación en la tarea no trivial de reconocer pautas publicitarias o comerciales en televisión [1], [2], [3], [4]. De manera superficial podemos dividir el problema en tres subproblemas:

- Obtener la señal de video digital transmitida por la cadena televisora.
- Separar comerciales de programación regular en la señal de video obtenida.
- Ser capaces de identificar las apariciones de comerciales de interés.

El primer subproblema, obtener un archivo de video de la señal digital transmitida por la televisora, puede ser atacado de diferentes maneras. Siendo que existe la IPTV podríamos obtener el *streaming* de la cadena televisora, si es que está disponible y se cuenta con el ancho de banda adecuado. Otra forma de resolver este punto seria con un sintonizador de televisión digital con puerto USB o bien una tarjeta sintonizadora de televisión digital dedicada. Ambos compatibles con el estándar ATSC A/53 y el sistema de compresión MPEG-2 y H.264, debido a que son los que se dictaminaron usar en México para la Televisión Digital Terrestre (TDT) por el Comité Consultivo de Tecnologías Digitales para la Radiodifusión (CCTDR) [5].

El segundo subproblema es, por sí mismo, parte de otro campo de investigación que consiste en separar un archivo de video compuesto por n segmentos de diferentes videos en los n archivos de videos correspondientes. En nuestro caso obviaremos este problema, ya que la TDT debería proveer el etiquetado necesario para hacer de este un problema trivial.

Finalmente el tercer subproblema, el cual es identificar pautas publicitarias o comerciales de TDT, puede ser visto como el tema para el que está pensada esta tesis.

1.2. Objetivo

El principal objetivo de este trabajo es proponer un método para identificar pautas publicitarias dentro de las 24 horas de transmisión digital, o por IPTV, de una televisora cualquiera cuya señal pueda ser guardada en un archivo de video MPEG-X.

1.3. Estructura de la tesis

El contenido de la presente tesis está organizado para presentar las investigaciones realizadas sobre la detección y reconocimiento de pautas publicitarias utilizando diferentes metodologías existentes, comparando los resultados obtenidos. El contenido se divide en los siguientes capítulos.

Capítulo 2 – Marco Teórico

En este capítulo se presenta una revisión de la literatura existente sobre el reconocimiento de video clips, organizada en 4 secciones: 2.1 Comparación basada en Pixeles, 2.2 Histogramas de Color Global (GCH), 2.3 Histogramas de Color usando Vectores de Color Coherente (CCV) y 2.4 Firmas Binarias usando CBA y VBA.

Capítulo 3 –Reconocimiento de Pautas Publicitarias

El contenido de este capítulo es el cuerpo fundamental de la tesis, aquí se describe la metodología seguida, así como la implementación de la misma.

Capítulo 4 – Pruebas y Resultados

En este capítulo se presentan la configuración de las pruebas realizadas, así como los resultados y comparaciones.

Capítulo 5 – Conclusiones y Trabajo a Futuro

En éste último capítulo se presentan las conclusiones obtenidas con base a la comparación de los resultados de las pruebas realizadas y finalmente se propone el trabajo a futuro.

Capítulo 2

Marco Teórico

En las últimas décadas se ha vuelto evidente que la cantidad de archivos digitales que se manejan tanto en la industria como en el hogar ha crecido de manera exponencial, en particular los tipos de archivos que se usan para representar imágenes y videos. Ya que estos ocupan un porcentaje significativo del tráfico de datos que ocurre en el día a día, así como en los espacios de almacenamiento que van desde unidades locales para usuarios finales, hasta bodegas repletas de racks con discos duros en constante cambio para las grandes empresas que se dedican a prestar el servicio de almacenamiento masivo o Big Data. Todo esto es, en parte, una consecuencia del avance constante en tecnologías de almacenamiento, lo cual provoca que las memorias y unidades de almacenamiento dedicadas se vuelvan más accesibles, es decir, que ya no se economiza el espacio para almacenar la información y en lugar de pasar por un arduo proceso selectivo, revisando y separando lo que es útil de lo que no lo es, simplemente se almacena todo. Tanto las empresas como el usuario final prefieren simplemente adquirir más espacio de almacenamiento para la información nueva que borrar la vieja y arriesgarse a perder algo que quizá necesiten en el futuro.

Este enorme crecimiento en el tráfico de imágenes y videos ha incrementado la demanda de algoritmos y métodos de búsqueda cuyo fin es encontrar imágenes similares en una base de datos de gran tamaño, de manera rápida y eficiente. Una estrategia popular para la búsqueda de imágenes en una base de datos es la denominada *Consultas usando Ejemplo*s, más conocido por su denominación en inglés *Query By Example* (QBE) [6], en la cual la consulta es expresada como una plantilla de la imagen. Esto se utiliza comúnmente para plantear consultas en la mayoría de los *sistemas para la recuperación de imágenes basados en contenido*, CBIR por las siglas de su nombre en inglés *Content-Based Image Retrieval*.

Típicamente, un sistema CBIR extrae características visuales de una imagen de búsqueda dada, que luego se utilizan para comparar con las características de otras imágenes almacenadas en la base de datos. La función de similitud se basa por lo tanto en el contenido extraído de la imagen, en lugar de basarse en la propia imagen. Hay que señalar que, dado el creciente volumen de imágenes disponible, el enfoque de confiar en las anotaciones humanas como medio de abstracción y reconocimiento de características no es factible, debido a la dificultad de expresar en palabras cualidades gráficas y sensaciones estéticas en la percepción de una imagen. El enfoque más

acertado es usar las características que una computadora puede medir, como la intensidad de color, el brillo, el contraste, el porcentaje de tal color, etc.

2.1. Comparación basada en Pixeles

Este es el método más simple en el que se puede pensar, intuitivamente podemos decir que funciona, pero también intuitivamente sabemos que sufrirá en comparación con otros métodos. Su ejecución consiste en calcular la diferencia en el color de cada píxel entre las imágenes que se están comparando. Para trabajar con videos se requiere tratar a los frames de los vídeos como una secuencia de imágenes que se desea comparar. Cabe mencionar que dichas imágenes deben tener las mismas dimensiones para que el algoritmo tenga sentido, en caso contrario se podría efectuar una operación de escalamiento en alguna de las imágenes a comparar, pero esto podría provocar resultados no deseados, siendo esto uno de los puntos débiles de este enfoque. Una recomendación en la implementación de este algoritmo es normalizar la función de distancia entre las dos imágenes dividiéndola por el número de píxeles W × H (ancho x alto) y los canales de color [7]:

$$Distancia(I_1, I_2, W, H, C) = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\sum_{c=0}^{C-1}|I_1(x,y,c) - I_2(x,y,c)|}{W \times H \times C}$$

Donde I_1e I_2 son las imágenes de entrada, (x, y, c) es una triada que representa el valor numérico del color que se tiene en el pixel que corresponde a las coordenadas x, y.

Como ya se adelantaba este enfoque es muy sensible a escalamientos, ruido y distorsión en las imágenes de entrada. Esto significa que cambios en la resolución, relación de aspecto o tasa de bits pueden tener un impacto dramático en el resultado de la comparación. Este enfoque también es muy ineficiente aún bien implementado, ya que su complejidad aumenta en forma directamente proporcional al tamaño de las imágenes de su entrada. Requiere efectuar al menos $W \times H$ operaciones entre enteros para corroborar la igualdad de un solo par de imágenes.

2.2. Histogramas de Color Global (GCH)

La distribución global del color de una imagen, es una característica que se utiliza ampliamente para calcular el contenido de la imagen. Exhibe características deseadas tales como baja complejidad para la extracción, invariante a escala, rotación, y oclusión parcial [8]. De hecho, es común el uso de Histogramas de Color Global (GCH), por las siglas de su nombre en inglés *Global Color Histogram,* para representar la distribución de colores en una imagen. Partiendo de un modelo de n colores, un GCH es entonces un vector de características $\langle h_1, h_2, \ldots, h_n \rangle$ de dimensión n, donde h_j representa el porcentaje normalizado de píxeles en la imagen correspondiente para cada elemento de color c_j. En este contexto, la recuperación de imágenes similares se basa en la similitud entre sus respectivos GCHs. Una métrica de similitud común es la distancia Euclidiana entre los vectores de características abstractas que representan a dos imágenes, la cual está definida como:

$$d(Q,I) = \sqrt{\sum_{j=1}^{n} \left(h_j^Q - h_j^I\right)^2}$$

Donde Q representa la imagen de búsqueda, I es una de las imágenes en el conjunto de imágenes de la BD y h representa las coordenadas de los vectores de características de estas imágenes. Cuanto menor sea el valor de la distancia mayor es la similitud entre las dos imágenes. Está técnica trata a todos los colores por igual, a pesar de su concentración relativa. Por el contrario, la experiencia nos dice que la percepción de los estímulos, como el del color de una imagen, sigue una curva sigmoidal [9]. De hecho, esto ha sido un fenómeno bien observado con respecto a muchos otros fenómenos relacionados con el grado de sensibilidad que se tiene a diferentes estímulos. Esta idea se ilustra por la línea continua en la Figura 2.1, en donde una percepción lineal (supuesto implícitamente utilizando GCH) se representa por la línea discontinua.

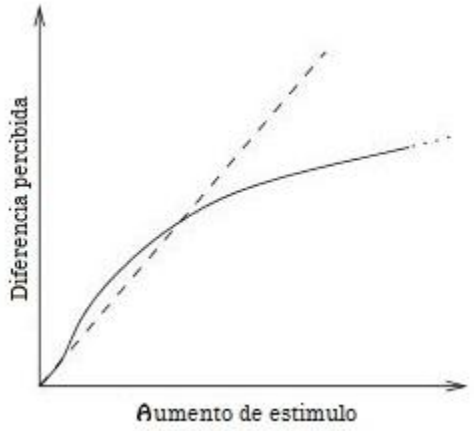

Figura 2.1: Sensibilidad a la percepción de estímulos visuales

2.3. Histogramas de Color usando Vectores de Color Coherente (CCV)

Una técnica refinada y mejorada de los histogramas de color es la llamada Vectores de Color Coherente o CCV por las siglas de su nombre en inglés *Color Coherence Vector* [10]. Un vector de coherencia representa la clasificación de píxeles coherentes y píxeles incoherentes para cada color en la imagen. En lugar de contar sólo el número de píxeles de un color determinado, el CCV calcula pixeles coherentes e incoherentes dentro de cada color j en función del tamaño del grupo de color al que pertenecen. Si el número de píxeles dentro del grupo es mayor que un cierto umbral, el pixel se considera coherente, de lo contrario se toma como incoherente. El grupo de

píxeles se calcula conectando componentes [11]. Por lo tanto, dos valores se asocian con cada color j:

- α_j, es el número de pixeles coherentes del color j
- β_j , es el número de pixeles incoherentes del color j

Entonces el CCV es definido como el vector $\langle(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_n, \beta_n)\rangle$. Teniendo dos CCVs: CCV_1 y CCV_2, estos son comparados basándose en la cantidad de pixeles coherentes e incoherentes:

$$\sum_{j=1}^{n} \left|(\alpha_j^1 - \alpha_j^2)\right| + \left|(\beta_j^1 - \beta_j^2)\right|$$

La diferencia normalizada entre CCV_1 y CCV_2, quedaría como sigue:

$$d_{normalizada} = \sum_{j=1}^{n} \left|\frac{\alpha_j^1 - \alpha_j^2}{\alpha_j^1 + \alpha_j^2 + 1}\right| + \left|\frac{\beta_j^1 - \beta_j^2}{\beta_j^1 + \beta_j^2 + 1}\right|$$

Este enfoque supera al histograma de color básico (GCH) en la recuperación de frames similares, sin embargo, usar CCV produce dos vectores de histograma de color, el coherente y el incoherente. Por lo que este enfoque requiere el doble de espacio de almacenamiento en comparación con el GCH.

2.4. Firmas Binarias usando CBA y VBA

Cabe señalar que GCH y CCV consumen una cantidad significativa de espacio de almacenamiento, debido a que utilizan vectores de grandes dimensiones para representar cada imagen de la base de datos. Por lo que un enfoque que ahorre espacio de almacenamiento y obtenga resultados similares o incluso supere a la forma de utilizar los histogramas de color es altamente deseable. La mayoría de las propuestas anteriores se han dirigido hacia una representación de sólo aquellos colores en el histograma de color que tienen un predominio significativo de píxeles. El enfoque siguiente difiere exactamente en este sentido. Dando más peso a los colores que son menos dominantes sin dejar de tomar en cuenta los demás colores. Con el fin de utilizar firmas binarias para la extracción de imágenes, se diseñó el siguiente esquema:

- Cada imagen en la base de datos se cuantifica en un número fijo de n colores $C = (c_1, c_2, \ldots, c_n)$ para eliminar el efecto de pequeñas variaciones dentro de las imágenes y también para evitar el uso de archivos de gran tamaño debido a representaciones de alta resolución [12].

- Cada elemento de color c_j es discretizado en t *contenedores binarios*, nombrados de aquí en adelante como *bins* $\left(B^j = b_1^j b_2^j \ldots b_t^j\right)$, *a cada bin* se le asigna (usando números naturales) un porcentaje del histograma de color para que lo represente, a esto se le llama tamaño o capacidad del *bin*. Si todos los *bins* tienen la misma capacidad, se dice que dicho arreglo es de un tamaño de Bin Constante o *Constant Bin Allocation* (CBA), si los *bins* son de diferentes tamaños entonces se le nombra de Bin Variable o Variable-Bin Allocation (VBA).

- Los valores normalizados obtenidos después de la extracción automática de color se utilizan en el conjunto correspondiente de *bin* para generar una asignación binaria de valores que indican la presencia o ausencia de un color dentro de un intervalo de densidad particular. Utilizando el enfoque de CBA, cada color c_j tiene sus *bins* de acuerdo con la siguiente condición:

$$b_i^j = \begin{cases} 1 & si\ i = \lceil h_j \times t \rceil \\ 0 & otro\ caso \end{cases}$$

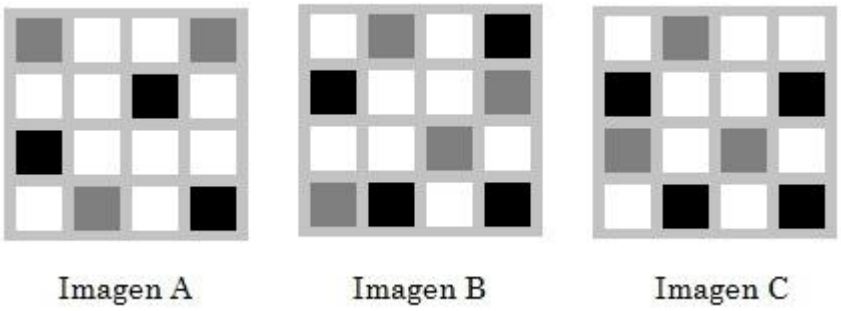

Figura 2.2: Imágenes para ejemplificar la obtención de una firma binaria.

