Universidad Michoacana de San Nicolás de Hidalgo

FACULTAD DE CIENCIA FISICO MATEMATICAS
"MAT. LUIS MANUEL RIVERA GUTIERREZ"

"Búsqueda de vídeos mediante una huella digital modificada."

Tesis que presenta
Augusto Alejandro Gutiérrez Reyes

para obtener el grado de
Licenciado en ciencias Físico Matemáticas

Asesor
Dr. Héctor Tejeda Villela

Coasesora
Dra. Karina Mariela Figueroa Mora

Morelia Michoacán
Febrero 2016

Índice general

Agradecimientos

Agradezco de primeras a mi familia, mis padres los cuales siempre está ahí y me apoyan en todo. Mi papá que también es mi amigo y del cual estoy muy orgulloso, de mi mamá la cual también quiero mucho y me ha enseñado muchas cosas de la vida y a mi hermano Alberto el cual es mi mejor amigo en este mundo y a la cual le he aprendido muchas cosas, principalmente nunca darse por vencido sea el problema que sea, a ustedes tres los quiero mucho. También quiero agradecer a mis abuelos, tíos y primos por brindarme su cariño y a los cuales llevo en mi corazón. Mis amigos de toda la vida Ivan y Daniel por estar siempre conmigo y saber que cuento con ustedes.

A mis grandes amigos de la preparatoria Ever, Fabela y Benjamín por compartir una amistad ya de 11 años más los que sigan, así como a mis otro gran grupo de amigos"los del cubo" Pelayo, Tero, Griego, Gaby, Bere, Yesenia, Ulises, Gabino y Lalo en orden de titulación para que no se me olvide, gracias a ustedes los del cubo por hacer todo este tiempo en la universidad más fácil y ponerme a estudiar cuando no quería o pellizcarme por cualquier razón, en verdad muchas gracias. A eli por ser una buena amiga y apoyarme cuando más lo necesitaba. También cómo no mencionar a los Hijos de Papa Pitufos tricampeones Hector, Beto, Toñito, Rafa, Rob y Ahtziri gracias por los buenos momentos en la cancha y en las chelas, a buscar otro campeonato.

También quiero agradecer a mis maestros que he tenido durante todos estos años de estudiado, desde mi profesor de matemáticas en la secundaria

Gabriel por decirme de la existencia de la facultad, como al profe Raul que me hizo sufrir en la prepa, pero que me dio buenas bases para el estudio de las matemáticas, a Joaquín que más que ser un profesor era un amigo y el cual siempre me apoyó en tanto académicamente como personalmente.

Agradezco a el profesor Héctor por todo este tiempo que me ha brindado, tanto como profesor como asesor en el desarrollo de la tesis.

Resumen

Con el presente trabajo de tesis se pretende obtener una caracterización de vídeos robusta, con la cual se pueda realizar una búsqueda de un vídeo contra otros previamente procesados para saber si este vídeo se encuentre o no en el conjunto antes mencionado.
Para realizar este procesos primero el vídeo pasa por una parte de procesamiento donde se le obtienen sus imágenes representativas de cada escena y con ellas se realiza la máscara con la cual se pretende realizar la búsqueda.
En la búsqueda se utilizan algoritmos de procesamiento de cadenas, para obtener un mejor resultado.
Palabras clave: Búsqueda, vídeos, huella, digital, modificada

abstract

In this thesis it is to obtain a characterization of robust videos, with which you can perform a search for a video against other previously processed to determine if this video is not situated within the aforementioned set.
To perform this process first video goes through a processing part where he obtained his representative images of each scene and with them the mask which is to search is performed.
In the search string processing algorithms are used to obtain a better result.

Capítulo 1

Introducción

En este trabajo se aborda el problema de búsqueda de un fragmento de vídeo en una colección de vídeos. El fragmento empleado para la búsqueda puede tener alguna modificación, que puede consistir en aplicar gotas de agua, agregar imágenes, incluir subtítulos, aplicar filtros o un escalamiento.

1.1. Justificación

El gran crecimiento de vídeos generados por los usuarios demanda una herramienta de búsqeuda y análisis, un ejemplo de esto es es la gran cantidad de videos que existen en youtube, nos llevaría aproximadamente 3600 años verlos todos. Con esta gran cantidad de videos se crean nuevos problemas tales como el *copyright* que consiste en la copia de un vídeo realizando modificaciones a este, y otro problema es la búsqueda en tiempo real de los vídeos. Atacar los dos problemas ha creado una tarea compleja, para esto se ha realizado distintas soluciones, unas de estas implementando soluciones basadas en texto que obtienen del vídeo mediante palabras clave, guiones, subtítulos, que se encuentran en los vídeos. Pero no todos los vídeos cuentan con este tipo de descriptores, por lo cual se han diseñado soluciones utilizando el contenido visual o auditivo del vídeo.
Con estos descriptores se ha mejorado la respuesta de búsqueda de los víde-

os, pero no el tiempo de ejecución, con esto en mente se trabajó en crear un algoritmo que compara un vídeo modificado contra otro para ver si son similares utilizando como descriptor del vídeo sus imágenes representativas y como motor de búsqueda el procesamiento de cadenas para mejorar el tiempo de ejecución.

1.2. Antecedentes

Los primeros intentos de atacar el problema se utilizaron la cantidad de tiempo de dicho vídeo como el descriptor, pero tenía un problema, Al haber una gran cantidad de vídeos pequeños, la detección era incorrecta, o mandaba múltiples respuestas que no reducían en nada el problema [6].

Con el problema anterior se empezó a buscar un identificador global de todo el vídeo, un primer intento fue tomar un fragmento del vídeo y tomar la cantidad de luz que contenía en sus imágenes, así como la cantidad de tiempo que dura el fragmento del vídeo. El problema con este procedimiento fue que al aplicar un filtro al vídeo, la cantidad de luz en éste se modifica, y al realizar la búsqueda nuevamente nos regresa falsos positivos [7].

Como podemos notar, el problema no es tan fácil de resolver, ya que implica la detección de imágenes que se encuentran en los vídeos. Se han realizado aproximaciones utilizando una firma binaria que tiene como datos el histograma de color de las imágenes del vídeo, pero este tiene sus fallas al incorporar gotas de agua o logotipos, ya que las toma como parte de la imagen a procesar [2].

1.3. Objetivos de la tesis

Entre los objetivos a seguir en la tesis se encuentra la realización de un descriptor característico del vídeo que sea lo suficientemente robusto como para notar una diferencia al compararlo con otros descriptores.

Otro objetivo que queremos alcanzar es la realización de un buscador de

vídeos. Se desea que procese un vídeo modificado, encuentre su descriptor característico y realice su búsqueda en una lista de vídeos ya procesados anteriormente, y así atacar el problema del *copyright*.

1.4. Planteamento del problema

Para determinar si un fragmento pertenece a un vídeo se utilizaron los siguientes etapas, las cuales preprocesan a los vídeos para obtener su descriptor.

Las caracterizaciones del vídeo son un subconjunto de imágenes, las cuales están agrupadas por bloques llamados escenas. Una escena es un conjunto de imágenes similares, que al pasar a una gran velocidad nos dan la sensación de ser sólo una pero con cierto movimiento. Así, un vídeo está formado por múltiples escenas.

El primer proceso que se aplica a un vídeo es la obtención de sus imágenes representativas o también llamadas *keyframes*. Una imagen representativa es aquella imagen que comparte una gran cantidad de información común con las imágenes vecinas. Como las escenas están compuestas por imágenes similares, se puede escoger arbitrariamente una imagen que represente a la escena.

Con el proceso descrito se representa un vídeo por un conjunto de imágenes, donde cada una de las imágenes representa una escena. El conjunto de imágenes representativas del vídeo se denominará máscara. El proceso anterior también permite reducir la magnitud del problema de búsqueda original, ahora las búsquedas se harán considerando solo un conjunto de imágenes.

A pesar de la reducción del problema de búsqueda, la comparación entre las diferentes imágenes de los vídeos y las del extracto todavía es computacionalmente costosa. Por lo cual se decidió transformar las imágenes representativas en una forma más compacta y simple, de tal manera que sea posible emplear algoritmos de búsqueda, como los empleados en el procesamiento de cadenas.

El siguiente paso será aplicar otro proceso que convierta la máscara obtenida compuesta por imágenes, a una máscara compuesta por caracteres. Debe ser robusta ante la gran cantidad de información que tiene cada imagen.

En cada elemento de la máscara se obtienen los puntos. Estos puntos de interés, también llamados *keypoints*, son pixeles donde se nota un cambio de iluminación y contraste en la imagen respecto de los pixeles vecinos. Para la obtención de estos *keypoints* se utilizó el algoritmo $SURF$[4], del cual se hablará más adelante.