Para ejemplificar el funcionamiento de la obtención de una firma binaria utilizando CBA consideremos la Imagen A de la Figura 2.1, por facilidad hemos hecho $n = 3$, solo tenemos tres colores de modo que $C = (c_1, c_2, c_3) = (negro, gris, blanco)$. Las densidades de color normalizadas por tanto pueden ser representadas por el vector $H_A = (h_1,\ h_2,\ h_3) = (0.1875,\ 0.1875,\ 0.625)$, donde cada h_j representa el porcentaje normalizado de pixeles dominado por el color c_j. Ahora debido a que estamos trabajando con CBA el tamaño de cada *bin* por definición debe ser el mismo para cada *bin*, hagamos $t = 10$ y también que cada *bin* represente 10% del histograma de color, por lo que el vector H_A y la distribución de color que representa deben ser discretizados de tal forma que cada *bin* represente una décima parte del dominio total de color. Por tanto b_1 representa un rango de 1% a 10% de pixeles dominados por el color correspondiente, b_2 representa el rango de 11% a 20% y así sucesivamente hasta llegar a b_{10} que representa el rango de 91% a 100%. Quedando entonces para la Imagen A de la Figura 2.1 compuesta por un 18.75% de C_1, otro 18.75% de C_2 y un 62.5% de C_3 la siguiente firma binaria: $F\_CBA_A =$ 0100000000 0100000000 0000001000, la cual tiene

intencionalmente un espacio entre las firmas individuales de C_1, C_2 y C_3 para facilitar la visualización al lector, dicho espacio no se necesita ni se usa realmente en la firma binaria. Se puede consultar con más detalle la firma binaria F_CBA_A en la Tabla 2.1, así como las firmas de las otras imágenes de la Figura 2.1.

Ahora que hemos ejemplificado la obtención de la firma binaria y el uso de CBA, es idóneo explicar con mayor detalle el uso de VBA y las diferencias que se tienen en comparación con CBA. Primero recordemos la principal y más notable, en VBA el tamaño de cada *bin* puede no ser el mismo, por eso recibe el nombre de *variable*. Pero esto no implica que el tamaño del *bin* cambiará en función del tiempo o de algún otro parámetro. Al igual que en CBA, una vez escogido el tamaño de cada *bin*, dicho valor de permanecerá constante durante todo el proceso de la firma binaria, la verdadera razón de la palabra *variable*, es simplemente que ahora se pueden escoger tamaños diferentes para los diferentes *bins* y esto no es un mero capricho, sino que tiene una razón, de la cual abundaremos en los siguientes párrafos pero básicamente tiene que ver con la cantidad de ceros que aparecen en la firma binaria cuando se utiliza CBA y en la observación de que para distinguir una imagen de otra los colores menos predominantes son más influyentes en el proceso de decisión que los colores más predominantes.

Conjunto de Bins	Densidad de Color	Firma binaria									
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
Imagen A											
c_1/B_A^1	18.75%	0	1	0	0	0	0	0	0	0	0
c_2/B_A^2	18.75%	0	1	0	0	0	0	0	0	0	0
c_3/B_A^3	62.5%	0	0	0	0	0	0	1	0	0	0
Imagen B											
c_1/B_B^1	25%	0	0	1	0	0	0	0	0	0	0
c_2/B_B^2	25%	0	0	1	0	0	0	0	0	0	0
c_3/B_B^3	50%	0	0	0	0	1	0	0	0	0	0
Imagen C											
c_1/B_C^1	25%	0	0	1	0	0	0	0	0	0	0
c_2/B_C^2	18.75%	0	1	0	0	0	0	0	0	0	0
c_3/B_C^3	56.25%	0	0	0	0	0	1	0	0	0	0

Tabla 2.1: Firmas binarias usando CBA

Podemos ejemplificar esto con el siguiente ejemplo, imaginemos dos diferentes imágenes de una figura básica como el círculo, un pequeño círculo dividido en cuatro partes, cada imagen tiene un fondo unicolor distinto y tiene las partes de su respectivo círculo iluminadas de color diferente al de su respectivo fondo. En el momento de querer diferenciar las imágenes, las pequeñas similitudes y diferencias entre los círculos serán más importantes que la gran diferencia del fondo. Si la imaginación no

está dispuesta podemos referirnos a la Figura 2.2. Esta observación es la razón detrás de la implementación de VBA. Recordemos que los *bins* representan a las densidades de color en una imagen, de modo que para enfatizar que las distancias debido a los colores menos dominantes son más importantes debemos variar el tamaño de los *bins* en la forma correcta.

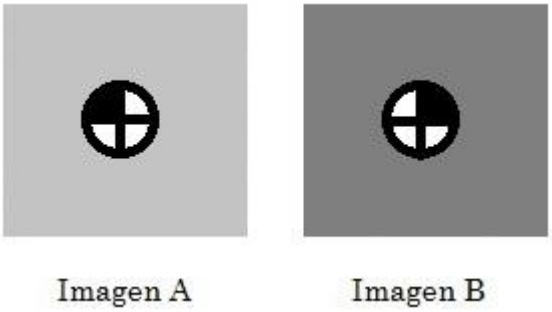

Figura 2.3: Imágenes ilustrativas de la mayor importancia de colores menos dominantes

Para ejemplificar el funcionamiento de VBA en la obtención de una firma binaria consideremos nuevamente a la Imagen A de la Figura 2.1 y a la Tabla 2.2 donde hemos puesto el tamaño de cada uno de los *bins*, así como el porcentaje del histograma de color que representan. Seguiremos usando los mismos valores para los demás parámetros como se usaron en el ejemplo anterior: $t = 10,\ n = 3$ y $C = (c_1, c_2, c_3) = (negro, gris, blanco)$. Las densidades de color normalizadas siguen siendo representadas por el mismo vector $H_A = (h_1,\ h_2,\ h_3) = (0.1875,\ 0.1875,\ 0.625)$. Ahora debido a que estamos trabajando con VBA el tamaño de cada *bin* puede no ser el mismo para cada *bin*, pero para además representar que ahora estamos dando a los colores menos predominantes más peso que a los colores más predominantes usaremos una asignación de tamaños como la mostrada en la Tabla 2.2. Quedando la siguiente firma binaria: $F\_VBA_A =$ 0000100000 0000100000 0000000001 para la Imagen A de la Figura 2.1, cuando se usa VBA. Se puede consultar con más detalle la firma binaria F_VBA_A en la Tabla 2.3, así como las firmas de las otras imágenes de la Figura 2.1.

	Firma Binaria									
	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
Tamaño del bin	3	3	3	5	5	10	10	10	10	41
Porcentaje de la Densidad de Color que representa	[1-3]%	[4-6]%	[7-9]%	[10-14]%	[15-19]%	[20-29]%	[30-39]%	[40-49]%	[50-59]%	[60-100]%

Tabla 2.2: Tamaño de los Bins usando VBA

Conjunto de Bins	Densidad de Color	Firma binaria									
		b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
Imagen A											
c_1/B_A^1	18.75%	0	0	0	0	1	0	0	0	0	0
c_2/B_A^2	18.75%	0	0	0	0	1	0	0	0	0	0
c_3/B_A^3	62.5%	0	0	0	0	0	0	0	0	0	1
Imagen B											
c_1/B_B^1	25%	0	0	0	0	0	1	0	0	0	0
c_2/B_B^2	25%	0	0	0	0	0	1	0	0	0	0
c_3/B_B^3	50%	0	0	0	0	0	0	0	0	1	0
Imagen C											
c_1/B_C^1	25%	0	0	0	0	0	1	0	0	0	0
c_2/B_C^2	18.75%	0	0	0	0	1	0	0	0	0	0
c_3/B_C^3	56.25%	0	0	0	0	0	0	0	0	1	0

Tabla 2.3: Firmas binarias usando VBA

Una vez que hemos completado el proceso de obtención de una firma binaria usando CVA o VBA podemos cuantificar que tan similares son dos imágenes usando, como en los enfoques anteriores, una función de distancia. La métrica está normalizada y es definida como sigue:

$$d(Q,I) = \frac{1}{t}\sum_{j=1}^{n} \left| posición\left(B_Q^j\right) - posición\left(B_I^j\right) \right|$$

Donde $posición\left(B_F^i\right)$ devuelve la posición del bit puesto en uno dentro del conjunto de $bins\ B^i$ de la imagen F. Por ejemplo usando la firma obtenida usando VBA de Imagen A de la Figura 2.2 $posición(B_A^1) = 5,\ posición(B_A^1) = 5\ y\ posición(B_A^3) = 10$.

Siguiendo con los ejemplos anteriores de CBA y VBA, calculemos la distancia entre las firmas de las imágenes en la Figura 2.2, empezando por las firmas obtenidas usando CBA de Figura A y la Figura B tendríamos $d(A,B) = \frac{1}{10}(|2-3| + |2-3| + |7-5|) = 0.4$. El resultado para el cálculo de las demás distancias en todas sus combinaciones pueden verse en la Tabla 2.4, naturalmente la distancia de una imagen consigo misma resulta en cero y la $d(X,Y) = d(Y,X)$.

Distancias cuando se usa CBA			
	Imagen A	Imagen B	Imagen C
Imagen A	0.0	0.4	0.2
Imagen B	0.4	0.0	0.2
Imagen C	0.2	0.2	0
Distancias cuando se usa VBA			
	Imagen A	Imagen B	Imagen C
Imagen A	0.0	0.3	0.2
Imagen B	0.3	0.0	0.1
Imagen C	0.2	0.1	0.0

Tabla 2.4: Distancia entre las imágenes de ejemplo

Finalmente podemos comprobar que tan eficiente es una firma binaria comparándola con los enfoques anteriores. Supongamos que el almacenamiento de un número real requiere $f\ bytes$, por lo tanto, para almacenar el GCH de una imagen compuesta de n colores, serían necesarios $(n \times f)\ bytes$, mientras que la firma binaria para la misma imagen solo requiere $(n \times t)\ bits$. Del mismo modo, los Vectores de Color Coherente (CCV) [10] requerirían $(2 \times n \times f)\ bytes$ para representar dicha imagen. Por tanto, las firmas binarias son mucho más eficientes en espacio que CCV y GCH.

Capítulo 3

Reconocimiento de Pautas Publicitarias

En el capítulo anterior se describieron enfoques para atacar el problema que pueden encajar en el objetivo de este trabajo, pero claramente el enfoque que sobresale entre los anteriores es el de firmas binarias usando VBA, por lo que el siguiente paso en nuestro camino es encontrar la forma más óptima de implementar el enfoque escogido para analizar archivos de video. Pero antes de continuar analizaremos un poco más a fondo la complejidad de las firmas binarias, si existen puntos débiles que se puedan reforzar o si realmente son tan prometedoras como parecen.

Empecemos por notar que una firma binaria no nos aporta ninguna de las características espaciales de la imagen firmada, lo único que nos porta es como se encuentra la distribución de los colores que hemos definido al crearla, es decir que si comparamos la firma de una imagen X, contra la firma de la misma imagen X pero rotada θ grados, la distancia entre ambas firmas valdría cero. Para ilustrar esto, consideremos las imágenes en la Figura 3.1, donde tenemos una imagen y su reflejo sobre el eje horizontal, junto con la firma binaria de cada una. Se han usado y se usarán de aquí en adelante los tamaños de *bins* que son mostrado en la Tabla 3.1 y los siguientes parámetros para obtener la firma: $t = 8$, $n = 3$ y $C = (c_1, c_2, c_3) = (rojo, verde, azúl, amarillo)$.

	Firma Binaria							
	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Tamaño del bin	3	3	3	10	10	10	10	51
Porcentaje de la Densidad de Color que representa	[1-3]%	[4-6]%	[7-9]%	[10-19]%	[20-29]%	[30-39]%	[40-49]%	[50-100]%

Tabla 3.1: Tamaños para nuestra firma binaria.

Es decir, el proceso de obtención de la firma binaria se considera invariante a transformaciones de rotación sobre la imagen. Esto puede ser tomado como fortaleza o robustez como puede ser visto como debilidad, todo dependiendo del uso final que se le

dé a la interpretación de la firma. Si queremos encontrar imágenes que son lo suficientemente parecidas e incluso iguales sin importar si se encuentran rotadas en alguna dirección, entonces es una fortaleza. Si por el contrario nos interesa que las imágenes que nos devuelva la función de distancia sean lo más parecidas posibles, excluyendo a las imágenes que se encuentren afectadas por alguna rotación (aún si fueran exactamente las mismas que la imagen de la consulta), entonces es una debilidad.

Figura 3.1: Comparación entre las firmas de un imagen y su reflejo sobre un eje horizontal

Afortunadamente existe una forma en la que las firmas binarias pueden reponerse de esta arma de doble filo que es su invariancia a la rotación de las imágenes de entrada. La solución propuesta es: antes de calcular la firma binaria dividir en cuatro cuadrantes la imagen (podrían ser mayor el número de divisiones, pero en nuestros experimentos hemos encontrado que la relación complejidad/eficiencia de este proceso es más óptimo cuando nos limitamos a cuatro cuadrantes) y después calcular la firma individual de cada cuadrante, una vez obtenidas, las cuatro firmas se concatenan para formar una nueva firma binaria, el orden para la concatenación será el siguiente: $[C1\ C2\ C3\ C4]$. Con esto nuestra firma binaria incluya ahora intrínsecamente el elemento espacial de la imagen del que carecía. La nueva firma binaria se ejemplifica en la Figura 3.2, en donde puede apreciarse que las firmas resultantes no son iguales.

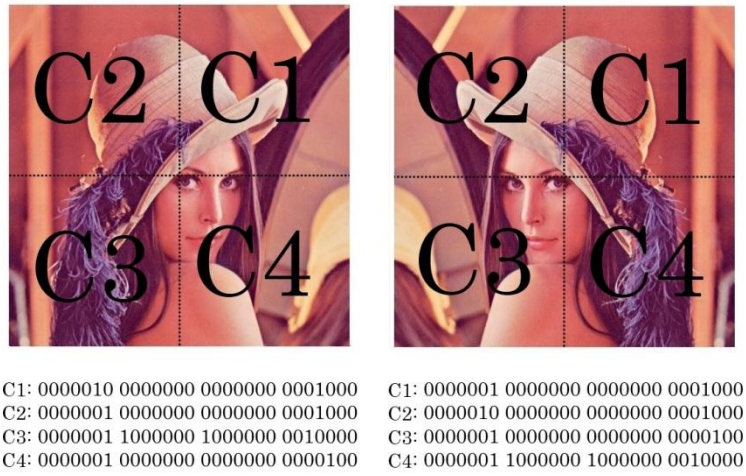

Figura 3.2: Firmas Binarias utilizando los cuatro cuadrantes

Al realizarse este nuevo paso en la construcción de la firma binaria cambia naturalmente su tamaño, se cuadruplica para ser exactos, pero a cambio de este incremento la firma binaria adquiere la robustez de poder nombrar como diferentes a imágenes rotadas respecto a su original. Aunque debido a que la construcción original de la firma es un modelo muy compacto en cuanto a tamaño, cuadriplicar dicho tamaño no supone un golpe tan grave a la eficiencia de la misma. En general, aún efectuando la división de cuadrantes y considerando un número de colores reducido (digamos entre cuatro y ocho) una firma binaria no va a superar el 5% del tamaño de la imagen de la que se obtuvo. Inclusive si se consideraran todos los colores y sus diferentes gamas (lo cual es realmente impráctico) y se dividiera la imagen en 16 partes en vez de cuatro, el tamaño resultante no sería mayor al ocupado por la imagen original. Por lo que el aplicar este nuevo paso a nuestras firmas no es razón suficiente para descartar el enfoque y comenzar la búsqueda de otro. Inclusive si el aumento de espacio fuera un problema más serio, un análisis más a detalle de la estructura que tiene la firma resultante nos indica que cada conjunto de *bins* (B^j) a lo más tendrá un bit puesto en uno y los demás serán ceros, con lo cual podríamos simplemente guardar en un número binario la posición del bit puesto en uno en lugar de todo el conjunto de *bins*, lo que haría que el tamaño de la firma pasará a ser del orden de $\lceil 4log_2 t \rceil$ por cada color en la firma. Esto debido a que cada conjunto de *bins* consta de t *bins* y se realizarían cuatro firmas individuales por cada imagen.

Ahora es razonable esperar que se use un número reducido de colores pero en caso de que esto no fuera cierto, podemos también esperar una gran secuencia de ceros, lo que permite el uso de técnicas de compresión, por ejemplo *Run-Length Encoding* (RLE) [13] que es una codificación muy sencilla que reduce en gran medida la longitud de lo que se desea codificar cuando existen muchos símbolos repetidos, lo cual encaja en este caso y nos permitiría ahorrar aún más espacio de almacenamiento. Por lo tanto y siendo que el ahorro de espacio de almacenamiento no es el objetivo principal de este trabajo nos quedaremos con ($4n\lceil log_2 t \rceil$) como cota asintótica inferior para describir que tanto espacio se estaría necesitando para la firma binaria en función de los parámetros n (número de colores) y t (número de *bins*).

3.1. Representación de la señal de Video

Hasta este momento tenemos seleccionada a las firmas binarias como la metodología para adaptar a nuestro problema de reconocimiento de video, pero no hemos hablado de videos ni de la forma en que se codifica la señal de video para ser transmitida, por lo que en esta sección se describirá de forma general el proceso que se lleva a cabo, con el fin de tener un panorama más adecuado para continuar con la implementación de las firmas binarias en el reconocimiento de video.

Empecemos reconociendo que para conseguir cierto grado de calidad en un video transmitido por un canal ruidoso existe un complicado equilibrio entre diversos factores como son:

- cantidad de datos necesarios para representarlo (también conocida como tasa de bits),
- complejidad de los algoritmos de codificación y decodificación,
- robustez frente a las pérdidas de datos y errores,
- y para nuestro interés en particular la facilidad de acceder a los frames que componen el video.

Afortunadamente ya existen algoritmos de compresión y descompresión en tiempo real que se encargan de lidiar con los problemas inherentes a la transmisión: los códec. Su finalidad es obtener un almacenamiento sustancialmente menor de la información de vídeo. Esta se comprime en el momento de guardar la información hacia un archivo y se descomprime, en tiempo real, durante la visualización. Fueron diseñados para que el proceso sea transparente para el usuario, es decir, que no intervenga o lo haga lo menos posible.

3.1.1. Códec de Video

Códec es la abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (stream) o una señal. Los códecs pueden codificar una señal para su transmisión, almacenaje o cifrado y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones. Son usados a menudo en videoconferencias y emisiones de medios de comunicación, aunque la mayoría provoque pérdidas de información para conseguir el tamaño más pequeño posible del archivo final. También existen códecs sin pérdidas o *lossless*, pero en la mayor parte de las aplicaciones prácticas, no tiene sentido un aumento considerable del tamaño de datos para un casi imperceptible aumento en la calidad. La excepción es si los datos sufrirán otros tratamientos en el futuro. En este caso, una codificación repetida con pérdidas a la larga dañaría demasiado la calidad.

Un códec de video es un tipo de códec que permite comprimir y descomprimir video digital. Como se dijo anteriormente la mayoría de los códecs, incluyendo los de video, comprimen la información para que pueda ser almacenada o transmitida ocupando el menor espacio posible y este proceso normalmente utiliza algoritmos de compresión que ocasionan una pérdida de información. Durante el proceso de compresión se aprovecha que las secuencias de video tienen redundancia en las dimensiones espacial y temporal. Por lo tanto, eliminando dicha información redundante se consigue codificar la información de manera más óptima. Para eliminar la información redundante en el plano temporal se utiliza la predicción por compensación de movimiento, donde se estima el movimiento entre bloques sucesivos de la imagen. Los bloques son formados por grupos de píxeles, generalmente de 8x8 o 16x16. Posteriormente se codifica la información de los vectores de movimiento y del error de predicción respecto al bloque anterior. Para eliminar la información redundante en el plano espacial se decorrela el error intercuadro y se hace la codificación de los coeficientes de la DCT. Podemos ver un resumen de la historia de los estándares en la Tabla 3.2.

Los códecs de vídeo buscan representar un conjunto de datos fundamentalmente analógicos en un formato digital. Debido al diseño de señales de video analógicas que representan por separado la luminancia ("luma") y la información de color (crominancia, "chroma"), un primer paso común en la compresión de imagen en el diseño de códec es representar y almacenar la imagen en un espacio de color YC_bC_r. [14] Pasar al espacio YC_bC_r proporciona dos ventajas: en primer lugar, mejora la compresibilidad proporcionando una decorrelación de las señales de color; Y en segundo lugar, separa la señal luma, que es perceptualmente mucho más importante, de la señal chroma, que es perceptualmente menos importante y que se puede representar con menor resolución para conseguir una compresión de datos más

eficiente. Es común representar las proporciones de la información almacenada en estos diferentes canales de la siguiente manera $Y:C_b:C_r$, donde Y hace referencia a la señal luma, C_b y C_r son la componente de crominancia expresada en diferencia de azul y diferencia de rojo respectivamente. [15]

Año	Estándar	Publicador	Implementaciones
1984	H.120	ITU-T	
1988	H.261	ITU-T	Videoconferencia, videotelefonía
1993	MPEG-1 Part 2	ISO, IEC	Video-CD
1995	H.262/MPEG-2 Part 2	ISO, IEC, ITU-T	DVD Video, Blu-ray, Digital Video Broadcasting, SVCD
1996	H.263	ITU-T	Videoconferencia, videotelefonía, video en teléfonos celulares(3GP)
1999	MPEG-4 Part 2	ISO, IEC	Video en Internet (DivX, Xvid ...)
2003	H.264/MPEG-4 AVC	Sony, Panasonic, Samsung, ISO, IEC, ITU-T	Blu-ray, HD DVD, Video Digital Broadcasting, Video para iPod, Apple TV, videoconferencia
2009	VC-2 (Dirac)	SMPTE	Video en Internet, HDTV broadcast, UHDTV
2013	H.265	ISO, IEC, ITU-T	

Tabla 3.2: Historial de estándares en compresión de video

Muchos archivos multimedia contienen tanto datos de audio como de vídeo, y alguna referencia que permite la sincronización del audio y el vídeo. Cada uno de estos tres flujos de datos puede ser manejado con programas, procesos, o hardware diferentes; pero para que estos streams sean útiles para almacenarlos o transmitirlos, deben ser encapsulados juntos. Esta función es realizada por un formato de archivo de vídeo (contenedor), como .mpg, .avi, .mov, .mp4, .rm, .ogg, .mkv, .tta, etcétera. Algunos de estos formatos están limitados a contener streams que se reducen a un pequeño juego de códecs, mientras que otros son usados para objetivos más generales.

3.1.2. Codificación de video Intra-Frame

La predicción Intra-frame se aprovecha de la redundancia espacial en los frames de vídeo, es decir, la correlación entre píxeles dentro de un frame, calculando los valores de predicción a través de la extrapolación de píxeles ya codificados para la codificación delta efectiva. Es una de las dos clases de métodos predictivos de codificación en codificación de video. Su contraparte es la predicción entre tramas que explora la redundancia temporal. Las tramas internas codificadas de forma independiente independientemente utilizan solamente la codificación intra. Las tramas predichas temporalmente codificadas (por ejemplo, los frames P y B de MPEG) pueden utilizar la predicción Intra-frame así como la Inter frame.

Por lo general, sólo algunas de las muestras espacialmente más cercanas se utilizan para la extrapolación. Los formatos que operan muestra por muestra como PNG (Portable Network Graphics) pueden usualmente usar uno de los cuatro píxeles adyacentes (izquierda, arriba, arriba a la izquierda, arriba a la derecha, como lo ilustra la Figura 3.3) o alguna función de los mismos, como por ejemplo su promedio.

Los formatos basados en bloques (transformaciones de frecuencia) llenan bloques enteros con valores de predicción extrapolados normalmente de una o dos líneas rectas de píxeles que corren a lo largo de sus bordes superior e izquierdo.

Figura 3.3: Pixeles adyacentes que se toman en cuenta para formatos como PNG

El término codificación Intra-frame se refiere al hecho de que las diversas técnicas de compresión sin pérdidas y con pérdidas se realizan con respecto a la información que está contenida solamente dentro de la trama actual y no con relación a ninguna otra trama en la secuencia de vídeo. En otras palabras, no se realiza procesamiento temporal fuera de la imagen o frame actual. Las técnicas de codificación no Intra son extensiones a estos fundamentos. Resulta que este diagrama de bloques es muy similar al de un codificador de vídeo de imagen fija JPEG, con sólo ligeras diferencias de detalle de implementación.

3.1.3. Codificación de video Inter Frame

Un *Inter Frame* es una frame en un flujo de compresión de vídeo que se expresa en términos de uno o más frames vecinos. La parte "inter" del término se refiere al uso de la predicción entre frames. Este tipo de predicción trata de aprovechar la redundancia temporal entre frames vecinos que permiten mayores tasas de compresión. La codificación *Inter Frame* divide los frames a codificar en bloques conocidos como *macrobloques*. Después de eso, en lugar de codificar directamente los valores de píxel sin procesar para cada bloque, el codificador intentará encontrar un bloque similar al que está codificando en un frame previamente codificado, denominado frame de referencia. Este proceso se realiza mediante un algoritmo de concordancia de bloques. Si el codificador tiene éxito en su búsqueda, el bloque podría ser codificado por un vector, conocido como vector de movimiento, que apunta a la posición del bloque de coincidencia en el frame de referencia. El proceso de determinación del vector de movimiento se llama estimación de movimiento. En la mayoría de los casos, el codificador tiene éxito, pero el bloque encontrado probablemente no es una coincidencia exacta con el bloque que está codificando. Por lo cual el codificador calculará las diferencias entre ellos. Esos valores residuales se conocen como el error de predicción y necesitan ser transformados y enviados al decodificador. En resumen, si el codificador logra encontrar un bloque de coincidencia en una frame de referencia, obtendrá un vector de movimiento que apunta al bloque emparejado y un error de predicción. Utilizando ambos elementos, el decodificador será capaz de recuperar los píxeles sin procesar del bloque. La Figura 3.4 muestra gráficamente todo el proceso.

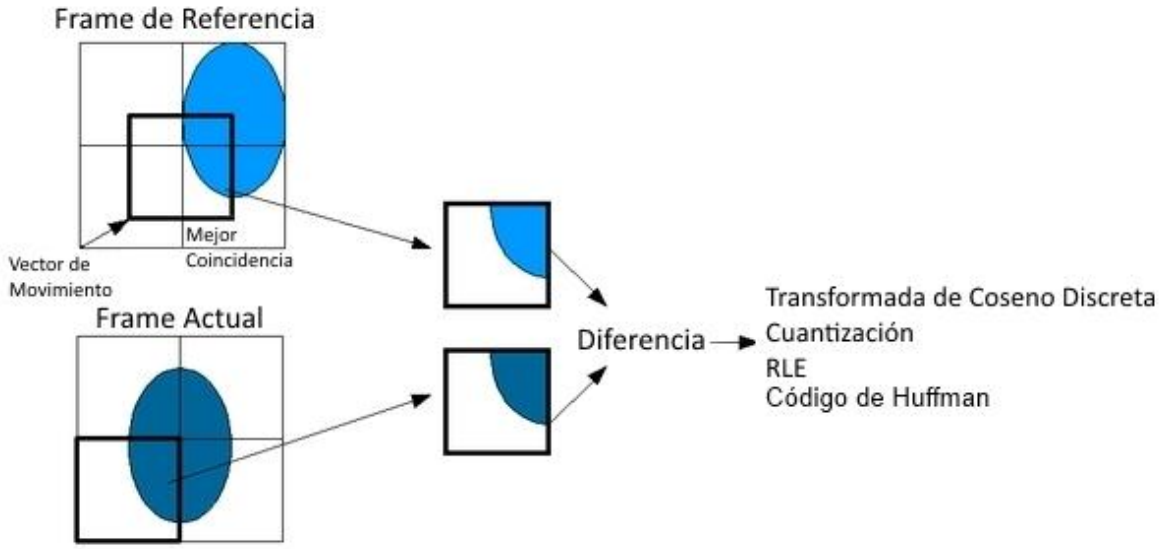

Figura 3.4: Proceso de predicción Inter-frame

Este tipo de predicción tiene algunos pros y contras:

- Si todo va bien, el algoritmo será capaz de encontrar un bloque de coincidencia con poco error de predicción para que, una vez transformado, el tamaño general del vector de movimiento más el error de predicción es menor que el tamaño de una codificación en bruto.

- Si el algoritmo de coincidencia de bloques no encuentra una coincidencia adecuada, el error de predicción será considerable. Por lo tanto, el tamaño total del vector de movimiento más el error de predicción será mayor que la codificación en bruto. En este caso, el codificador haría una excepción y enviaría una codificación en bruto para ese bloque específico.