Una primer estrategia de búsqueda de caracteres que consideramos fue sólo contar la cantidad de puntos de interés y con esto formar la palabra de búsqueda. Pero, ya al realizar pruebas nos dimos cuenta que ocupamos que fueran más robustas; ya que al aplicar las transformaciones a la copia del vídeo, la variación de la cantidad de *keypoints* en cada elemento de la máscara era muy variable.

En virtud del problema se optó por utilizar el centroide formado por los *keypoints* de cada imagen. Con dicho punto, y tomando el origen de la imagen calculamos el ángulo que forma con respecto al eje X, y usamos su valor para crear nuestra palabra de búsqueda representativa al vídeo o fragmento de consulta, compuesta por los ángulos de cada una de las imágenes de nuestra máscara inicial.

Para la realización de la búsqueda utilizamos el apareamiento aproximado de cadenas. Ya que, con las copias de vídeo, al realizar el proceso de la obtención de las imágenes representativas nos dimos cuenta que las modificaciones pueden generar o quitar imágenes. Éste error de edición, y otro error que se da al aumentar o disminuir los *keypoints* de una imagen, afectan al cálculo del centroide, que es con el cual se calcula el ángulo de la imagen y da origen al error de caracteres en la palabra de búsqueda.

1.5. Descripción de capítulos

En el capítulo 2 trataremos de la descripción de los algoritmos, así como de las herramientas matemáticas que utilizamos. Éstos estarán divididos en dos subcapítulos: Extracción de caracteres y búsqueda.

En la parte de extracción de caracteres hablaremos de los algoritmos de extracción de imágenes representativas, así como el *surf*, que es para la obtención de los *keypoints* de una imagen. Daremos una descripción del centroide y el ángulo que forma con respecto al origen, en el apartado de búsqueda se hablará del algoritmo de apareamiento aproximado de cadenas.

En el capítulo 3 se describirán de manera secuencial todo el procesamiento que se realiza para la obtención de la extracción característica de los vídeos, así como el procesamiento de búsqueda utilizado para obtención de resultados. También se dará paso a un subcapítulo que tendrá como contenido nuestros primeros acercamientos a la búsqueda de un buen descriptor del vídeo.

En el capítulo 4 se dirán con detalle los resultados obtenidos en la tesis para la búsqueda de vídeos con diferentes modificaciones realizadas, así como un comparativo entre todas ellas.

Las conclusiones obtenidas así como los trabajos a futuro que se desprenden de la tesis serán descritos en el capítulo 5.

Capítulo 2

Estado del Arte

En este capítulo daremos a conocer los distintos procesos utilizados para la detección de las copias de vídeo. Los cuales están divididos en tres aspectos:

- Extración de las características del vídeo.
- Proceso de búsqueda.

2.1. Extracción de las características del vídeo

Los vídeos están compuestos por una sucesión de imágenes a una alta velocidad, la cual nos transmite una sensación de movimiento.

El objetivo principal de la extracción de características es extraer imágenes distintivas pertenecientes al vídeo. Se utiliza este método de extracción por medio de imágenes y no por medio del audio, ya que existen una gran cantidad de vídeos a los cuales se les dobla el audio original o carecen de este.

A un conjunto de imágenes similares las podemos agrupar en un concepto utilizado en teatro llamado escena, que es la representación de un suceso en una misma locación con la interacción de los personajes, para nuestro trabajo utilizaremos este concepto para dividir la película en su número de

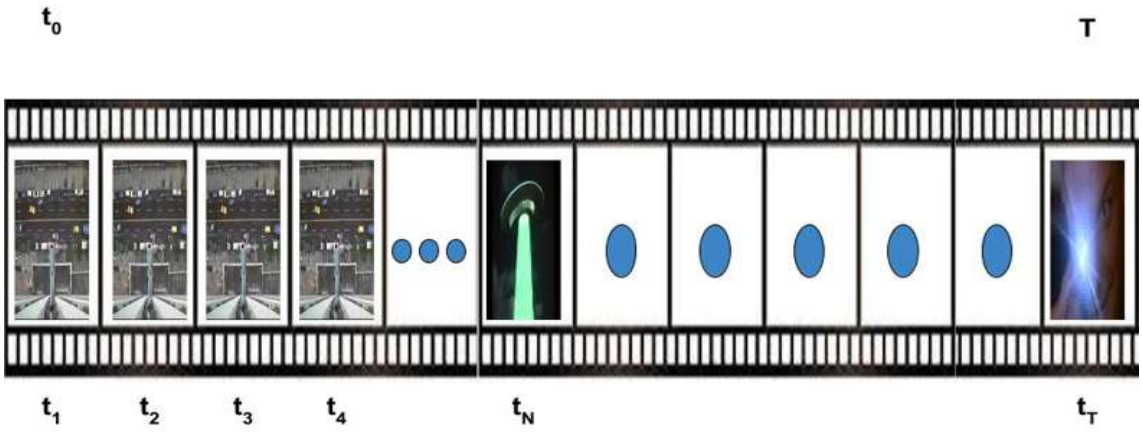

Figura 2.1: Sucesión de imágenes (frame) en alta velocidad (vídeo). Entre los tiempos t_0 y T existe un subconjunto sucesivo de estos que se llamará ventana (i.e. entre t_1 y t_N hay una ventna de tamaño N).

escenas, con el cual a cada una de estas escenas le obtendremos una imagen representativa.

Para esto antes tenemos que saber lo que es el modelo de color de una imagen ya que esto nos brinda la información característica de una imagen.

2.1.1. Modelos de color

Los modelos de color son representaciones matemáticas de los colores en una imagen de forma numérica, de esta forma convertimos una imagen en una matriz, donde cada entrada es el valor del color de cada pixel en la imagen, así facilitamos el procesamiento de las imágenes, existen varios tipos de estos modelos pero nosotros solo describiremos el RGB y el HSV.

Modelo RGB

El modelo RGB *(red, green, blue)*, representa los colores en 3 valores los cuales determina la intensidad de los colores primarios rojo, verde y azul los cuales corresponden a las triadas (1,0,0),(0,1,0) y (0,0,1) respectivamente. Este modelo lo podemos representar como un tetraedro donde los colores primarios se localizan en los vértices del tetraedro y la unión de cualesquiera dos de ellos nos genera un color secundario ya sea magenta, cian o amarillo. En el origen de este tetraedro obtenemos el color negro y en el punto más alejado al origen tenemos el color blanco (intersección de los tres colores primarios) esto lo podemos ver más claramente en la figura 2.2.

Modelo HSV

El HSV *(Hue, Saturation, Value)* también está representado por tres vectores los cuales indican la intensidad de los canales de matíz, saturación y valor*(Ih, Iv, Is)* respectivamente. El canal de matíz nos dice la pureza del color en la imagen. Es decir, la diferencia de los colores en la imagen, el de saturación nos dice que tanta cantidad de color blanco se encuentra y el canal de valor es inversamente proporcional a la cantidad de color blanco. Como