- Si el bloque coincidente en la trama de referencia también ha sido codificado utilizando la predicción de Inter Frame, los errores cometidos para su codificación se propagarán al siguiente bloque. Si cada trama se codificaba utilizando esta técnica, no habría manera de que un decodificador se sincronice con un stream de vídeo porque sería imposible obtener las imágenes de referencia.

Debido a estos inconvenientes, debe utilizarse un frame de referencia periódico y confiable para que esta técnica sea eficiente y útil. Ese frame de referencia se conoce como *Intra-frame*, que está *intra* codificado, por lo que siempre se puede decodificar sin información adicional.
En la mayoría de los diseños, hay dos tipos de inter frames: P-frames y B-frames. Estos dos tipos de frames y los I-frames (imágenes *intra-codificadas*) por lo general se unen en un Grupo de imágenes (GOI, por sus siglas en inglés). El I-frame no necesita información adicional para ser descodificada y puede utilizarse como una referencia fiable. Esta estructura también permite obtener una periodicidad del I-frame, que es necesaria para la sincronización del decodificador.

3.2. Frames de Video y Grupos de Imágenes (GOP)

Con lo que hemos aprendido hasta ahora podemos notar que existen diferentes tipos de frames, con diferentes propósitos, los I-frames, los P-frames y los B-frames. Básicamente los I-frames son una referencia para los B-frames y P-frames. Los P-frames y B-frames expresan un cambio en su frame referencia y la diferencias entre ellos es el frame de referencia que se les permite usar.

- **I-Frame** es un tipo de frame de referencia independiente de los otros tipos de frames, ya que está *intra* codificado, por lo que siempre se puede decodificar sin información adicional. Cada GOP empieza por una frame de este tipo.

- **P-frame** es el nombre para definir las imágenes predichas hacia adelante. La predicción se realiza a partir de una imagen anterior, principalmente una I-frame, por lo que requieren menos datos de codificación (≈50% en comparación con el tamaño de un I-frame). La cantidad de datos necesarios para realizar esta predicción consiste en vectores de movimiento y coeficientes de transformación que describen la corrección de la predicción. Implica el uso de compensación de movimiento.

- **B-frame** es el término para imágenes bidireccionalmente predichas. Este tipo de método de predicción ocupa menos datos de codificación que los P-frames (≈25% cuando se compara con el tamaño de un I-frame) porque pueden predecirse o interpolarse a partir de una trama anterior y/o posterior. Similar a los P-frames, los B-frames se expresan como vectores de movimiento y coeficientes de transformación. Con el fin de evitar un creciente error de propagación los B-frames no se utilizan como una referencia para hacer predicciones adicionales en la mayoría de las normas de codificación. Sin embargo, en los métodos de codificación más recientes como el Códec de Vídeo Avanzado (AVC, por sus siglas en inglés), B-frames se pueden utilizar como referencia.

Ahora que hemos definido los tipos de frames que se utilizan en la codificación de video, veamos lo que es el llamado Grupo de Imágenes (GOP, por sus siglas en inglés). Un GOP siempre empieza con un I-frame. A continuación, le siguen varios P-frames, en cada caso, con varios frames de distancia. Finalmente, los huecos restantes son ocupados por los B-frames. Cuantos más I-frame haya en un stream de vídeo, más fácil será su edición, pero en contraposición este stream ocupará más tamaño. Para ahorrar ancho de banda y espacio en el disco, los vídeos preparados para su difusión digital, solo tienen un I-frame por GOP. La estructura GOP suele estar referenciada por dos valores M y N. El primero de ellos nos dice la distancia que hay entre dos P-frame o entre un I-frame y un P-frame. El segundo nos dice la distancia que hay entre dos I-frame: es decir la longitud del GOP, por ejemplo para M=3, N=9 la estructura que le correspondería sería: IBBPBBPBBI. El I-frame se utiliza para predecir el primer P-frame y estos dos frames también se utilizan para predecir el primer y segundo B-frame. El segundo P-frame se predice también utilizando el primer I-frame. Ambos P-

frame se unen para predecir el tercero y cuarto B-frame. El esquema se muestra en la Figura 3.5, donde los vectores inferiores indican las referencias predictivas.

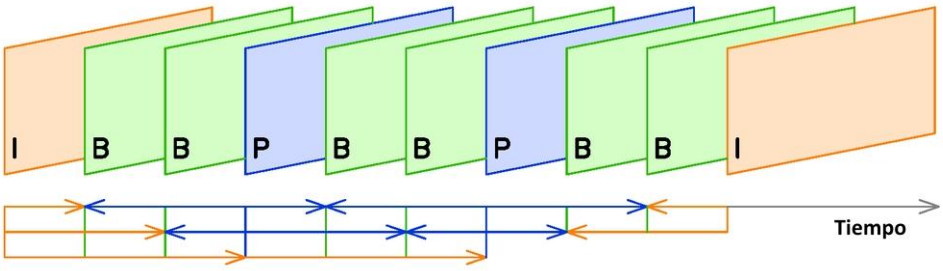

Figura 3.5: Secuencia de frames en un periodo de tiempo.

Esta estructura sugiere un problema porque el cuarto frame (un P-frame) es necesaria para predecir el segundo y el tercero (B-frame). Así que necesitamos transmitir el P-frame antes del B-frame y retrasar la transmisión (debido a que necesitamos mantener guardado en el búfer el P-frame).

Esta estructura tiene puntos fuertes:

- Minimiza el problema de posibles áreas descubiertas.
- P-frames y B-frames necesitan menos datos que los I-frames, por lo que se transmiten menos datos.

Pero también tiene puntos débiles:

- Aumenta la complejidad del decodificador, lo que puede significar que se necesita más memoria para reorganizar los frames.
- Los frames interpolados (es decir, los B-frames) requieren más vectores de movimiento lo que significa incrementar la tasa de bits.

3.3. Aplicando las firmas binarias al video codificado

Hasta ahora hemos definido un archivo de video como una sucesión de imágenes en el tiempo, dicha definición intuitivamente nos habla de imágenes completas, es decir, que no necesitan de otras imágenes para poder visualizarse. Tal como las fotos tomadas en un día de campo y luego ordenadas en un álbum familiar, en el cual podemos revisar las fotografías en el orden que deseemos, podemos empezar por el principio, por el medio o por el final. También podemos quitar una de las fotografías del álbum dejando el espacio vacío y naturalmente no afectaremos a ninguna de las otras fotografías. Pero ahora sabemos que un stream de video no es como un álbum de fotografías, debido a que los archivos de video son codificados para ocupar un menor espacio de almacenamiento. En la sección anterior aprendimos que cuando se transmite un stream de video digital realmente no se están transmitiendo muchas imágenes sucesivas, sino que transmite un conjunto donde existe una imagen completa que sirve de referencia y el resto son vectores de movimiento e instrucciones para reconstruir otras imágenes que deberían estar en el conjunto a partir de la imagen de referencia. Una vez que se inicia el proceso de reproducción del video transmitido, este se descodifica para poder rearmar la sucesión de imágenes que lo componen y

caracterizan. Es decir, la entrada del decodificador de video es un stream de I-frames, B-frames y P-frames ordenados en múltiples GOP´s y la salida es una sucesión de imágenes completas que se proyectaran sobre la pantalla a la velocidad de visualización que marque el códec de video.

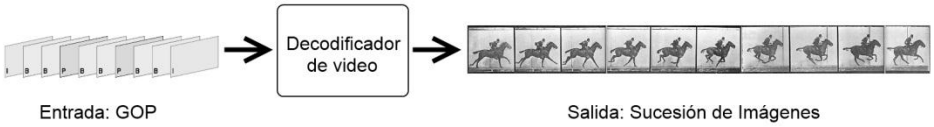

Figura 3.6: Entrada y salida simplificadas de un decodificador de video.

De modo que para la implementación de la firma binaria en archivos de video, sería de gran interés no trabajar con la salida del codificador de video, sino con la entrada. Es decir con los GOP y en específico con los I-frames. Ya que cada GOP contiene solo un I-frame y dicho I-frame no se repite en los demás GOP, cada I-frame es un descriptor suficientemente buenos de su GOP y considerando que cada GOP puede contener entre nueve y doce frames, trabajar solamente con los I-frame reduciría el tiempo de procesamiento y la longitud de la firma resultante entre nueve y doce veces comparado con el que se necesitaría para firmar todos las imágenes en la salida del codificador de video. De modo que esta es la metodología elegida para atacar el problema, aplicar la firma binaria solamente a los I-frame de los videos con los que estaremos trabajando. Y ahora que ya tenemos la metodología seleccionada podemos empezar a hablar del siguiente paso: la implementación.

3.4. Descripción de la implementación de firmas binarias en archivos de video

Para la implementación hemos seleccionado trabajar con videos codificados en MPEG, ya que este formato maneja su codificación con I-Frames y queremos aprovechar esta característica. Aclarando que existen formatos de video como los MP4, que también trabajan con I-Frames, pero el número de I-frames en estos es menor debido a la mayor compresión que tiene dicho formato. Además la norma en Televisión Digital Terrestre designa al formato MPEG como el que se debe usar al transmitir, por lo que todo parece apuntar a que la decisión es la correcta. En cuanto a las herramientas que se usarán podemos nombrar a OpenCV, que al estar liberado bajo la licencia BSD (Berkeley Software Distribution), es libre para uso tanto comercial como académico [16] y además tiene amplio soporte para funcionar sobre diferentes lenguajes de programación como son C, C++, Python y Java y soporte para Windows, Linux, Mac OS, iOS y Android. Empezaremos a trabajar sobre Python, un robusto lenguaje interpretado de alto nivel que cuenta con diversas librerías. Una de las librerías de Python que nos servirán es Numpy, esta librería nos facilitará las operaciones con los arreglos y matrices en donde cargaremos los datos de los archivos de video. Ahora a pesar de sus múltiples virtudes ni Python ni OpenCV contemplan formas de trabajar con los archivos de video codificados, por lo que nos ayudaremos de un programa externo a ambos, existen numerosas formas de obtener los I-Frames de un video codificado, incluso algunas con interfaz gráfica pero para nuestros fines

usaremos una muy sencilla, prácticamente es un comando. Nuestro nuevo amigo se llama **FFmpeg** está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Windows. Es una colección de software libre que puede grabar, convertir (transcodificar) y hacer streaming de audio y vídeo, incluye una biblioteca de códecs (libavcodec) [17] y nos permitirá obtener los I-Frames que necesitamos de los archivos de video, así como convertir a **MPEG** videos que se encuentren en otros formatos. Un diagrama de bloques donde se muestra la metodología que seguiremos para la implementación se puede ver en la Figura 3.7.

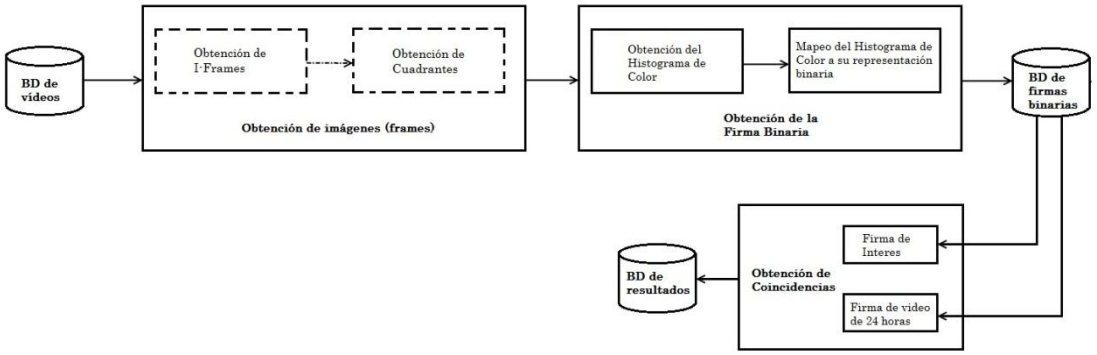

Figura 3.7: Diagrama de bloques de la metodología.

3.4.1. Obtención de I-Frames con FFmpeg

Como hemos explicado anteriormente la forma más eficiente de obtener los I-Frames de un video es previo a la reproducción del mismo, es posible saber el tipo de Frame en tiempo de reproducción pero esto implicaría que el procesamiento para pasar de los P-Frames y B-Frames a imágenes completas ya se realizó y por lo tanto tendríamos un desperdicio de recursos dado que no vamos a ocupar las imágenes obtenidas de P-Frames ni de B-Frames. Afortunadamente FFmpeg nos permite filtrar los I-Frames previamente a la reproducción del video. Su uso es muy simple, una vez instalado basta con invocar al programa y decirle que queremos que nos devuelva los I-Frames del video de entrada, así como el número de Frame que tiene dentro del video para no perder la secuencia de reproducción. También podemos guardar el tiempo de reproducción que le correspondería a cada I-Frame. FFmpeg nos devolverá los I-Frames convertidos en imágenes JPEG y la información de los tiempos de reproducción y números de Frame quedaran en un archivo. Para poder hacer esto dentro de nuestro código en Python basta con importar la librería OS y pasar el comando en una cadena de texto, escapando los caracteres especiales, como se muestra en la Figura 3.8.

```
1   # FirmaVideoUsandoCuadrantes.py : Obtiene la firma del video deseado usando
    OpenCV, FFMPEG y Python
2   import os           #Para ejecutar comandos de consola externos a Python
.                                   .
.                                   .
.                                   .
172 #Obtenemos los keyframes/I-frames del video, usando el nombre/ruta del video
    os.system("ffmpeg -i " + args["video"] +" –vf
    select=\"eq(pict_type\,PICT_TYPE_I)\" -vsync 2 -f image2 \"I-frame_" +
    args["video"] + "_%03d.jpeg\" -loglevel debug 2>&1 | grep \"^\[\" | grep
    \"pict_type:I scene:nan -> select:1\" | cut -d \" \" -f 4,6 -> " + args["video"] +
    "_keyframe-timecodes.txt")
```

Figura 3.8: Fragmento de código en Python para ilustrar el uso de FFMPEG

La línea 172 de código Python mostrado en la Figura 3.8 es una llamada a la función que nos permite ejecutar un comando en la consola de nuestro Sistema Operativo, el comando que estamos ejecutando es **ffmpeg**, sobre el cual influyen algunos modificadores, como el modificador –i que indica el nombre del archivo de video (si el video se encontrara en una carpeta diferente de la actual, debe indicarse la ruta de la misma), el argumento del modificador –vf indica que queremos filtrar los I–Frames, –f image2 le indica a **ffmpeg** que va a guardar los I–Frames filtrados en archivos individuales de imagen con un nombre consecutivo. La salida del comando es un archivo donde se lista el número de frame donde se encontró un I–Frame y el tiempo en segundos que le corresponde al frame encontrado para aparecer en la reproducción del video. Estos datos nos serán de utilidad después del proceso de reconocimiento, cuando tengamos que reportar la información de la pauta reconocida.

3.4.2. Detección de Colores con Python

Uno de las primeras decisiones que deben tomarse al empezar la implementación es el número de colores que se van a usar y cuáles serán esos colores, porque en los ejemplos que se han visto han sido imágenes en escala de grises, pero a partir de aquí trabajaremos con lo que se ve en la televisión actualmente que son imágenes a color. Empezaremos la implantación usando cuatro colores: rojo, verde, azul y amarillo. Esta elección en principio solo contenía a los tres primero colores, motivados por la intuición de que los colores primarios del modelo RGB serían una buena base para el histograma de color. Posteriores indagaciones en modelos publicitarios y teorías del color para la mercadotecnia nos motivaron a agregar el color amarillo a nuestro conjunto de colores reconocibles en el histograma de color y también nos mantenemos abiertos a la idea de agregar o quitar colores para mejorar el rendimiento en los experimentos posteriores. Pero saber los colores que vamos a reconocer solo es la mitad de la solución, ya que es extremadamente raro encontrar estos o cualquier color en concentraciones convencionales en videos de cualquier tipo, en general lo que se suele hallar son tonalidades más claras o más oscuras de los colores. Por ejemplo, tenemos definido en el modelo RGB el color rojo como (255, 0, 0). Por lo que para decir que un

pixel o una zona de pixeles son rojos deben tener exactamente (255, 0, 0) como valor de color. Lo cual raramente llega a pasar, debido a que encontrar un rojo o cualquier color 100% digamos "puro" o mejor dicho con concentraciones exactas a las que podamos definir como alguna constante es muy improbable, la mejor manera de detectar colores no es definirlos de una forma cerrada, sino más bien de una forma más abierta. Es decir con rangos, por lo que para cada uno de nuestros colores tendremos un rango que define los límites dentro de los cuales podemos decir que un pixel pertenece a dicho color. Los límites de color para los cuatro colores antes mencionados son definidos en código Python como muestra la Figura 3.9.

```
1   # FirmaVideoUsandoCuadrantes.py : Obtiene la firma del video deseado usando
    OpenCV, FFMPEG y Python
2   import os           #Para ejecutar comandos de consola externos a Python
.                                   .
.                                   .
.                                   .
199 boundaries =  [([0, 0, 100], [128, 128, 255]),      #Rango del color rojo
                   ([0, 50, 0], [128, 255, 128]),       #Rango del color verde
                   ([100, 0, 0], [255, 128, 128]),      #Rango del color azul
                   ([0, 100, 100], [128, 255, 255])]    #Rango del color amarillo
```

Figura 3.9: Fragmento de código en Python ejemplificando las fronteras de color definidas

En la lista denominada "boundaries" dentro del código mostrado en la Figura 3.9 tenemos cuatro elementos, cada elemento es una dupla de valores de RGB, el primer elemento de cada dupla es el límite inferior y el segundo elemento es el límite superior del rango que hemos seleccionado para representar a nuestros colores. Cabe señalar un detalle en la implementación para Python y es que a diferencia de otros lenguajes los arreglos que representan componentes de RGB tienen el siguiente formato: [Azul (**B**), Verde (**G**), Rojo (**R**)], en lugar del tradicional e intuitivo [Rojo (**R**), Verde (**G**), Azul (**B**)]. Si esto llegará a ser demasiado confuso, se puede escribir una función que mapee la forma de representación del RGB en Python a la forma digamos "intuitiva". Aunque esto sería meramente una ayuda al programador y no tendría mayor impacto en el resto del código. Para mostrar el alcance de los rangos de RGB que hemos elegido para definir a nuestros colores, se muestra a continuación en la Figura 3.10 cuatro ventanas de Python compuestas de dos imágenes, la imagen de la izquierda (repetida en las cuatro ventanas) es una circulo iluminado de diferentes tonalidades de colores y la imagen de la derecha es el mismo circulo pero aplicando un filtro para identificar los colores rojo, verde, azul y amarillo tal cual los hemos definido usando los rangos anteriores.

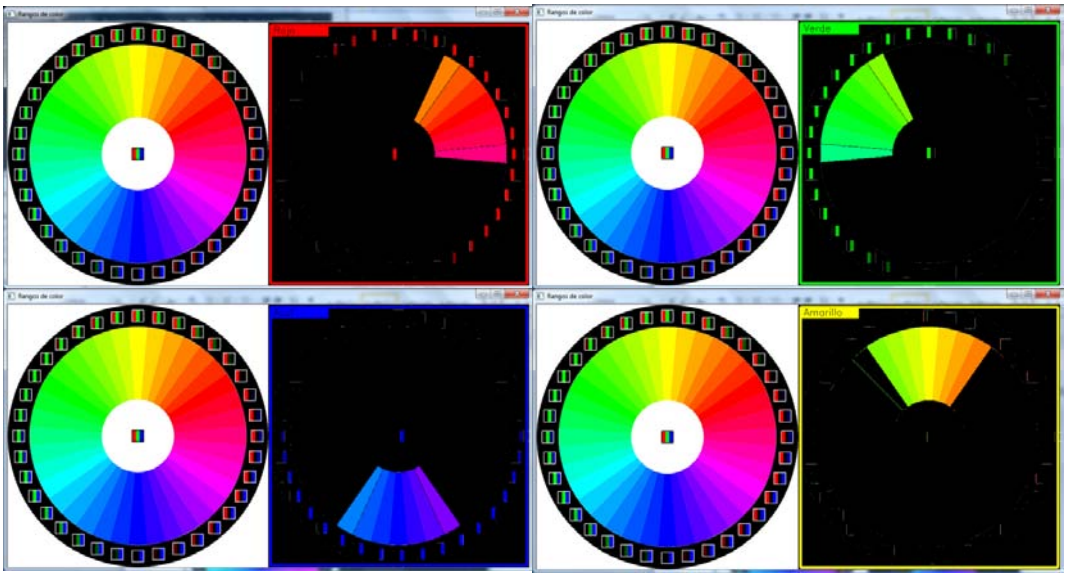

Figura 3.10: Reconocimiento de los colores rojo, verde, azul y amarillo.

Recordemos que cuando nos encontramos con imágenes o porciones de imágenes que no contienen alguno de los colores que hemos definido para efectuar la firma binaria por definición se les asigna un cero al *bin* correspondiente, por lo que si la firma de una imagen resultara ser cero en los cuatro *bins*, esto implicaría que la imagen no posee más de tres por ciento de los colores que definen la firma y por lo tanto los colores que forman dicha imagen no se contemplaron en la definición de la firma binaria. El ejemplo más ilustrativo de esto es una imagen de cualquier tamaño completamente negra o blanca, su firma binaria será un cero en cada *bin*. Esto ocurrirá siempre que nos encontremos con imágenes cuyos colores no tengamos contemplados en la definición de la firma. En la Figura 3.10 podemos notar que existe una porción del área del círculo que no fue identificada por ninguno de los cuatro colores. Los colores que podemos identificar y los que no podemos, del círculo de colores, aparecen claramente en la Figura 3.11, del lado izquierdo aparecen las tonalidades que caen dentro de alguno de los rangos de color definidos y del lado derecho las que no entran en dichos rangos. Este es un problema menor, debido a que es muy raro encontrar un comercial que requiera ser capaces de identificar una amplia variedad de colores, es más practico concentrarse en los que son más probables de aparecer en dicho comercial de televisión.

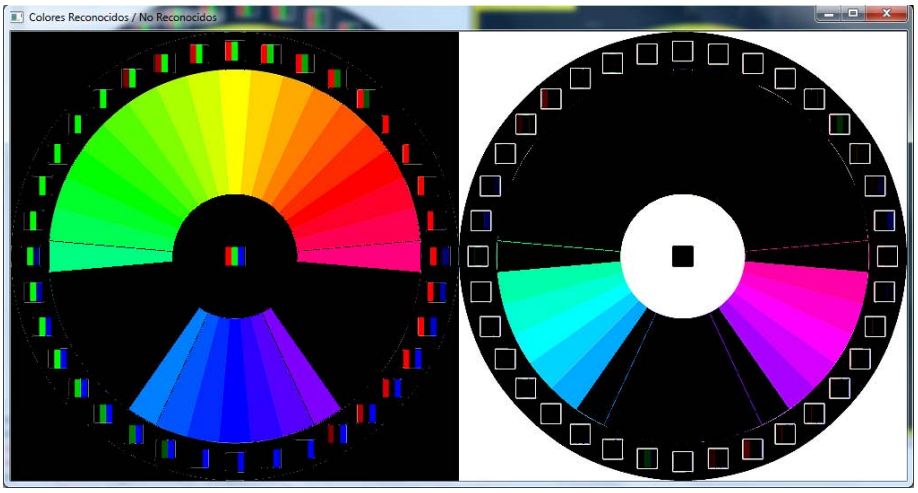

Figura 3.11: Colores que entran en los rangos definidos y colores que no.

Otro detalle en cuanto a la definición de los rangos de color es que en lo posible se debe cuidar que cada uno sea disjunto de los demás, aunque los rangos obviamente se han creado a partir de una particular percepción humanan del color, diferentes personas suelen tener diferentes opiniones respecto a que tono pertenece a cual color, esta fascinante discusión sobre la pertenencia de una tonalidad a un color u otro no tiene mayor indagación en este trabajo más que el reconocimiento de su existencia. Los rangos de colores elegidos lo han sido así porque demostraron un buen reconocimiento en los experimentos para comerciales de televisión. El alcance de este trabajo nos limita a dichos comerciales, pero si el tema fuera, por ejemplo, la clasificación de pinturas renacentistas o barrocas probablemente un ajuste a los colores y sus rangos sería benéfico en los experimentos.

3.4.3. Firmando videoclips con Python

Para la implementación de en Python se decidió usar la versión de la firma binaria con cuadrantes, aunque esto puede ser modificado fácilmente en la interfaz discutida en la Sección 3.5. Naturalmente después de decidir si se usaran los cuadrantes o no, se debe mantener esta decisión para todo el proceso, debido a que como discutimos al inicio del capítulo las firmas de un mismo videoclip hechas con y sin cuadrantes no son iguales, ni siquiera en tamaño. Decidido entonces si se usará una implementación con cuadrantes o una sin ellos, nos encontramos con el detalle de que a pesar de que la idea ha sido concebida para trabajar con números binarios (1´s y 0´s) en la mayoría de los Sistemas Operativos como Windows o Mac OS la codificación por defecto para los archivos de tipo texto, como el usado para guardar la firma, es tal que cada caracter ocupa 1 byte, excepto algunos caracteres no visibles. En general, en todas las plataformas ocurrirá lo mismo si la codificación de caracteres del Sistema Operativo es de 8 bits. Por lo que, para realmente guardar la firma en bits necesitaríamos hacer un mapeo de la representación en la que se ocupa 1 byte por cada cero y uno en la firma a una representación donde realmente cada uno y cero se guardara en un bit. Lo cual no es algo imposible de hacer, bastaría con agrupar cada siete dígitos de la firma completa para luego buscar en una tabla ASCII como la mostrada en la Figura 3.12 el carater que corresponde a la representación de nuestro byte y guardarlo.

Análogamente un mapeo inverso se necesitaría para poder recuperar la firma original y cargarla en memoria.

Decimal	Hexadecimal	Binario	Octal	Caracter	Decimal	Hexadecimal	Binario	Octal	Caracter	Decimal	Hexadecimal	Binario	Octal	Caracter
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Figura 3.12: Código ASCII extendido.

Y aunque este proceso no representa gran dificultad, tampoco aporta nada significativo, debido a que el único propósito de dicho mapeo seria lograr que las firmas resultantes se guardaran en archivos con un bajo consumo de almacenamiento. Pero hemos visto que aún sin este mapeo los archivos de las firmas binarias no superan el 0.10% del total de espacio ocupado por el archivo de video, como puede verse en la Tabla 3.3, en donde hemos ordenado los datos de menor a mayor de acuerdo con la columna que expresa la relación entre el tamaño de la firma resultante y el video original. Por lo que, con la intención de obtener el mejor rendimiento posible, hemos decido no realizar este mapeo y trabajar con la representación de 1 byte por cada carácter que nos ofrece el Sistema Operativo.

Continuando con la toma de decisiones de la implementación y con el ánimo de aprovechar mejor el espacio de almacenamiento se optó por emplear una forma más compacta de la firma binaria, en la cual en lugar de guardar los *n bins* en *n* caracteres, solo guardaremos la posición del *bin* puesto en uno. Y dicha posición será guardada en el archivo que contenga la firma usando un número decimal. Esto debido a que guardar cualquier número del 0 al 9 (o cualquier carácter) ocupa el mismo espacio (1 byte) y tener nuestra firma binaria guardada de esta forma nos evita tener que hacer una consulta para saber cuál de los *bins* es el que está puesto en uno. La trasformación de la firma binaria esta ilustrada en la Figura 3.13.

Nombre Video	Tamaño Video (bytes)	Nombre Firma	Tamaño de la Firma (bytes)	Relación de tamaño con su video (%)
CP_11.mpg	15,362,570	Firma_CP_11.mpg.bin	1,496	0.009737954
CP_16.mpg	20,136,710	Firma_CP_16.mpg.bin	2,330	0.011570907
CP_15.mpg	13,439,498	Firma_CP_15.mpg.bin	1,560	0.011607576
CP_02.mpg	13,013,514	Firma_CP_02.mpg.bin	1,529	0.011749325
CP_08.mpg	13,519,370	Firma_CP_08.mpg.bin	1,593	0.011783093
CP_01.mpg	6,928,654	Firma_CP_01.mpg.bin	889	0.012830775
CP_13.mpg	11,160,038	Firma_CP_13.mpg.bin	1,465	0.013127195
CP_03.mpg	11,762,186	Firma_CP_03.mpg.bin	1,561	0.013271343
CP_10.mpg	14,125,578	Firma_CP_10.mpg.bin	1,883	0.013330428
CP_06.mpg	13,062,666	Firma_CP_06.mpg.bin	1,787	0.013680209
CP_18.mpg	12,679,690	Firma_CP_18.mpg.bin	1,769	0.013951445
CP_14.mpg	12,452,362	Firma_CP_14.mpg.bin	1,741	0.013981283
CP_21.mpg	11,663,882	Firma_CP_21.mpg.bin	1,660	0.014231968
CP_05.mpg	16,528,134	Firma_CP_05.mpg.bin	2,361	0.014284734
CP_23.mpg	11,508,234	Firma_CP_23.mpg.bin	1,657	0.014398386
CP_04.mpg	11,401,738	Firma_CP_04.mpg.bin	1,691	0.014831072
CP_22.mpg	5,978,382	Firma_CP_22.mpg.bin	921	0.015405506
CP_09.mpg	9,255,434	Firma_CP_09.mpg.bin	1,498	0.016185087
CP_07.mpg	9,919,004	Firma_CP_07.mpg.bin	1,626	0.016392775
CP_19.mpg	9,929,190	Firma_CP_19.mpg.bin	1,708	0.017201806
CP_17.mpg	5,202,190	Firma_CP_17.mpg.bin	1,016	0.019530236
CP_20.mpg	5,999,114	Firma_CP_20.mpg.bin	1,463	0.024386934
CP_00.mpg	12,169,738	Firma_CP_00.mpg.bin	3,123	0.025662015
CP_12.mpg	335,872	Firma_CP_12.mpg.bin	137	0.040789348

Tabla 3.3: Relación entre el tamaño de archivos de video y su firma binaria.

```
Firma Binaria (Completa)
0010000 0001000 1000000 0000000
Firma Binaria (Guardando de forma binaria el bin puesto en uno)
011     100     001     000
Firma Binaria (Guardando de forma decimal el bin puesto en uno)
3       4       1       0
```

Figura 3.13: Representación de la firma binaria para la implementación.