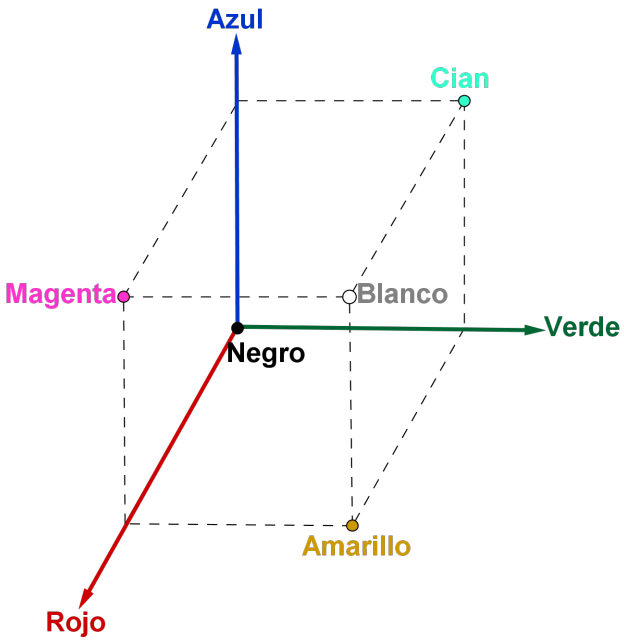

Figura 2.2: Modelo RGB

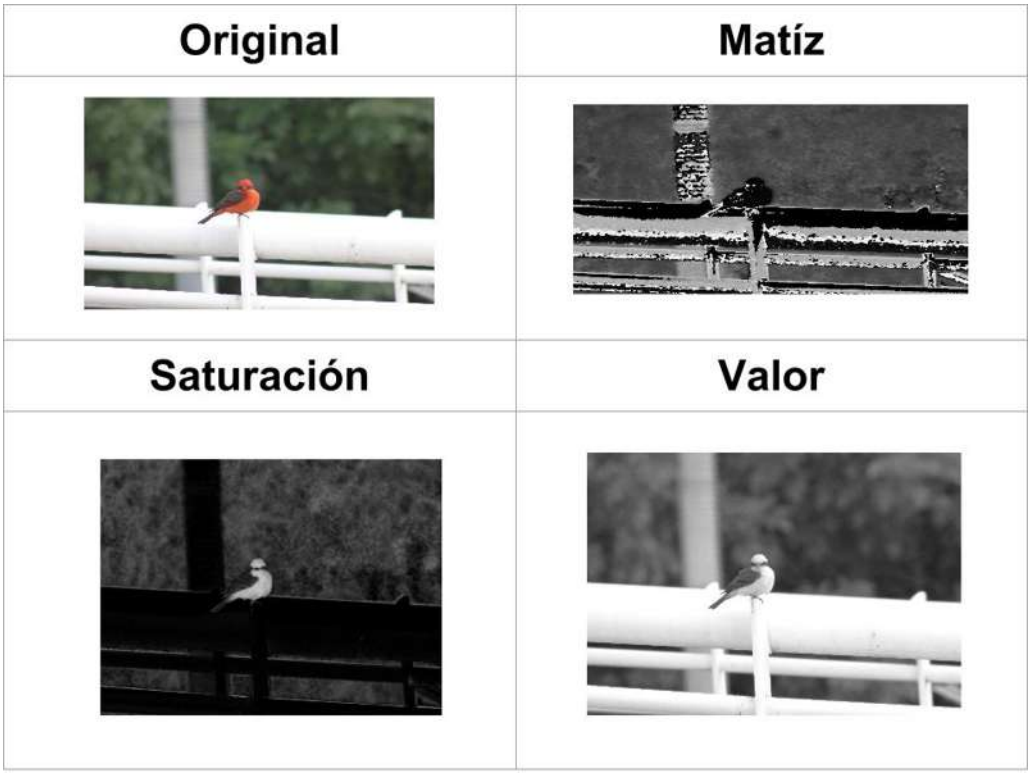

Figura 2.3: Imagen a la cual se le extrajeron sus canales de Matiz, Saturación y Valor.

se muestra en la figura 2.3 la obtención de los canales de matíz, saturación y valor respectivamente.

2.1.2. Cuadros representativos

Como anteriormente habíamos dicho que un vídeo está conformado por múltiples imágenes en una escena y que éstas podrían ser muy similares unas con otras, esto realiza que el costo de comparación de imagen a imagen sea muy alto, por eso es importante el seleccionado de solo una imagen representativa de la escena. Para esto utilizamos el algoritmo de extracción de

Figura 2.4: Imágenes representativas de una escena.

cuadros representativos.

Algoritmo de extracción de cuadros representativos.

En [1] se presenta un algoritmo utilizado para la extracción de imágenes representativas en un vídeo deportivo. Cada vídeo se descompone en una sucesión de imágenes, ver figura 2.1. cada imagen (o frame) se puede representar con un vector utilizando una descomposición de los canales HSV de ella misma, de la siguiente forma :

$$x^{t_i} = [h_H h_S h_V] \tag{2.1}$$

donde t_i representa la posición en esa sucesión. Note que los histogramas h_H, h_S, h_V cada uno contiene su respectiva información de intensidad I_H, I_S, I_V la cual nos proporciona el tamaño de los histogramas, *con tamaño* $L = I_H + I_S + I_V$.
Con estos vectores x^{t_i}, con $1 \leq i \leq N$ se puede formar una matriz tal que:

$$X = \begin{bmatrix} x^{t_1} \\ x^{t_2} \\ . \\ . \\ . \\ x^{t_N} \end{bmatrix} \tag{2.2}$$

Esta matriz la llamaremos ventana de tiempo.

El proceso de este algoritmo se basa en la descomposición de valores singulares de la matriz X. La descomposición en valores singulares para cada matriz X es la factorización de la matriz de la siguiente forma:

$$X = U\Sigma V^T \tag{2.3}$$

Donde U, V son matrices ortogonales y Σ es una matriz diagonal de X, además de Σ se obtiene el vector $\sigma_1, ..., \sigma_N$ el cual contiene los elementos de la diagonal de Σ de manera descendente. Note que estos son exactamente los valores singulares de X.

Se dice que r^N es el rango de X que se define como los valores singulares divididos por el máximo valor singular excedente a una cota γ ya definida. Es decir r^N es un el número de σ_i tal que:

$$\frac{\sigma_i}{\sigma_1} \geq \gamma \tag{2.4}$$

Para *i=1 ... N.* Para procesar el video completo la ventana debe desplazar en uno, para calcular los nuevos valores singulares y entonces encontrar su rango, es decir, la segunda vez el t_1 es ahora t_2 de la ventana anterior y t_N es t_{N+1}.

Mediante las pruebas se observó que si el rango de la matriz X_N es mayor que el de la matriz X_{N-1} (una ventana anterior) se concluye que las imágenes en el tiempo N son más semejantes entre sí que los de la ventana con tiempo *N-1*. Pero si el rango de X_N es menor que el rango X_{N-1} podemos decir que los cuadros en la ventana con tiempo igual a N tienen más información de la siguiente escena que los de la ventana con tiempo *N-1*.

Con estos resultados podemos decir que existe un cambio de escena cuando $X_N = 1$ y el rango de X_{N-1}es mayor que el rango X_N.

En el proceso del algoritmo se trabajo con ventanas de tamaño 12 y $\gamma = 0{,}25$, para disminuir un poco el tiempo de ejecución se trabajó solo con el canal de valor, ya que éste nos otorga una mejor cantidad de datos.

Para seleccionar la imagen representativa de la escena se tomó a t_i como el tiempo inicial de la escena y a t_f (No necesariamente es N) como el tiempo final de ella, con esto definimos como *keyframe* ó imagen representativa de la escena como:

$$keyframe = \frac{t_i + t_f}{2} \tag{2.5}$$

Para así obtener la imagen que se encuentra a la mitad de la escena, ya que experimentalmente es la que tiene mas información semejante con las demás imagenes de la escena.

Con esto podemos construir nuestra máscara de tamaño E, donde E es el número de *keyframes* que se obtienen por el vídeo y en cada una de las entradas se encuentra un *keyframe*.

2.1.3. Obtención de los puntos de interés de una imagen

Los puntos de interés son aquellos que son invariantes a rotaciones, escalamiento o cambios de iluminación en la imagen.

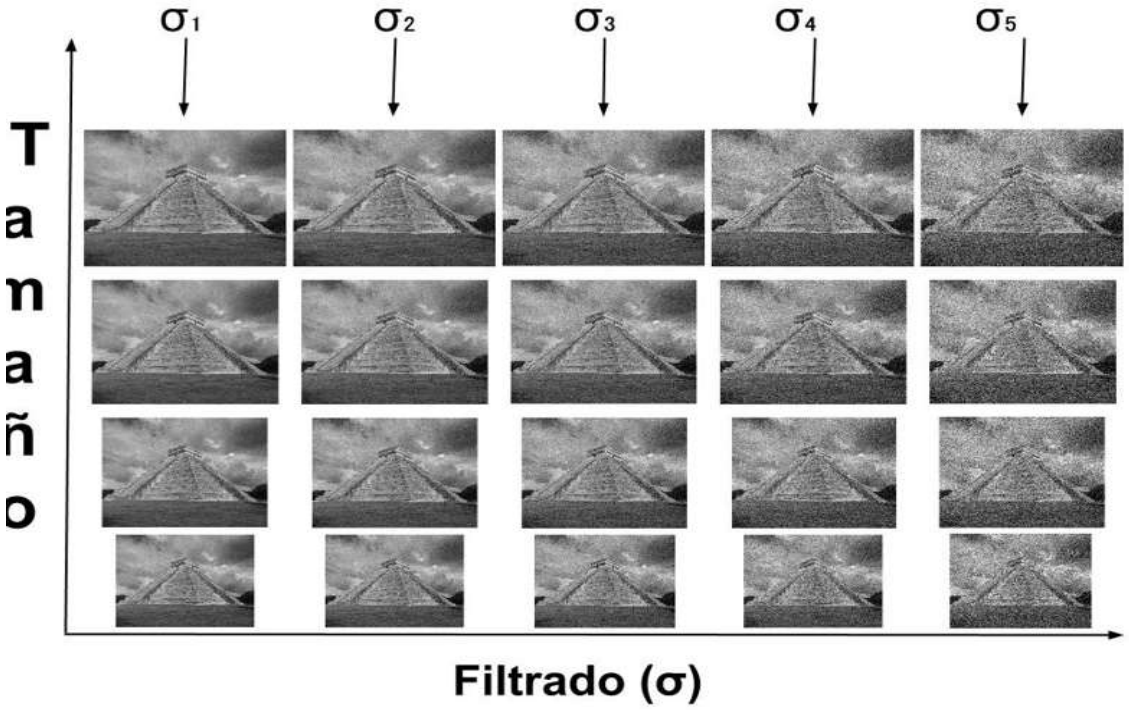

Figura 2.5: piramide gaussiana.

Detección de los puntos de interés utilizando el algoritmo *SURF*

Para la detección de los puntos de interés primero se localizan los máximos y mínimos locales y estos son considerados como los primeros puntos de interés. Para discretizar estos puntos se crea una pirámide formada por la imagen I filtrada con un parámetro σ pero a diferentes escalas, ver figura 2.5

Se llama octava a un conjunto de imágenes del mismo tamaño pero formadas con diferente convolución de una gaussiana con diferente valor σ. En cada nueva octava el filtro se incrementó usando el doble de su valor,tomando como patrón $\sigma_1 = 1{,}6$.

Para mejorar la obtención de los puntos de interés se utiliza la matriz hessiana, debido a la buena aproximación de la curvatura en la imagen. El

determinante de la matriz hessiana describe la escala de los puntos en la imagen. La hessiana $H(X, \sigma)$ de un punto $X = (x, y)$ con escala σ se define como:

$$H(x, \sigma) = \begin{pmatrix} L_{x,x}(x, \sigma) & L_{x,y}(x, \sigma) \\ L_{x,y}(x, \sigma) & L_{y,y}(x, \sigma) \end{pmatrix} \tag{2.6}$$

Dónde

$$L_{x,x}(x, \sigma) = I(x) * \frac{\partial^2}{\partial_{x^2}} g(\sigma) \tag{2.7}$$

$$L_{x,y}(x, \sigma) = I(x) * \frac{\partial^2}{\partial_{xy}} g(\sigma) \tag{2.8}$$

Dónde $L_{x,x}(X, \sigma)$ es la convolución de la derivada de segundo orden de la gaussiana en la imagen I con el punto X de igual manera con $L_{x,y}(X, \sigma)$ y $L_{y,y}(X, \sigma)$. Es muy costoso el calcular el determinante de la Hessiana $H(X, \sigma)$, por lo cual se utilizan aproximaciones $D_{x,x}$, $D_{x,y}$ y $D_{y,y}$ de la derivada de segundo orden de la gaussiana $G(\sigma)$ que mejoran el proceso de cómputo. Véase la figura 2.6

Así el cálculo del determinante aproximado de $H(\sigma)$ queda de la siguiente forma:

$$det(H(\sigma)) = D_{x,x} D_{y,y} - (\omega D_{x,y})^2 \tag{2.9}$$

El valor de ω varía sobre las escalas, pero se puede tomar de forma constante con un valor de 0.9.

Para determinar los puntos de diferentes escalas se toma la pirámide de imágenes, utilizando diferentes gaussianas para eliminar los puntos de interés menos útiles. las escalas se toman comenzando con filtros 9x9, luego 15x15 , 21x21 como se observa en la figura 2.7.

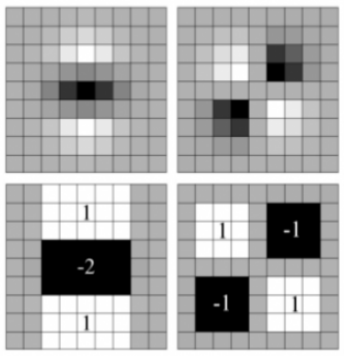

Figura 2.6: Las imagenes de arriba son las derivadas parciales de los términos $L_{y,y}(X,\sigma)$ y $L_{x,y}(X,\sigma)$de la gaussiana $G(\sigma)$ con sus respectivas aproximaciones $D_{y,y}$ y $D_{x,y}$.

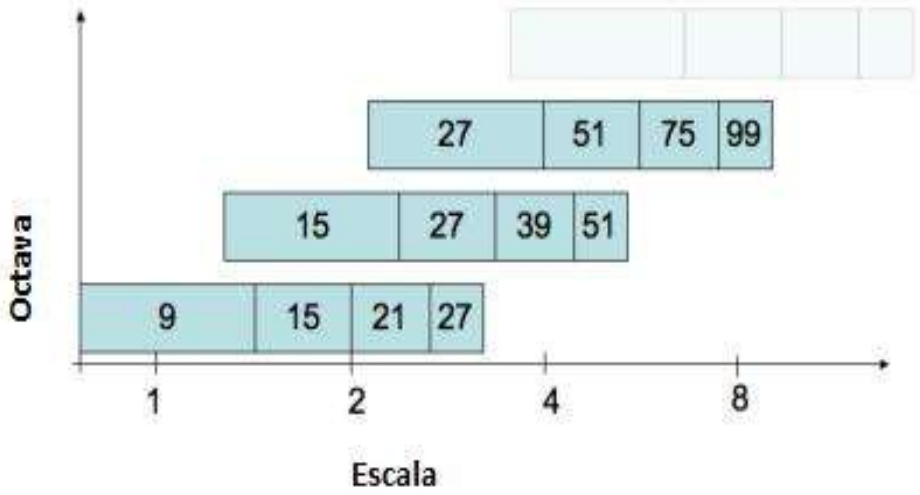

Figura 2.7: Escalas utilizadas en el algoritmo $SURF$.

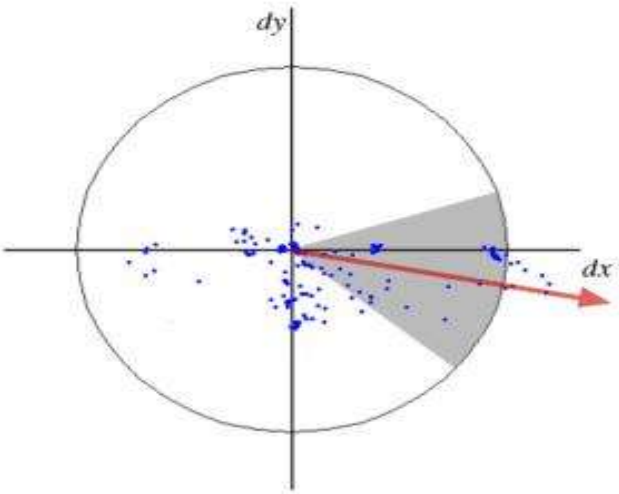

Figura 2.8: Orientación principal.

La invariante a rotaciones la obtenemos al calcular la orientación de cada punto de interés, para esto a cada punto se le determina una región circular de $6s$ donde s es la escala de la imagen, calculando las resultantes Haar-Wavelet de las direcciones x y y de los vecinos.

Para obtener la orientación principal se calcula la suma de las resultantes obtenidas en una ventana de $\frac{\pi}{3}$ como se muestra en la Figura 2.8.

En la región principal se obtiene una región circular de $20s$, la cual se dividirá en regiones de 4x4, calculando a cada región su resultante Haar-Wavelet para x y y, utilizando la gaussiana para el suavizado de los resultados y así obtener d_x y d_y.

Los descriptores están formados por un vector que guarda la suma de los resultados $\sum d_x$ y $\sum d_y$, y la suma de los valores absolutos $\sum \mid d_x \mid$ y $\sum \mid d_y \mid$. Véase figura 2.9

Para mejorar el *matching* entre los descriptores se utiliza el signo de la laplaciana para determinar el empate entre descriptores o no, sin ningún

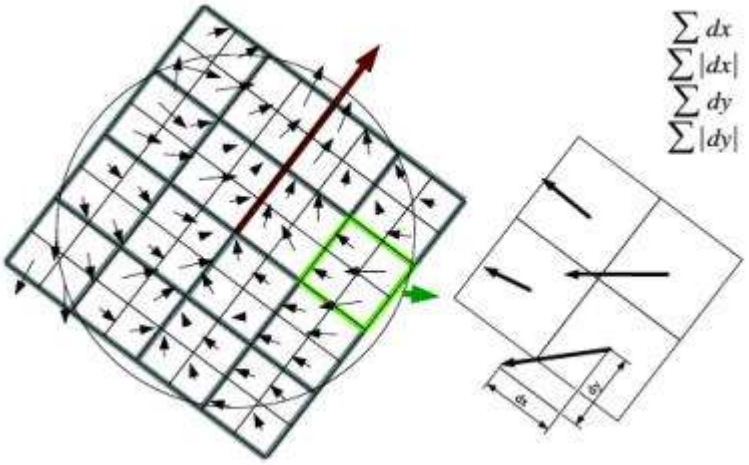

Figura 2.9: Cálculo de los descriptores en el algoritmo SURF.

costo adicional. Una de las principales características de SURF [4] es que al ser detectado un punto de interés siempre será detectado bajo distintas transformaciones.