3.4.4. Encontrando la mejor coincidencia del comercial con Python

Dentro del alcance de este trabajo hemos especificado que el objetivo es encontrar coincidencias de pautas publicitarias en un archivo de video en donde previamente se ha grabado un día completo de transmisión (la duración de la grabación puede variar,

pero se tiene pensado que el videoclip para la búsqueda sea de tamaño menor o igual al video donde se buscaran coincidencias) teniendo también previamente grabado la pauta publicitaria que deseamos buscar. La forma de proceder a partir de tener estos dos archivos de video es obtener la firma binaria de cada uno y una vez teniendo ambas firmas ejecutamos una algoritmo recursivo de correlación entre ambas, usando la función de distancia de la que se habló en la Sección 2.4, el cual nos devuelve (si es que existe) las posibles posiciones donde se encuentra la pauta buscada en el tiempo de reproducción del video donde se realizó la búsqueda. Todos los experimentos realizados han devuelto resultados satisfactorios siempre y cuando ambas firmas fueran obtenidas usando los mismos parámetros.

El momento de reproducción devuelto puede llegar a tener desfase con el auténtico inicio de la pauta publicitaria, esto es inherente a la base que tomamos para el diseño de la firma: los I-Frames. Recordemos que el total de frames que componen un archivo de video se dividen en los llamados Grupos de Imágenes o GOP, cada uno de estos GOP solo contiene un I-Frame (para más detalle revisar la Sección 3.2), pero el frame inicial de la pauta buscada pudiera estar dentro de cualquier frame dentro del GOP o en cualquier frame en cualquiera de los GOP vecinos. El desfase de reconocimiento es la a distancia entre el frame inicial de la pauta publicitaria y la posición devuelta por el algoritmo y acorde a nuestros experimentos el desfase no supera dos segundos.

3.5. Construcción de un modesto sistema como parte final de la implementación

Para la construcción de nuestro sistema en Python usaremos el módulo Tkinter, el cual nos permite crear ventanas, cuadros de texto, botones, menús, etiquetas, listas, etc. Que son elementos con los cuales podemos crear una interfaz de usuario de una forma sencilla. En esta interfaz podemos colocar el código que tenemos ya hecho como respuesta a acciones del usuario. La pantalla inicial de nuestro sistema se ve como se muestra en la figura 3.14.

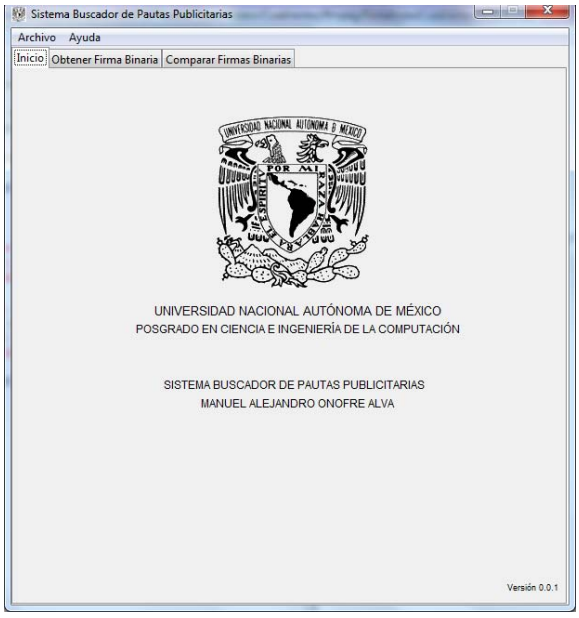

Figura 3.14: Pantalla inicial del sistema donde se implementa el trabajo realizado

Capítulo 4

Pruebas y Resultados

En éste capítulo se presentan las pruebas y resultados del reconocimiento de pautas publicitarias utilizando la técnica de firmas binarias descrita anteriormente.

4.1. Configuración de la Prueba

Para comenzar describiremos las características de Hardware y Software que se usaron durante el desarrollo de las pruebas realizadas.

4.1.1. Hardware

Las pruebas fueron realizadas en un equipo de cómputo con las siguientes características:

- Procesador Intel Celeron 900 a 2.2GHz como velocidad máxima.
- 6 GB de memoria RAM DDR3 a 1333 MHz.

4.1.2. Software

El software utilizado y las versiones se listan a continuación:

- Windows 7 Ultimate Service Pack 1 de 64 bits. El Sistema Operativo del equipo de cómputo.
- Python 2.7.5. Utilizado para la codificación de los algoritmos.
- OpenCV 3.1.0. Se usó para la adquisición y el tratamiento de los archivos multimedia.

- Numpy 1.8.0 para Python 2.7. El paquete fundamental para la computación científica sobre Python

- FFMPEG versión N-81322-ge8b355a. Utilizado para obtener los I-Frame de los archivos de MPEG y convertir a este formato los videos en otros formatos.

- GetGnuWin32 0.6.3. Utilizado para el procesamiento de los reportes obtenidos de FFMPEG.

- VLC Media Player 2.2.4. Se ocupó para obtener los archivos de video para las pruebas a través de IPTV.

4.1.3. Pruebas

Los archivos de video utilizados para las pruebas son MPEG (*.mpg) con:

- Tamaño de frame 720x480 pixeles.

- Velocidad de reproducción de 29 frames/segundo

Las pruebas se dividieron en dos partes, la primera es la obtención de la firma, en donde queremos evaluar el tiempo que nos toma firmar un video con diferentes configuraciones para poder decir que es una solución viable. La segunda parte consiste en utilizar las firmas binarias obtenidas de diferentes videos y videoclips para buscar coincidencias entre las mismas.

La primera parte, la obtención de la firma binaria se probó como sigue:

- Se obtuvo la firma binaria de comerciales de aproximadamente 20 segundos de duración cada uno con las configuraciones que se muestran en la Tabla 4.1.

Tipo de Configuración	Número de Colores	Obtener I-Frames	Divisiones en Cuadrantes
4-A	Cuatro	No	No
4-B	Cuatro	No	Si
4-C	Cuatro	Si	No
4-D	Cuatro	Si	Si
6-A	Seis	No	No
6-B	Seis	No	Si
6-C	Seis	Si	No
6-D	Seis	Si	Si
8-A	Ocho	No	No
8-B	Ocho	No	Si
8-C	Ocho	Si	No
8-D	Ocho	Si	Si

Tabla 4.1: Configuraciones para probar la obtención de firmas binaria

Para la segunda parte, la búsqueda de coincidencias en las firmas. Se usaron las firmas binarias obtenidas de diferentes videoclips cuyo contenido es una comercial de aproximadamente 20 segundos de duración, también las firmas de videos de una hora de duración donde sabemos que se encuentran comerciales específicos, para tener una validación inicial. Y finalmente las firma de videos que contienen la programación grabada durante una hora de un mismo canal de televisión donde desconocemos el contenido. Las firmas se probaron como sigue:

- Usando las firmas de los videoclips de 20 segundos: se seleccionó cada una y se buscó cuál de las firmas restantes en el conjunto tenía la distancia más corta a la firma seleccionada y se comparó el contenido de ambos videoclips.

- Usando las firmas de los videos de una hora de duración y las firmas de los comerciales que sabemos aparecen dentro de dichos videos: se buscó usando la firma del comercial el número de veces que aparecía dicho comercial dentro del video una hora y el tiempo de reproducción en el que aparecía. El número de apariciones y los tiempos eran datos conocidos y se esperaba que el sistema nos devolviera los mismos resultados o similares.

- Usando las firmas de los videos de una hora de duración y las firmas de comerciales, de los cuales no sabíamos si aparecían dentro de dichos videos: se buscó usando la firma del comercial el número de veces que aparecía del video una hora de video y el tiempo de reproducción. Estos valores eran desconocidos y los resultados obtenidos se verificaron posteriormente a la prueba.

4.2. Resultados

A continuación se muestran los resultados de las pruebas realizadas.

4.2.1. Firmando Pautas

Se obtuvieron las firmas con las configuraciones mostradas en la Tabla 4.1 de los videos del conjunto de prueba arrojando los resultados mostrados en la Tabla 4.2, los cuales son un promedio de los resultados individuales de cada conjunto de videos de prueba. De donde podemos ver que la configuración que en promedio toma más tiempo para obtener la firma binaria es la 8-B, la cual corresponde a usar 8 colores, no obtener los I-Frames del video y usar los cuadrantes. Por otro lado en promedio el menor tiempo se logra con cualquiera de las configuraciones tipo C, en las cuales se obtienen los I-Frames pero no se usan los cuadrantes. Independientemente de cuantos colores sean utilizados, las configuraciones 4-C, 6-C, 8-C obtuvieron en promedio el mismo tiempo. Las configuraciones tipo D también obtuvieron tiempos bajos y cercanos a las tipo C, sobre todo la configuración 6-D, la cual obtuvo un tiempo promedio igual a las tipo C. De esto podemos decir que el mayor impacto en cuanto a costo de tiempo de genera no cuando se agregan colores a la firma sino cuando el número de frames de video que se deben procesar es mayor, puesto que las configuraciones con los mejores resultados son en las cuales se han obtenido los I-Frames de los videos de prueba.

Tipo de Configuración	Tamaño del Video [bytes]	Tamaño Firma [bytes]	Frames Procesados para la Firma	Duración del Video [hh:mm:ss]	Tiempo en obtener la Firma [hh:mm:ss]
4-A	12,339,046	2,443	611	00:00:20	00:00:13
4-B	12,339,046	9,773	611	00:00:20	00:00:23
4-C	12,339,046	186	47	00:00:20	00:00:02
4-D	12,339,046	745	47	00:00:20	00:00:03
6-A	12,339,046	3,665	611	00:00:20	00:00:17
6-B	12,339,046	14,660	611	00:00:20	00:00:23
6-C	12,339,046	279	47	00:00:20	00:00:02
6-D	12,339,046	1,118	47	00:00:20	00:00:02
8-A	12,339,046	4,887	611	00:00:20	00:00:17
8-B	12,339,046	19,547	611	00:00:20	00:00:29
8-C	12,339,046	373	47	00:00:20	00:00:02
8-D	12,339,046	1380	47	00:00:20	00:00:03

Tabla 4.2: Promedio de los resultados de la primera parte de las pruebas.

4.2.2. Comparación entre las firmas

A continuación se describen las pruebas realizadas y los resultados específicos de cada una de ellas:

- Usando las firmas de los videoclips de 20 segundos: se seleccionó cada una de las firmas obtenidas utilizando las configuraciones en la Tabla 4.1 y se buscó cuál de las firmas restantes en el conjunto tenía la distancia más corta a la firma seleccionada y se comparó el contenido de ambos videoclips. Cada firma corresponde a un video diferente seleccionado según la Tabla 4.3

Tema del Video Comercial	Nombre firma	Abreviatura
Jugo de Fruta	FirmaBin1	FB1
Barra de Chocolate	FirmaBin2	FB2
Producto de Calzado tipo tenis	FirmaBin3	FB3
Revista para niños	FirmaBin4	FB4
Golosina agridulce	FirmaBin5	FB5
Aceite para cocina	FirmaBin6	FB6
Medicamento para prevenir la gripa	FirmaBin7	FB7
Medicamento para dolor de cabeza	FirmaBin8	FB8
Servicios Bancarios	FirmaBin9	FB9
Jarabe para la tos	FirmaBin10	FB10

Tabla 4.3: Temas de los videos usados en las pruebas

Los resultados parciales del total del conjunto de video de prueba usando la configuración 4-A se ilustran en la Tabla 4.4, donde se puede comprobar como la distancia más corta es cero y siempre es cuando la firma se compara consigo misma, la distancia promedio, para este caso, entre las firmas tiene un valor de 0.22.

4-A	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.20	0.23	0.21	0.16	0.21	0.23	0.34	0.30	0.16
FirmaBin2	0.20	0.00	0.24	0.24	0.18	0.17	0.20	0.31	0.27	0.15
FirmaBin3	0.23	0.24	0.00	0.25	0.22	0.24	0.26	0.29	0.30	0.22
FirmaBin4	0.21	0.24	0.25	0.00	0.24	0.29	0.23	0.44	0.42	0.19
FirmaBin5	0.16	0.18	0.22	0.24	0.00	0.11	0.21	0.27	0.25	0.13
FirmaBin6	0.21	0.17	0.24	0.29	0.11	0.00	0.22	0.19	0.24	0.16
FirmaBin7	0.23	0.20	0.26	0.23	0.21	0.22	0.00	0.31	0.34	0.16
FirmaBin8	0.34	0.31	0.29	0.44	0.27	0.19	0.31	0.00	0.23	0.30
FirmaBin9	0.30	0.27	0.30	0.42	0.25	0.24	0.34	0.23	0.00	0.28
FirmaBin10	0.16	0.15	0.22	0.19	0.13	0.16	0.16	0.30	0.28	0.00

Tabla 4.4: Distancias entre el conjunto de videos de prueba usando la configuración 4-A.

Los resultados parciales del total del conjunto de video de prueba usando la configuración 4-B se ilustran en la Tabla 4.5, puede verse de mono análogo a la prueba anterior como la distancia más corta siempre es cuando la firma se compara consigo misma, en este caso la distancia entre firmas tiene un valor promedio de 0.24.