2.2. Caracterización de la imagen

Con la obtención de los puntos de interés de cada imagen para tener una caracterización de esta, se utilizó como herramienta para calcular el centroide del polígono irregular que se forma con los puntos de interés de cada imagen.

El centroide de un polígono regular o irregular es el punto cuyas entradas X y Y corresponden a la media aritmética que se obtienen de las coordenadas de los vértices de dicho polígono. Figura 2.10

$$C_x = \frac{X_1 + ... + X_n}{n};\ C_y = \frac{Y_1 + ... + Y_n}{n} \tag{2.10}$$

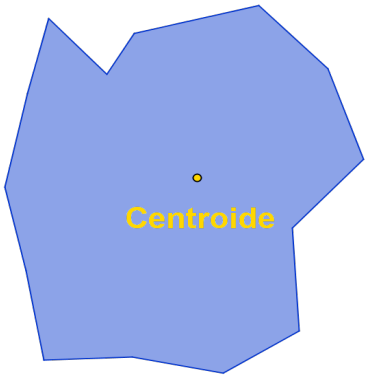

Figura 2.10: Centroide de un polígono irregular.

2.3. Algoritmo de búsqueda

En apareamiento aproximado de cadenas, el problema que resuelve es encontrar la ocurrencia de un patrón p en un texto T con un número limitado de errores K.[5]

Por otro lado la distancia de edición consiste en realizar operaciones ya sea: inserción, borrado y sustitución de un carácter en una cadena *S1* de largo n para convertirla en una cadena *S2* de largo m.

Entonces la distancia de edición para *S1, S2* es *ed(S1,S2)* es el número mínimo de operaciones que se utilizan para convertir a *S1* en *S2* ó viceversa. Ejemplo: *ed (annual,annealing)=4*.

Donde el apareamiento aproximado de cadenas se convierte en encontrar todas las ocurrencias en T de cada p' que satisfacen $ed(p,p') \leq K$. El problema tiene sentido si la $0<K<m$, de otra forma podríamos convertir cada

subcadena del texto de longitud p' en p sustituyendo los p' carácteres.

El caso con $K=0$ es cuando se busca el apareamiento exacto entre cadenas.

Para el cálculo de la distancia de edición se realiza de la siguiente forma:

Consideremos la Matriz $M : [\ 0\ldots \mid x \mid, 0\ldots \mid y \mid\ ]$, donde $M_{i,j}$ es el costo mínimo de convertir x_i en y_j.

Tomemos los carácteres x_i y y_j de las palabras X y Y respectivamente. Si $x_i = y_j$, entonces solo debemos convertir las subcadenas $x_{1\ldots i-1}$ a $y_{1\ldots j-1}$ con un costo $M_{i-1,j-1}$.

Sino utilizamos las operaciones permitidas:

- Sustituir x_i por y_j y convertir $x_{1\ldots i-1}$ a $y_{1\ldots j-1}$ con un costo $M_{i-1,j-1}+1$.
- Borrar x_i y convertir $x_{1\ldots i-1}$ a $y_{1\ldots j}$ con un costo $M_{i-1,j}+1$.
- Insertar y_j al final de $x_{1\ldots i}$ y convertir $x_{1\ldots i}$ a $y_{1\ldots j-1}$ con un costo $M_{i,j-1}+1$.

El insertar en una cadenas es igual que el borrado en otra cadena, por lo tanto estas tres operaciones cuestan lo mismo. El ejemplo anterior lo podemos ver en la figura 2.11

El tiempo que toma este algoritmo es de $O(\mid X \mid\mid Y \mid)$.

La búsqueda de un patrón p en un texto T es similar a la distancia de edición, lo único que se debe de permitir en este caso es que la ocurrencia pueda iniciar en cualquier parte del texto. lo anterior es hacer una modificación en la matrix $M_{0,j} = 0$ para toda $j \in 0...n$, esto nos dice que puede haber una ocurrencia en cualquier parte del texto con 0 errores porque es una subcadena de longitud 0.

Para optimizar el código en vez de trabajar con un matriz $M \times N$, se puede trabajar con una columna C_i donde se va a leer caráter por carácter

		a	n	n	e	a	l	i	n	g
	0	1	2	3	4	5	6	7	8	9
a	1	**0**	1	2	3	4	5	6	7	8
n	2	1	**0**	1	2	3	4	5	6	7
n	3	2	1	**0**	1	2	3	4	5	6
u	4	3	2	1	**1**	2	3	4	5	6
a	5	4	3	2	2	**1**	2	3	4	5
l	6	5	4	3	3	2	**1**	**2**	**3**	**4**

Figura 2.11: Cálculo de la distancia de edición de entre *.ªnnual"* y *.ªnnealing"*, el camino que se ve en negritas es el único camino óptimo, también se muestra el apariamiento de las palabras donde la línea punteada significa sustitución.

y se actualizará después de leer una nueva posición j de T.

$$C_i = M_{i,j}. \tag{2.11}$$

El algoritmo inicia la columna $C_{0...m}$ con los valores C_i a i y el proceso es carácter a carácter y se actualiza la columna cada vez que se lee un nuevo carácter t_j de la siguiente forma:

C'_i toma el valor de C_{i-1} cuando $p = t_j$. si no, entonces toma el valor de $1 + min(C_{i-1}, C'_{i-1}, C_i)$.

La posición del texto donde $C_m \leq k$ son reportadas como posiciones de ocurrencia.

En la figura 2.12 se muestra el proceso de programación dinámica permitiendo dos errores.

		a	n	n	e	a	l	i	n	g
	0	0	0	0	0	0	0	0	0	0
a	1	**0**	1	1	1	0	1	1	1	1
n	2	1	**0**	1	2	1	1	2	1	2
n	3	2	1	**0**	1	2	2	2	2	2
u	4	3	2	1	**1**	2	3	3	3	3
a	5	4	3	2	2	**1**	2	3	4	4
l	6	5	4	3	3	2	**1**	**2**	3	4

Figura 2.12: Muestra el algoritmo de programación dinámica para encontrar las ocurrencias entre el patrón *.ªnnual"* y el texto *.ªnnealing"* permitiendo dos errores de edición.

Para una mayor optimización utilizamos el algoritmo DP [3]. La idea básica de este algoritmo es el no calcular las celdas no activas. Una celda activa es aquella que su valor es a lo más K, el algoritmo lleva la cuenta de la última celda activa y evita el trabajo con celdas subsecuentes, esto se debe a que no puede haber mas de n incrementos de valor en el proceso.

En el algoritmo 1 se muestra el pseudocódigo del algoritmo DP.

Algorithm 1 Algoritmo DP

```
DP (p = p_1, p_2, ..., p_m, T = t_1, t_2, ..., t_n, K)
for i ∈ 0...m do C_i ← i
    uca ← k + 1        //última cadena activa
    for pos ∈ 1...n do
        pC ← 0, nC ← 0
        for i ∈ i...uca do
            if p_i = t_pos then
                nC ← pC
            else
                if pC < nC then nC ← pC
                end if
                if c_i < nC then nC = C_1
                    nC = nC + 1
                    pC ← C_i, C_1 ← nC
                end if
            end if
        end for
        while C_uca > K do uca ← uca − 1
            if uca = m then //Reporta una ocurrencia en pos
            else  uca ← uca + 1
            end if
        end while
    end for
end for
```

Capítulo 3

Desarrollo

En este capítulo se describe la implementación de los difrentes algoritmos descrito en el capítulo 2, los cuales realizan principalmente dos tareas, la caracterización de vídeos y la busqueda por contenido en una colección de vídeos.

3.1. Extracción característica de un vídeo

La implementación de los códigos para extraer la información de un vídeo se realizó en la plataforma de matlab, ya que permite una fácil manipulación tanto de vídeos como de imágenes, así como contener el paquete SURF para la extracción de puntos representativos de una imagen.

El algoritmo 2 procesa un vídeo para extraer sus imágenes representativas de cada escena, se crea un arreglo H de tamaño 12×256 donde se guardarán en cada uno de los renglones el cálculo del histograma del canal de valor para cada imagen que se encuentre en la ventana de tamaño 12.

Del arreglo H se calcula el vector de valores singulares V, El primer elemento del vector V se le asigna a la variable w.Las variables *rango* y *rangoant* son inicializadas en 1 y 0, respectivamente.Se extraerán los valores restantes del vector V una a uno y se dividirán entre w si estos son menores o iguales que 0.25, el *rango* incrementa en 1.

Si el $rango = 1$ y el $rangoant < rango$ se puede decir que existe un cambio de escena, sino el $rangoant = rango$ y se sigue recursivamente hasta terminar el tamaño del vídeo.

Algorithm 2 Extracción de *Keyframes*

```
N        N es la cantidad de Frames en una película.
H(12,256)        Se crea una Matriz de ceros.
frameint → 0, rangoant → 0
for i ← 1...N do
    b ← i + 11;
    for g → i...b do
        fram → imhist(g)        //Calcula el Histograma del canal de valor de la imagen g
        H → frame
        V → svd(H)        //Vector de valores singulares de H.
        w → V(1)
        for k → 1..,12 do
            if V(k) ≥ 0,25 then
                rango → rango + 1
            end if
        end for
        if rango == 1 then
        end if
        if rangoant>rango then        //Existe un cambio de escena.
            Keyframe → (frameint + 1)/2
        end if
    end for
    frameint → i, rangoant → rango
end for
```

Para obtener la imagen representativa en un cambio de escena se suma la posición de la escena inicial mas la posición de la escena final y se divide entre 2 para obtener la imagen que se encuentra justo en medio de la escena,

Figura 3.1: *Keyframes* consecutivos de un vídeo.

experimentalmente se notó que es la imagen que contiene información que caracteriza a dicha escena. Figura 3.1

Todos los *keyframes* resultantes se guardan en una carpeta con el nombre del vídeo procesado, el siguiente paso es extraer los *keypoints* de cada uno de los *keyframes* mediante el algoritmo *SURF*, el cual esta implementado en matlab.En la Figura 3.2 se muestra de color verde los *keypoints* hallados para la primera imagen de la figura 3.1.

Ya con la obtención de los *keypoints* utilizando el algoritmo 3 se procederá a calcular el centroide de estos puntos y calcular el ángulo que forma éste con el origen de la imagen. Figura 3.3

Para formar un cadena de tamaño igual al número de *keyframes* del vídeo, donde cada uno de los ángulos es una posición del arreglo y este se guardará en un archivo.txt. Tabla 3.1

0	29	0	14	28	23	16	**23**	**22**	**25**	**27**	**22**	**19**	17	27	17	26	20	21	25

Figura 3.2: Extracción de *keypoints*

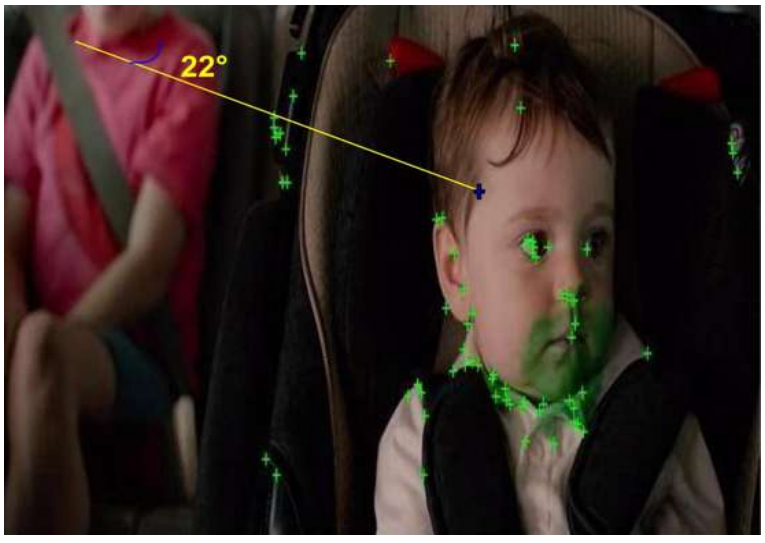

Figura 3.3: Ángulo generado por el centroide y el origen de la imagen.

Tabla 3.1: Cadena C del vídeo V, donde los ángulos en negritas son los correspondientes a las imágenes de la figura 3.2 .

Algorithm 3 Extracción de cadena.

list Arreglo con los keyframes de una película.
$l \rightarrow length(list)$ //Tamaño del arreglo.
for $i \leftarrow 1...l$ **do**
  $g \leftarrow i, X \rightarrow 0, Y \rightarrow 0;$
  $S \rightarrow imhist(g)$ //Calcula el Histogrma del canal de valor de la imagen g
  $P \rightarrow detectSURF(S)$ //Número de *keypoints* en la imagen S.
  for $k \rightarrow 1...P$ **do**
    $x \leftarrow strongest.Location(k, 1)$
    $y \leftarrow strongest.Location(k, 2)$ //Extraccción de las cordenadas x, y del punto k.
    $X \leftarrow X + x, Y \leftarrow Y + y$
  end for
  $Cx \leftarrow X/p, Cy \leftarrow Y/p$
  $angulo \leftarrow atan((Cy, Cx))$
end for
dlmwrite('Cadena.txt', ángulo , ' ') //Crea la cadena con la información de los ángulos.

Todo este proceso es para convertir un vídeo en una cadena de palabras formada en cada entrada con el ángulo del centroide con el origen. Como la obtención de los *keyframes* con el algoritmo SURF es invariante contra escalamiento y rotaciones nos permite decir que el ángulo que se obtiene es robusto para considerarlo como un dato que representa al keyframe al que pertenece así como a la escena.

3.2. Búsqueda de una cadena en el texto.

Con la realización de convertir vídeos en cadenas representativas de este, se procederá a tratar el problema de reconocimiento de vídeos, como un problema de apareamiento múltiple de cadenas.

Enseguida se hablará de las primeras ideas que se tuvieron para hacer el indexado y la búsqueda de los vídeos.

3.2.1. Aproximaciones

Antes de utilizar el ángulo para la creación de las cadenas, se crearon dichas cadenas con la cantidad de *keypoints* que contenía cada keyframe. Y se procedió a hacer una búsqueda con apareamiento aproximado de cadenas.

Para nuestras primeras búsquedas se utilizaron fragmentos de una película de aproximadamente 30 a 45 segundos, sin ningún tipo de modificación. La cantidad de *keypoints* encontrados en un *keyframe* varía desde 0 hasta un aproximado de 1500 *keypoints*, lo cual nos dice que la probabilidad de encontrar un par consecutivo de entradas en la cadena C iguales es muy pequeña y con esto se construyó la cadena del vídeo.

Al realizar la búsqueda contra un conjunto de películas ya indexadas se obtuvieron buenos resultados ya que al ser la búsqueda exacta de la cadena del vídeo prueba contra las cadenas de las películas indexadas, se reportaba un matching solo en la película a la cual se le había extraído el vídeo.

Ahora uno se pregunta qué pasaría si al vídeo se le aplica un filtro, osea una modificación a su contenido visual ¿seguirá reportando una ocurrencia?.

Para esto se realizaron modificaciones al vídeo prueba con diferente tipo de filtrajes, Figura 3.4 muestra los diferentes tipos de filtraje o modificaciones que se utilizaron.

El primer problema que nos encontramos al extraer la información representativa del vídeo es que se agregan o quitan *keyframes* esto representa una variación en el tamaño de la cadena o también llamado error de edición, ya que al realizar la búsqueda no generaba *matchings* en lugar correspondiente por la modificación en la longuitud de la cadena, Figura 3.5

El siguiente problema que se detectó es que la cantidad de *keypoints* entre un keyframe del vídeo modificado y su similar en el vídeo original tienen una gran variación. Figura 3.6

Por lo cual las pruebas con estos vídeos modificados no dieron buenos

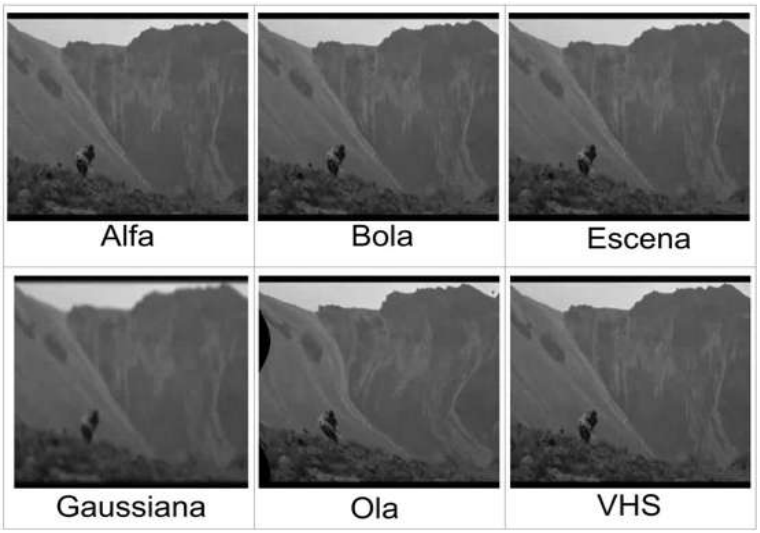

Figura 3.4: imagen con diferentes filtros.