4-B	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.22	0.26	0.26	0.20	0.23	0.27	0.34	0.31	0.25
FirmaBin2	0.22	0.00	0.26	0.28	0.21	0.20	0.24	0.31	0.27	0.23
FirmaBin3	0.26	0.26	0.00	0.30	0.25	0.24	0.31	0.30	0.30	0.29
FirmaBin4	0.26	0.28	0.30	0.00	0.30	0.31	0.29	0.43	0.42	0.29
FirmaBin5	0.20	0.21	0.25	0.30	0.00	0.15	0.27	0.26	0.26	0.21
FirmaBin6	0.23	0.20	0.24	0.31	0.15	0.00	0.26	0.20	0.24	0.20
FirmaBin7	0.27	0.24	0.31	0.29	0.27	0.26	0.00	0.31	0.34	0.25
FirmaBin8	0.34	0.31	0.30	0.43	0.26	0.20	0.31	0.00	0.22	0.29
FirmaBin9	0.31	0.27	0.30	0.42	0.26	0.24	0.34	0.22	0.00	0.31
FirmaBin10	0.25	0.23	0.29	0.29	0.21	0.20	0.25	0.29	0.31	0.00

Tabla 4.5: Distancias entre el conjunto de videos de prueba usando la configuración 4-B

Los resultados parciales del total del conjunto de video de prueba usando la configuración 4-C se ilustran en la Tabla 4.6, como en los casos anteriores se puede ver que la distancia más corta tiene un valor de cero y es cuando la firma se compara consigo misma, para esta configuración la distancia entre cada firma tiene un valor promedio de 0.22.

4-C	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.23	0.23	0.21	0.16	0.21	0.28	0.34	0.29	0.17
FirmaBin2	0.23	0.00	0.24	0.28	0.19	0.20	0.21	0.32	0.29	0.18
FirmaBin3	0.23	0.24	0.00	0.24	0.20	0.21	0.26	0.30	0.31	0.21
FirmaBin4	0.21	0.28	0.24	0.00	0.22	0.29	0.27	0.41	0.42	0.22
FirmaBin5	0.16	0.19	0.20	0.22	0.00	0.13	0.21	0.26	0.25	0.13
FirmaBin6	0.21	0.20	0.21	0.29	0.13	0.00	0.24	0.19	0.25	0.17
FirmaBin7	0.28	0.21	0.26	0.27	0.21	0.24	0.00	0.33	0.37	0.18
FirmaBin8	0.34	0.32	0.30	0.41	0.26	0.19	0.33	0.00	0.23	0.31
FirmaBin9	0.29	0.29	0.31	0.42	0.25	0.25	0.37	0.23	0.00	0.29
FirmaBin10	0.17	0.18	0.21	0.22	0.13	0.17	0.18	0.31	0.29	0.00

Tabla 4.6: Distancias entre el conjunto de videos de prueba usando la configuración 4-C

Los resultados parciales del total del conjunto de video de prueba usando la configuración 4-D se ilustran en la Tabla 4.7, igualmente que el resto de las configuraciones con cuatro colores, la distancia más corta se obtiene cuando la firma se compara consigo misma, en promedio la distancia entre cada firma es 0.25.

4-D	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.25	0.26	0.26	0.21	0.23	0.31	0.33	0.30	0.26
FirmaBin2	0.25	0.00	0.26	0.29	0.22	0.22	0.24	0.32	0.28	0.24
FirmaBin3	0.26	0.26	0.00	0.29	0.23	0.24	0.31	0.30	0.31	0.27
FirmaBin4	0.26	0.29	0.29	0.00	0.28	0.31	0.32	0.40	0.42	0.29
FirmaBin5	0.21	0.22	0.23	0.28	0.00	0.16	0.27	0.26	0.26	0.21
FirmaBin6	0.23	0.22	0.24	0.31	0.16	0.00	0.27	0.19	0.24	0.21
FirmaBin7	0.31	0.24	0.31	0.32	0.27	0.27	0.00	0.33	0.36	0.26
FirmaBin8	0.33	0.32	0.30	0.40	0.26	0.19	0.33	0.00	0.22	0.30
FirmaBin9	0.30	0.28	0.31	0.42	0.26	0.24	0.36	0.22	0.00	0.30
FirmaBin10	0.26	0.24	0.27	0.29	0.21	0.21	0.26	0.30	0.30	0.00

Tabla 4.7: Distancias entre el conjunto de videos de prueba usando la configuración 4-D

Los resultados parciales del total del conjunto de video de prueba usando la configuración 6-A se ilustran en la Tabla 4.8, en esta configuración con seis colores igualmente puede verse como la distancia más corta se mantiene en cero y solo se obtiene cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.23.

6-A	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.17	0.17	0.14	0.11	0.16	0.23	0.43	0.38	0.22
FirmaBin2	0.15	0.19	0.19	0.18	0.15	0.14	0.22	0.39	0.34	0.19
FirmaBin3	0.17	0.00	0.00	0.18	0.17	0.19	0.25	0.39	0.36	0.26
FirmaBin4	0.14	0.18	0.18	0.00	0.16	0.21	0.22	0.49	0.46	0.24
FirmaBin5	0.11	0.17	0.17	0.16	0.00	0.09	0.21	0.38	0.34	0.20
FirmaBin6	0.16	0.19	0.19	0.21	0.09	0.00	0.24	0.32	0.33	0.22
FirmaBin7	0.23	0.25	0.25	0.22	0.21	0.24	0.00	0.34	0.34	0.15
FirmaBin8	0.43	0.39	0.39	0.49	0.38	0.32	0.34	0.00	0.21	0.29
FirmaBin9	0.38	0.36	0.36	0.46	0.34	0.33	0.34	0.21	0.00	0.26
FirmaBin10	0.22	0.26	0.26	0.24	0.20	0.22	0.15	0.29	0.26	0.00

Tabla 4.8: Distancias entre el conjunto de videos de prueba usando la configuración 6-A.

Los resultados parciales del total del conjunto de video de prueba usando la configuración 6-B se ilustran en la Tabla 4.9, puede verse como la distancia más corta se da cuando la firma se compara contra sí misma, en promedio la distancia entre cada firma es 0.24.

6-B	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.17	0.19	0.18	0.14	0.17	0.25	0.41	0.36	0.26
FirmaBin2	0.17	0.00	0.20	0.21	0.17	0.16	0.24	0.37	0.32	0.24
FirmaBin3	0.19	0.20	0.00	0.21	0.18	0.18	0.27	0.38	0.36	0.28
FirmaBin4	0.18	0.21	0.21	0.00	0.20	0.22	0.26	0.47	0.44	0.29
FirmaBin5	0.14	0.17	0.18	0.20	0.00	0.12	0.25	0.35	0.33	0.24
FirmaBin6	0.17	0.16	0.18	0.22	0.12	0.00	0.25	0.30	0.31	0.23
FirmaBin7	0.25	0.24	0.27	0.26	0.25	0.25	0.00	0.33	0.33	0.20
FirmaBin8	0.41	0.37	0.38	0.47	0.35	0.30	0.33	0.00	0.21	0.29
FirmaBin9	0.36	0.32	0.36	0.44	0.33	0.31	0.33	0.21	0.00	0.29
FirmaBin10	0.26	0.24	0.28	0.29	0.24	0.23	0.20	0.29	0.29	0.00

Tabla 4.9: Distancias entre el conjunto de videos de prueba usando la configuración 6-B

Los resultados parciales del total del conjunto de video de prueba usando la configuración 6-C se ilustran en la Tabla 4.10, puede verse de igual forma como la distancia más corta siempre es cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.23.

6-C	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.19	0.18	0.16	0.14	0.17	0.25	0.42	0.35	0.22
FirmaBin2	0.19	0.00	0.19	0.22	0.15	0.17	0.23	0.40	0.34	0.22
FirmaBin3	0.18	0.19	0.00	0.18	0.16	0.17	0.24	0.39	0.35	0.24
FirmaBin4	0.16	0.22	0.18	0.00	0.17	0.22	0.24	0.47	0.44	0.25
FirmaBin5	0.14	0.15	0.16	0.17	0.00	0.11	0.22	0.38	0.33	0.20
FirmaBin6	0.17	0.17	0.17	0.22	0.11	0.00	0.25	0.31	0.33	0.22
FirmaBin7	0.25	0.23	0.24	0.24	0.22	0.25	0.00	0.36	0.34	0.16
FirmaBin8	0.42	0.40	0.39	0.47	0.38	0.31	0.36	0.00	0.23	0.30
FirmaBin9	0.35	0.34	0.35	0.44	0.33	0.33	0.34	0.23	0.00	0.26
FirmaBin10	0.22	0.22	0.24	0.25	0.20	0.22	0.16	0.30	0.26	0.00

Tabla 4.10: Distancias entre el conjunto de videos de prueba usando la configuración 6-C

Los resultados parciales del total del conjunto de video de prueba usando la configuración 6-D se ilustran en la Tabla 4.11, puede verse que la distancia más corta siempre es cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.24.

6-D	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.20	0.19	0.19	0.16	0.17	0.27	0.40	0.35	0.27
FirmaBin2	0.20	0.00	0.21	0.23	0.17	0.18	0.25	0.38	0.33	0.26
FirmaBin3	0.19	0.21	0.00	0.21	0.18	0.19	0.28	0.37	0.35	0.28
FirmaBin4	0.19	0.23	0.21	0.00	0.20	0.23	0.27	0.45	0.44	0.28
FirmaBin5	0.16	0.17	0.18	0.20	0.00	0.14	0.25	0.36	0.32	0.24
FirmaBin6	0.17	0.18	0.19	0.23	0.14	0.00	0.25	0.29	0.31	0.23
FirmaBin7	0.27	0.25	0.28	0.27	0.25	0.25	0.00	0.34	0.33	0.20
FirmaBin8	0.40	0.38	0.37	0.45	0.36	0.29	0.34	0.00	0.22	0.30
FirmaBin9	0.35	0.33	0.35	0.44	0.32	0.31	0.33	0.22	0.00	0.28
FirmaBin10	0.27	0.26	0.28	0.28	0.24	0.23	0.20	0.30	0.28	0.00

Tabla 4.11: Distancias entre el conjunto de videos de prueba usando la configuración 6-D

Los resultados parciales del total del conjunto de video de prueba usando la configuración 8-A se ilustran en la Tabla 4.12, puede verse claramente como la distancia más corta siempre es cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.17.

8-A	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.11	0.13	0.12	0.09	0.12	0.17	0.32	0.28	0.17
FirmaBin2	0.11	0.00	0.14	0.15	0.11	0.11	0.17	0.29	0.26	0.15
FirmaBin3	0.13	0.14	0.00	0.15	0.12	0.14	0.18	0.29	0.27	0.19
FirmaBin4	0.12	0.15	0.15	0.00	0.14	0.17	0.18	0.39	0.36	0.20
FirmaBin5	0.09	0.11	0.12	0.14	0.00	0.07	0.16	0.28	0.26	0.15
FirmaBin6	0.12	0.11	0.14	0.17	0.07	0.00	0.18	0.24	0.25	0.16
FirmaBin7	0.17	0.17	0.18	0.18	0.16	0.18	0.00	0.26	0.25	0.11
FirmaBin8	0.32	0.29	0.29	0.39	0.28	0.24	0.26	0.00	0.16	0.22
FirmaBin9	0.28	0.26	0.27	0.36	0.26	0.25	0.25	0.16	0.00	0.20
FirmaBin10	0.17	0.15	0.19	0.20	0.15	0.16	0.11	0.22	0.20	0.00

Tabla 4.12: Distancias entre el conjunto de videos de prueba usando la configuración 8-A.

Los resultados parciales del total del conjunto de video de prueba usando la configuración 8-B se ilustran en la Tabla 4.13, puede verse claramente como la distancia más corta siempre es cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.19.

8-B	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.13	0.14	0.14	0.11	0.13	0.18	0.30	0.27	0.19
FirmaBin2	0.13	0.00	0.15	0.17	0.13	0.12	0.18	0.28	0.24	0.18
FirmaBin3	0.14	0.15	0.00	0.17	0.14	0.14	0.21	0.28	0.27	0.21
FirmaBin4	0.14	0.17	0.17	0.00	0.16	0.18	0.20	0.36	0.34	0.23
FirmaBin5	0.11	0.13	0.14	0.16	0.00	0.09	0.19	0.27	0.25	0.18
FirmaBin6	0.13	0.12	0.14	0.18	0.09	0.00	0.19	0.23	0.24	0.17
FirmaBin7	0.18	0.18	0.21	0.20	0.19	0.19	0.00	0.25	0.25	0.15
FirmaBin8	0.30	0.28	0.28	0.36	0.27	0.23	0.25	0.00	0.16	0.22
FirmaBin9	0.27	0.24	0.27	0.34	0.25	0.24	0.25	0.16	0.00	0.21
FirmaBin10	0.19	0.18	0.21	0.23	0.18	0.17	0.15	0.22	0.21	0.00

Tabla 4.13: Distancias entre el conjunto de videos de prueba usando la configuración 8-B

Los resultados parciales del total del conjunto de video de prueba usando la configuración 8-C se ilustran en la Tabla 4.14, puede verse claramente como la distancia más corta siempre es cuando la firma se compara consigo misma, en promedio la distancia entre cada firma tiene un valor de 0.18.

8-C	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.14	0.13	0.13	0.10	0.13	0.19	0.31	0.26	0.16
FirmaBin2	0.14	0.00	0.14	0.18	0.11	0.13	0.17	0.30	0.25	0.16
FirmaBin3	0.13	0.14	0.00	0.15	0.12	0.13	0.18	0.29	0.26	0.18
FirmaBin4	0.13	0.18	0.15	0.00	0.14	0.18	0.19	0.36	0.35	0.20
FirmaBin5	0.10	0.11	0.12	0.14	0.00	0.09	0.16	0.29	0.25	0.15
FirmaBin6	0.13	0.13	0.13	0.18	0.09	0.00	0.18	0.24	0.24	0.16
FirmaBin7	0.19	0.17	0.18	0.19	0.16	0.18	0.00	0.27	0.25	0.12
FirmaBin8	0.31	0.30	0.29	0.36	0.29	0.24	0.27	0.00	0.17	0.22
FirmaBin9	0.26	0.25	0.26	0.35	0.25	0.24	0.25	0.17	0.00	0.20
FirmaBin10	0.16	0.16	0.18	0.20	0.15	0.16	0.12	0.22	0.20	0.00

Tabla 4.14: Distancias entre el conjunto de videos de prueba usando la configuración 8-C

Los resultados parciales del total del conjunto de video de prueba usando la configuración 8-D se ilustran en la Tabla 4.15, puede verse claramente como la distancia más corta siempre es cuando la firma se compara consigo misma en promedio la distancia entre cada firma tiene un valor de 0.18.