Figura 3.5: *Keyframes* del vídeo original y de uno modificado con efecto gaussiano.

Figura 3.6: Diferentes cantidades de *keypoints* en las imágenes con distintos filtros.

resultados al aplicar la búsqueda ya que generan múltiples *matchings* con distintas películas.

3.2.2. Aplicación del algoritmo DP

Para atacar los problemas detectados en nuestras primeras búsquedas se consideró trabajar con algoritmos de programación dinámica más en particular con el algoritmo DP, ya que nos permite trabajar con el error de edición de la cadena C que se desea buscar en un texto T y con modificación en el algoritmo podemos adaptar el código para que también considere el error de carácteres.

El error de carácteres es el rango entre el cual vamos a considerar que existe una relación de similitud entre los carácteres. Este se consideró al ver la variación entre la cantidad de *keypoints* de un keyframe original y su similar modificado. La cual se intentó aproximar por un rango constante de error así como por medio de una ecuación lineal o exponencial, sin poder tener

buenos resultados en la búsqueda. Por lo cual fue que se considero buscar otro parámetro más robusto para realizar la búsqueda, este parámetro es el ángulo ya descrito anteriormente.

Con la consideración del ángulo como nuestro parámetro robusto realizamos la búsqueda de estas nueva cadena C contra nuestra colección de películas. Obteniendo de forma experimental que el error de edición como el error de carácteres lo consideraremos con valor de 2.

Capítulo 4

Resultados

En este capítulo se presentan los resultados de las búsquedas de vídeo por contenido en una colección de vídeos, con la finalidad de validar experimentalmente los modelos que están planteados en este trabajo. Los fragmentos de vídeos que fueron empleados para realizar las consultas se usaron de las siguientes formas:

- sin modificación alguna,
- con modificación ligera, y
- con modificación fuerte.

Enseguida se detallan las diferentes modificaciones que les fueron aplicadas a los vídeos de consulta. Se usaron fragmentos de vídeo con una duración aproximada entre 30 y 45 segundos. Del conjunto de escenas de vídeo que forma al fragmento, se decidió descartar la primera y la última escena, debido a que la duración de estas escenas generalmente varía respecto a las escenas originales, y por lo tanto, la imagen representativa de la escena no coincidiría con la de los vídeos de consulta. En la figura 4.1 se puede apreciar como las imagenes representativas de la primera y última escena difieren.

Figura 4.1: Se muestra la diferencia de los *kyeframes* iniciales da cada uno de los vídeos.

4.1. Vídeos sin modificación

En las primeras pruebas que se realizaron se utilizó un fragmento de una película, al cual no se le realizó ninguna modificación. al fragmento de vídeo, utilizado para la consulta, se le aplican los algoritmos descritos en el capitulo 3,la obtención de sus imágenes representativas, así como el cálculo del centroide de cada una de ella para formar la máscara de búsqueda representativa para este vídeo.

Cuando se realiza una búsqueda de un fragmento de vídeo sin modificación, se logra encontrar la cadena que representa al segmento en la cadena del vídeo del cual se obtuvo el fragmento. También se probó buscar la cadena de un fragmento de vídeo que no hubiese sido extraído de la colección de vídeos, y no se logró aparear la cadena con alguno de la colección. En la tabla 4.1 se muestra una tabla de 5 fragmentos de vídeo, en la cual se indica la cantidad de puntos de interes y el centroide de esos puntos para las imágenes

representativas de cada escena.

4.2. Vídeos con una ligera modificación

En esta sección se describen los resultados al modificar los fragmentos de vídeo con una modificación ligera. Una modificación ligera, definida cualitativamente, es aquella en donde el ojo humano casi no percibe diferencia, o bien, cuando la modificación es sólo en una región pequeña de la imagen, como lo hace el efecto onda, las modificaciones realizadas a los vídeos de efectuaron en el programa vlc[8], se utilzó este programa por su fácil manejo así como contar con una grán cantidad de filtros aplicables a los vídeos y su fácil extracción.

Los filtros utilizados para estas pruebas fueron los siguientes:

- Efecto borroso. Este efecto le da una difuminación, por lo que el vídeo parece estar desenfocado.Se utilizó un parárametro de factor borroso de 80 en una escala de 1 a 127.

- Efecto pantalla azul. El efecto mezcla las partes azules del primer plano de la imagen con el fondo.Los parametros utilizados fueron Valores U y V bluescreen 120 y 90 respectivamente, y la tolerancia en la pantalla azul tanto para U y V es de 17.

- Efecto de película vieja. Provoca una distorción ligera en el vídeo, por lo que el vídeo parece tener un suavisado en la imagen. los parametros modificados son el valor de suavizado y el tamaño de la ventana del vídeo, estos dos parametros son aplicados en 10 fotogramas por segundo durante la duración del vídeo.

- Efecto de onda. Distorsiona el vídeo en su parte inferior en forma de onda descendente. El parametro modificado la cantidad de pixeles distorcionados tomando un 20 porciento de la imagen de abajo hacia arriba.

Vídeo 1												
Keyframes	1	2	3	4	5	6	7	8	9	10	11	12
Puntos de interés	630	200	239	925	719	467	75	371	897	328	240	312
Centroide	17	19	22	19	18	22	25	21	22	19	19	18
Vídeo 2												
Keyframes	1	2	3	4	5	6	7	8	9	10	11	12
Puntos de interés	144	241	258	167	108	92	456	175	488	253	436	241
Centroide	29	23	23	32	45	37	28	26	23	31	30	28
Vídeo 3												
Keyframes	1	2	3	4	5	6	7	8	9	10	11	12
Puntos de interés	442	49	316	146	71	198	323	96	100			
Centroide	20	30	21	25	14	23	30	18	20			
Vídeo 4												
Keyframes	1	2	3	4	5	6	7	8	9	10	11	12
Puntos de interés	378	81	169	136	181	176	79	152	191	132	130	117
Centroide	26	34	27	27	29	32	32	34	33	32	25	32
Vídeo 5												
Keyframes	1	2	3	4	5	6	7	8	9	10	11	12
Puntos de interés	269	134	231	210	74	37	33	670	122	528	368	
Centroide	27	11	26	19	20	19	24	20	18	19	24	

Cuadro 4.1: Se muestra un comparativo de vídeos a los cuales se le extrajeron sus keyframes y se calcularon sus puntos de interés, así como su centroide.

- Efecto de grano. El efecto añade ruido gaussiano diferido al vídeo, esto provoca en el vídeo un efecto pimienta.Los parametros modificados para realizar este efecto fueron con un valor de 2 en la cantidad de ruido gaussiano y un grosor de 3 para el tamaño del grano.

Al realizar el proceso de búsqueda, se utiliza un error en el rango del ángulo formado por el eje x y el centroide, este es causado por la incorporación de *keypoints* en los *keyframes* y así afectando el calculo del centroide, al cual le daremos un valor igual a dos, este mismo valor se le dará al error de edición, que es el error casado al añadir el filtro al vídeo provocando que se añadan o quiten *keyframes*. Utilizando los vídeos con los diferentes filtros encontramos que existen ocurrencias en más de una película pero con una cantidad de error distinta entre ellas. Esto quiere decir que la información que se tiene en la máscara para el patrón de búsqueda es similar en diferentes películas. Para dar una solución a esto se tomó como respuesta la que tenga el error mínimo en tanto en el error de edición como en el error del centroide y en caso de existir un empate se muestran las dos como posibles ocurrencias para el patrón de búsqueda dado que la información que comparten dichas películas es muy similar y con el proceso descrito anteriormente no se puede decir a cuál de ellas pertenece el fragmento de vídeo procesado.

Lo que sí podemos asegurar es que entre estos posibles resultados se encuentra la película a la cual pertenece el fragmento del vídeo.Figura 4.2

4.3. Vídeos con modificaciones fuertes

Una modificación fuerte es aquella que al aplicarla a un vídeo este se distorsiona de tal manera que es muy notorio para el ojo humano, algunos errores son el filtro sepia que cambia toda la imagen a un tono más cálido, así como el reflejo de la luz natural o artificial, movimiento en la imagen o la unión de varios de estos errores, estos tipos de modificaciones causan un

Figura 4.2: Imagen con diferentes filtros a la cual se le calculó sus puntos de interés.

gran problema en el algoritmo planteado, ya que la información obtenida de ellas es muy diferente a la que se obtiene con el mismo vídeo pero sin ninguna modificación.
En particular nos enfocamos con vídeos que fueron tomados con la cámara de un celular directamente del vídeo original para buscar una posible apareamiento, los errores encontrados en este tipo de vídeos son el movimiento por las pulsaciones, el reflejo de la luz artificial, el contraste y brillo que tiene el monitor así como la calidad con la cual graba la cámara, con tal cantidad de errores al realizar el proceso nos dimos cuenta que se obtienen múltiples apareamientos en distintas películas, de igual forma que con los vídeos con una modificación ligera se puede dar como respuesta la que tenga la menor cantidad de errores pero en caso de existir un empate no se puede dar una respuesta concreta, la diferencia es que en este proceso no se puede asegurar que siempre se encuentra a la película a la cual pertenece el fragmento, ya que durante las pruebas en pocas ocasiones no se encontró con un apareamiento en la película a la cual pertenece el vídeo.

Para los demás casos de modificaciones fuertes los vídeos tenían problemas desde el procesamiento de la obtención de los *keyframes*, por ejemplo, un fragmento de vídeo no modificado tiene catorce *keyframes* y cuando se le aplica el filtro sepia a ese fragmento de vídeo solo se obtiene un *keyframe* debido a que el filtro uniformiza la mayoria de los colores, por lo que hay un contraste pobre de luz entre cambio de escena, entonces con la muy poca información obtenida con este tipo de filtro no se puede realizar un proceso de búsqueda con el 4.3.

Con los resultados obtenidos, podemos concluir que la utilización del algoritmo implementado en este trabajo nos brinda buenos resultados para la búsqueda de vídeos con modificaciones ligeras o sin modificación, pero para vídeos con modificaciones más fuertes el algoritmo puede ser un buen filtrador para la búsqueda de vídeos, con esto minimizar la cantidad de vídeos

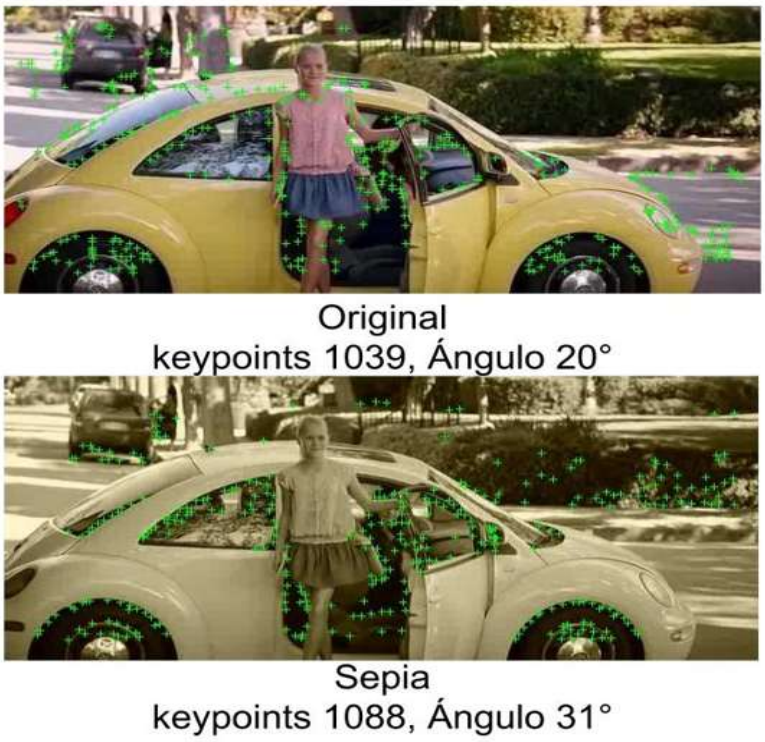

Figura 4.3: Se muestran dos imágenes una con el filtro sepia a las cuales se les calcularon sus puntos de interés y su ángulo.

y con ellos utilizar algún otro tipo de algoritmo para identificar el vídeo al que pertenece.

Capítulo 5

Conclusiones

Los objetivos que se plantearon en la tesis fueron la obtención de un descriptor robusto que caracterizará un vídeo y la realización de un buscador de vídeos, para la realización de estos objetivos se implementó un algoritmo, al cual se le pasa un vídeo y este realiza un procesamiento de él para extraerle sus cuadros representativos *keyframes*, los cuales se obtienen al haber un cambio de escena en el vídeo, con estos *keyframes* se obtiene una huella digital del vídeo. El siguiente proceso es calcular los puntos de interés llamados *keypoints* de una imagen, con estos *keypoints* formar un polígono irregular al cual se le calcula su centroide, con el centroide y el origen se construye una línea recta, a la cual se le obtendrá su ángulo de inclinación, con este proceso aplicado a cada *keyframe* perteneciente a la huella digital de un vídeo se construye un patrón de búsqueda que represente al vídeo, ya con el patrón se realiza una búsqueda contra patrones de vídeos preprocesados, para así identificar a cuál de estos pertenece el vídeo.

El mayor problema que se presentó durante la realización de la tesis fue la obtención de un descriptor para una imagen que fuera lo suficientemente robustó, para cuando se le aplicara un filtro a la imagen, el descriptor no sufriera una gran modificación.

Los vídeos que se utilizaron para el trabajo se les aplicó transforma-

ciones de distintos tipos, unas que son casi imperceptibles al ojo humano así como unas que distorsionan el vídeo de forma considerada, para la parte de la búsqueda se implementó el algoritmo DP para poder tener un mejor apareamiento de entre la máscara y los vídeos preprocesados.

La búsqueda con vídeos obtenidos directamente del vídeo original y sin realizarles ninguna modificación, los resultados fue ron que siempre se puede saber a cual vídeo es al que pertenece ya que lo que se realiza es una búsqueda exacta entre palabras.

Con videos a los cuales se les aplicó una transformación ligera el algoritmo propuesto tiene un buen funcionamiento, con las pruebas realizadas se encuentra con seguridad a qué película pertenece el vídeo a buscar, ya que aunque existen casos en el que nos da algunos falsos positivos como el algoritmo de búsqueda que se implemento permite una cantidad de errores para mejorar la eficacia de este, se considera la respuesta que tenga menor error arrojado por el algoritmo y en caso de un empate se muestran las soluciones.

El caso de mayor interés o que tiene como una aplicación práctica sería que el vídeo de búsqueda fuera obtenido por medio de la grabación por un celular, este tipo de vídeos entra en el conjunto de vídeos con modificaciones fuertes, ya que en él existen diferentes modificaciones y realizan un cambio considerable tanto en la obtención de los keyframes como de los keypoints. Con este tipo de vídeos se obtuvieron resultados similares a los obtenidos con vídeos con modificaciones ligeras, salvo que cuando la cantidad modificación considerable el algoritmo manda múltiples apareamientos con distintas películas el cual no reduce el problema de la búsqueda.

Los objetivos planteados en esta tesis de encontrar un descriptor característico del vídeo, así como crear un algoritmo de búsqueda para detectar vídeos, se alcanzaron parcialmente, ya que para los vídeos sin modificación

o con alguna muy pequeña, se obtienen buenos resultados en la búsqueda del vídeo, pero cuando el vídeo a procesar tiene modificaciones fuertes nos arroja en algunos casos falsos positivos o ninguna ocurrencia con los vídeos preprocesados.

Para la parte del desarrollo de un algotimo de búsqueda se optó por utilizar un algoritmo implementado en el procesamiento de cadenas, dado que se ajusta muy bien a las necesidades del trabajo, ya que permite incluir los dos tipos de errores con los cuales se permite trabajar para encontrar un mejor apareamiento entre las palabras de búsqueda y los textos ya procesados.

Como futuros trabajos se pueden pensar en encontrar una caracterización los vídeos con alguna propiedad que sea más robusta a las modificaciones fuertes para que se tengan mejores resultados y probar otros algoritmos de apareamiento de cadenas con error que sean más rápidos.

Bibliografía

[1] Wael Abd-Almageed. Online, simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing. pages 3200–3203, 2008.

[2] Jurandy Almeida, Neucimar J Leite, and Ricardo da S Torres. Comparison of video sequences with histograms of motion patterns. pages 3673–3676, 2011.

[3] Ricardo Baeza-Yates and Gonzalo Navarro. Multiple approximate string matching. In *Algorithms and Data Structures*, pages 174–184. Springer, 1997.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[5] William I Chang and Jordan Lampe. Theoretical and empirical comparisons of approximate string matching algorithms. In *Combinatorial Pattern Matching*, pages 175–184. Springer, 1992.

[6] Piotr Indyk, Giridharan Iyengar, and Narayanan Shivakumar. Finding pirated video sequences on the internet. 1999.

[7] Job Oostveen, Ton Kalker, and Jaap Haitsma. Feature extraction and a database strategy for video fingerprinting. pages 117–128, 2002.

[8] VideoLan project. Vlc media player. 2001.