8-D	FB1	FB2	FB3	FB4	FB5	FB6	FB7	FB8	FB9	FB10
FirmaBin1	0.00	0.15	0.15	0.15	0.12	0.13	0.21	0.30	0.27	0.20
FirmaBin2	0.15	0.00	0.16	0.18	0.13	0.14	0.19	0.29	0.24	0.19
FirmaBin3	0.15	0.16	0.00	0.17	0.14	0.14	0.21	0.28	0.26	0.21
FirmaBin4	0.15	0.18	0.17	0.00	0.16	0.19	0.22	0.35	0.34	0.22
FirmaBin5	0.12	0.13	0.14	0.16	0.00	0.11	0.19	0.27	0.25	0.18
FirmaBin6	0.13	0.14	0.14	0.19	0.11	0.00	0.19	0.22	0.23	0.17
FirmaBin7	0.21	0.19	0.21	0.22	0.19	0.19	0.00	0.26	0.25	0.15
FirmaBin8	0.30	0.29	0.28	0.35	0.27	0.22	0.26	0.00	0.17	0.22
FirmaBin9	0.27	0.24	0.26	0.34	0.25	0.23	0.25	0.17	0.00	0.21
FirmaBin10	0.20	0.19	0.21	0.22	0.18	0.17	0.15	0.22	0.21	0.00

Tabla 4.15: Distancias entre el conjunto de videos de prueba usando la configuración 8-D

En todas las configuraciones probadas se obtienen resultados similares, lo cual nos indica que la configuración 4-A ha creado firmas que están lo suficientemente disjuntas para no confundirse entre ellas.

De esta prueba, además de comprobar experimentalmente que ninguna firma binaria tuviera una distancia menor con cualquiera de las otras firmas que la distancia que tiene consigo misma, se busca ver si las distancias entre las firmas nos podían dar indicios de una clasificación, es decir que los comerciales que fueran de temas relacionados o parecidos tuvieran distancias más cortas a comparación de los comerciales cuyo tema no tuviera relación. Y dentro de los experimentos existieron ocasiones donde parecía que esto ocurría, pero posteriores pruebas demostraron que no siempre funcionaba así. Según indica la Tabla 4.3 las firmas cuyo tema de video comercial es un medicamento son la

FirmaBin7, FirmaBin8 y FirmaBin10, tomando como ejemplo los resultados de la Tabla 4.16 podemos ver que la distancia entre la FirmaBin7 y FirmaBin8 tiene un valor de 0.26, la distancia entre la FirmaBin7 y FirmaBin10 tiene un valor de 0.15 y la distancia entre la FirmaBin8 y FirmaBin10 tiene un valor de 0.22. Los cuales son valores cercanos en comparación del promedio individual de cada firma. El promedio individual de las firmas FirmaBin7, FirmaBin8 y FirmaBin10 es respectivamente 0.19 ,0.24 y 0.18. Pero observando con más detalle la Tabla 4.16 podemos ver que firmas cuyo tema no es el de algún medicamento tienen también distancias cercanas a FirmaBin7, FirmaBin8 y FirmaBin10, incluso más cercanas como por ejemplo la FirmaBin9 cuyo tema es Servicios Bancarios tiene una distancia con la FirmaBin8 de 0.17 y es de hecho la menor distancia que tiene la FirmaBin8 con cualquiera de las otras firmas, análogamente es la menor distancia que la FirmaBin7 tiene con las demás firmas. Por otro lado mientras la pareja de firmas anteriores son una negación de la hipótesis de distancias más cortas entre videos con temas similares, la distancia entre FirmaBin7 y FirmaBin10 dan muestra de que esto si ocurre, teniendo la distancia menor con cualquiera de las otras firmas entre sí

Casos aprobatorios y desaprobatorios como los de las parejas de firmas anteriores aparecen por igual en los experimentos, por lo tanto como conclusión en esta prueba no aprobamos ni desaprobamos la hipótesis. Se requerirán más pruebas y resultados para poder tomar un juicio definitivo y dado que el tema de este trabajo va enfocado al reconocimiento de videos específicos y no a la agrupación por contenido, nos hemos limitado a hacer esta pequeña prueba. La hipótesis tiene cierta lógica, dado que las firmas están basadas en histogramas de color y usualmente los comerciales de temas parecidos usan escenas similares, pero esto no quiere decir que un comercial con un tema diferente no pueda usar también escenas que se asemejen a comerciales de otros temas.

- Usando las firmas con la configuración 6-D de la Tabla 4.1 de los videos de una hora de duración y las firmas, con la misma configuración 6-D, de los comerciales que sabemos aparecen dentro de dichos videos: se buscó usando la firma del comercial el número de veces que aparecía dicho comercial dentro del video una hora y el tiempo de reproducción en el que aparecía. El número de apariciones y los tiempos eran datos conocidos y se esperaba que el sistema nos devolviera los mismos resultados.

  Los tiempos devueltos y los tiempos reales de las apariciones del comercial buscado se muestras en la Tabla 4.16, donde solo se encontraron tres coincidencias para el comercial buscado, lo cual es correcto. Solo había tres comerciales como el buscado en el lapso de una hora que duraba el video de prueba, los tiempos no son exactos pero son buenas aproximaciones y ya hemos explicado que cuando su usan los I-Frames para obtener las firmas suelen ocurrir estos desfases.

Número de Coincidencia	Tiempo Devuelto	Tiempo Real	Desfase
1	00:15:50	00:15:45	00:00:05
2	00:34:22	00:34:14	00:00:08
3	00:45:12	00:45:03	00:00:09

Tabla 4.16: Resultados de la búsqueda de un comercial en un video de prueba.

El anterior es uno de los resultados de las pruebas realizadas, el resto de las pruebas obtuvo resultados similares, en promedio el desfase presentan los resultados devueltos es de alrededor de cinco segundos, lo cual a nuestro parecer es un resultado aceptable. Ya que si se está encontrando las coincidencias que se deben de encontrar, aunque existen falsos positivos estos son más frecuentes en videos donde no existen coincidencias del comercial, el sistema nos devuelve puntos donde la distancia entre las firmas es menor, para disminuir los falsos positivos agregamos un umbral para aceptar o no la coincidencia encontrada. El valor de dicho umbral se obtuvo experimentalmente y tiene un valor de 0.11. Recordando que las distancias están normalizadas para solo tener valores entre cero y uno, este umbral impide que firmas con una distancia de 0.11 o mayor sean aceptadas como coincidencias, ya que experimentalmente hemos comprobado que las coincidencias reales tienen distancias cercanas a cero y cuando se llegan a alejar del cero no sobrepasan el límite marcado por nuestro umbral.

- Usando las firmas de los videos de una hora de duración y las firmas de comerciales, de los cuales no sabíamos si aparecían dentro de dichos videos: se buscó usando la firma del comercial el número de veces que aparecía del video una hora de video y el tiempo de reproducción. Estos valores eran desconocidos y los resultados obtenidos se verificaron posteriormente a la prueba.

  Los tiempos devueltos y los tiempos reales de las apariciones del comercial buscado se muestras en la Tabla 4.17, donde se encontraron siete coincidencias para el comercial buscado, lo cual es correcto. Se comprobó solo había siete comerciales como el buscado en el lapso de una hora que duraba el video de prueba, los tiempos no son exactos pero son buenas aproximaciones y ya hemos explicado que cuando su usan los I-Frames para obtener las firmas suelen ocurrir estos desfases.

  Los resultados mostrados en la Tabla 4.17 son uno de los arrojados por las pruebas realizadas, el resto de las pruebas obtuvo resultados similares. En promedio el desfase que se presenta es de alrededor de cinco segundos, lo cual nos sigue pareciendo un resultado aceptable. Ya que si se está encontrando las coincidencias que se deben de encontrar y gracias a el umbral que hemos puesto para aceptar las coincidencias, el porcentaje de falsos positivos en videos donde no existen coincidencias del comercial es prácticamente cero.

Número de Coincidencia	Tiempo Devuelto	Tiempo Real	Desfase
1	00:21:12	00:21:10	00:00:02
2	00:24:24	00:24:20	00:00:04
3	00:34:18	00:34:12	00:00:06
4	00:39:09	00:39:03	00:00:06
5	00:47:09	00:47:02	00:00:07
6	00:49:40	00:49:32	00:00:08
7	00:51:50	00:51:42	00:00:08

Tabla 4.17: Resultados de la búsqueda de un comercial en un video de prueba.

Capítulo 5

Conclusiones y Trabajo a Futuro

En éste capítulo se presentan las conclusiones y el trabajo a futuro.

5.1. Conclusiones

Con base en los resultados obtenidos en el capítulo anterior se puede concluir lo siguiente:

- Los tiempos de reproducción devueltos por el algoritmo presentado no son 100% precisos, pero esto no quiere decir que el algoritmo sea incorrecto, la falta de precisión radica en que no estamos trabajando con la totalidad de los frames en los videos y además el desfase que existe entre el tiempo devuelto por el algoritmo y el real es un porcentaje lo suficientemente pequeño para servir a nuestros propósitos. En el caso de que la precisión de esto fuera significativo en la línea de investigación se podrían usar todos los frames, a costo de tiempo de procesamiento. Y si aun así la precisión no fuera lo suficientemente buena, entonces si se debe replantear el algoritmo.

- Como se demostró en las pruebas, la granularidad que nos ofrecen los cuadrantes puede aplicarse a hacer más exhaustiva la función de distancia, permitiéndonos evaluar la similitud entre los videos con una mayor profundidad, dicho esquema se diseñó para casos donde podríamos encontrar una misma imagen rotada y esto la hiciera diferente para los propósitos del reconocimiento, pero en la implementación se ha visto que obtener los cuadrantes no representa un costo computacional excesivo y podemos incluso recomendarlo para los casos donde no se esperan rotaciones en los videos.

- El reconocimiento presentado está basado en histogramas de color, por lo que si llegamos a encontrarnos con videos a los que se les han hecho transformaciones de color, como por ejemplo transformaciones en escala de grises, y la firma que tuviéramos como muestra fuera obtenida del video original, no sería posible efectuar tal reconocimiento. Este punto débil que

tienen todos los descriptores que se basan en histogramas de color no fue tomado en cuenta por el alcance del trabajo, pero somos conscientes de su existencia.

- Finalmente podemos decir que las firmas binarias han probado ser apropiadas para el reconocimiento de video específico, requieren poco espacio de almacenamiento y su obtención es relativamente rápida por lo que trabajar con ellas a nuestro parecer es una muy buena opción. En el trabajo presentado nos limitamos a buscar videos específicos pero las firmas binarias tienen capacidad para incluso ser usadas para buscar videos similares al video muestra o clasificar videos por su contenido.

5.2. Trabajo a futuro

Con base en los resultados obtenidos se plantea el siguiente trabajo a futuro:

- Todas las pruebas realizadas fueron hechas fuera de línea, es decir con videos grabados antes de iniciar los procesos de firmado y comparación. Como trabajo a futuro lograr un reconocimiento de video en línea siempre será una buena meta, ya que el trabajo actual está basado en tener por separado la captura del video y la clasificación del mismo.

- Investigar más a fondo si es viable la aplicación de las firmas binarias para encontrar videos de contenido similar, por ejemplo aplicando un algoritmo como K-medias sobre la distancia obtenida entre las firmas binarias, ya que en las pruebas realizadas no encontramos suficiente evidencia para aprobar o descartar esta idea.

- Aprovechar el reconocimiento obtenido por la firma binaria para filtrar los comerciales que no se deseen ver, aunque de momento esto sería fuera de línea y lo ideal sería conseguirlo en tiempo real.

- El trabajo presentado ocupa únicamente reducciones en el dominio del tiempo, esto debido a que se extrae el histograma de color de cada frame y después de eso no se vuelven a ocupar las imágenes que representan a los frames, en algunos trabajos [18] de CBVR (*Content-Based Video Retrieval*) antes de extraer las características de su interés de los frames hacen reducciones en dominios diferentes al temporal. Sería conveniente evaluar dicha alternativa.

Bibliografía

[1] C. K. L. a. W. Effelsberg, «On the detection and recognition of television commercials,» *Proc. of the IEEE Conf. on multimedia computing and Systems,* pp. 509-516, 1997.

[2] Y. L. a. C. Kuo, «Detecting commercial breaks in real TV programs based on audiovisual information,» *Proc of SPIE,* vol. 4210, pp. .225-236, 2000.

[3] L. Angihotri y N. Dimitrova, «Evolvable Visual Commercial Detector,» *Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03),* vol. 2, pp. 79-84, Junio 2003.

[4] J. Nafeh, «Method and Apparatus for Classifying patterns of Television Programs and Commercials Based on Discerning of Broadcast Audio and Video Signal». U. S. A. Patente 5, 343, 251, 30 Agosto 1994.

[5] INSTITUTO FEDERAL DE TELECOMUNICACIONES, MÉXICO, «Televisión Digital Terrestre,» 2015. [En línea]. Available: http://www.tdt.mx/. [Último acceso: 5 Enero 2015].

[6] A. D. Bimbo, Visual Information Retrieval, capítulo 3 y 4, Morgan Kaufmann Publishers, 1999.

[7] S. O. Belkasim, M. Shridhar y M. Ahmadi, Pattern Recognition with Moment Invariants: A comparative study and new results, in Pattern Recognition., vol. 24, 1991.

[8] M. J. Swain y D. H. Ballard, «Color indexing,» *International Journal of Computer Vision,* p. 11–32, 1991.

[9] J. C. Faimagne, Handbook of Perception and Human Performance,, vol. 1, Willey Interscience, 1986.

[10] G. Pass, R. Zabh y J. Miller, «Comparing Images Using Color Coherence Vectors,» *Proc. of the fourth ACM international conference on Multimedia,* pp. 65-73, 1996.

[11] L. Angihotri y B. Horn, Lisp, 2da ed., Addison-Wesley, 1984.

[12] D. S. Park, J. S. Park, T. Y. Kim y J. H. Han, «Image indexing using weighted color histogram,» *Proceedings of the 10th International Conference on Image Analysis and Processing,* p. 909–914, septiembre 1999.

[13] P. Hemnath y V. Prabhu, «Compression of FPGA bitstreams using improved RLE algorithm,» *International Conference on Information Communication and Embedded Systems (ICICES),* pp. 834 - 839, 2013.

[14] C. J. van den Branden Lambrecht, Vision models and applications to image and video processing, Boston: Kluwer Acedemic Publishers , 2001.

[15] M. Livingstone, «The First Stages of Processing Color and Luminance: Where and What,» *Vision and Art: The Biology of Seeing,* pp. 46-67, 2002.

[16] «OpenCV,» [En línea]. Available: http://opencv.org/. [Último acceso: septiembre 2015].

[17] «FFMPEG,» [En línea]. Available: http://ffmpeg.org/. [Último acceso: noviembre 2015].

[18] A. Cedillo-Hernández, M. Cedillo-Hernández, F. García-Ugalde, M. Nakano-Miyatake y H. Pérez-Meana, «An efficient content-based video retrieval for large databases,» *International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE),* pp. 15 - 19, 2015.