UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Efectos de vídeo para sistemas del Museo de la Telecomunicación Vicente Miralles de la ETSIT de Valencia

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Campillo Rodríguez, Ignacio

Tutor/a: Mossi García, José Manuel

CURSO ACADÉMICO: 2022/2023

Resumen

El propósito de este trabajo fin de grado es la implementación en Matlab de efectos sobre vídeo, centrándonos, específicamente en efectos Super 8, VHS, efectos de TV antiguos y efectos de vídeo con la finalidad de incorporarlos a la página web del Museo de Telecomunicaciones.

Para lograr este objetivo, utilizamos varias técnicas de programación y ajustes de color para imitar el aspecto de películas y vídeos antiguos. El efecto Super 8 se logró usando filtros y ajustes de color, y el efecto VHS se logró usando distorsión de imagen y ajustes de color. El efecto de TV antiguo se logró usando filtros de ruido y ajustes de color, y el efecto de vídeo antiguo inestable se logró agregando un efecto de fluctuación a la imagen.

Este trabajo demostró la viabilidad de las técnicas de programación para crear efectos de vídeo con el propósito de implementarlo en la página web del museo de la universidad.

Resum

El present treball de fi de grau té com a objectiu analitzar els efectes de vídeo realitzats amb Matlab, específicament l'efecte *Super 8, l'efecte *VHS, l'efecte de televisió antiga, i l'efecte de vídeo antic amb tremolor en blanc i negre i en color sépia.

Per a aconseguir aquest objectiu, es van utilitzar diferents tècniques de programació i ajustos de color per a imitar l'aparença de pel·lícules i vídeos antics. L'efecte *Super 8 es va aconseguir mitjançant l'ús de filtres i ajustos de color, mentre que l'efecte *VHS es va aconseguir mitjançant la distorsió de la imatge i l'ús d'ajustos de color. L'efecte de televisió antiga es va aconseguir mitjançant l'ús de filtres de soroll i ajustos de color, i l'efecte de vídeo antic amb tremolor es va aconseguir mitjançant la introducció d'un efecte de tremolor en la imatge.

En general, els efectes de vídeo creats amb Matlab poden millorar significativament l'aparença d'un vídeo i brindar una experiència visual més atractiva. Aquest treball va demostrar la utilitat d'aquestes tècniques de programació per a crear efectes de vídeo amb el propòsit d'implementar-lo en la pàgina web del museu de la universitat.

Abstract

The objective of this final degree work is to analyze the video effects made with Matlab, specifically the Super 8 effect, the VHS effect, the old television effect, and the old video effect with black and white and sepia color trembling.

To achieve this goal, different programming techniques and color adjustments were used to mimic the look of old movies and videos. The Super 8 effect was achieved through the use of filters and color adjustments, while the VHS effect was achieved through image distortion and the use of color adjustments. The old TV effect was achieved by using noise filters and color adjustments, and the old video effect with shaking was achieved by introducing a shaking effect into the image.

Overall, video effects created with Matlab can significantly enhance the appearance of a video and provide a more engaging visual experience. This work demonstrated the usefulness of these programming techniques for creating video effects for the purpose of implementation on the university's museum website.

Índice

Capítulo 1. Introducción

1.1 Introducción al trabajo desarrollado.

Este prototipo surge de la idea de implementar en la página web del museo de la universidad politécnica de Valencia una herramienta capaz de aplicar un efecto de vídeo antiguo a un vídeo cualquiera que sea cargado por un usuario.

Se ha escogido la herramienta de Matlab para el procesado del vídeo ya que es realmente útil e interesante para este tipo de trabajo por varias razones:

1. Tiene una gran biblioteca de procesamiento de vídeo integrada que incluye bastantes herramientas que nos van a resultar útiles a lo largo de este prototipo.
2. Las funciones de procesamiento de imágenes están realmente avanzadas, nos pueden ser útiles a la hora de procesar y manipular cada frame, ya que podemos realizar varias operaciones como pueden ser segmentar o eliminar/añadir ruido en la imagen.
3. Análisis de datos, puede ser útil para el usuario poder analizar datos de sensores de vídeo, medir la velocidad y la dirección del movimiento, y realizar análisis de series temporales en el vídeo.
4. Herramientas de visualización: Matlab también cuenta con herramientas de visualización avanzadas que permiten a los usuarios visualizar los resultados del procesamiento de vídeo de manera clara y efectiva. Esto es especialmente útil para la presentación de datos a otros usuarios o para el análisis visual de los resultados.

Una vez el vídeo ha sido cargado procedemos a su descomposición y al procesado de cada frame para luego volver a juntarlo en un único vídeo de salida configurando su como resultado final.

Capítulo 2. Museo

2.1 Web del Museo

La idea del proyecto es replicar como serían diferentes vídeos grabados con diferentes instrumentos que están presentes en la web del Museo.
En la web podemos encontrar un ejemplo de procesado de audio, en el cual la aplicación te permite seleccionar entre los diferentes equipos de audio y a partir de un fichero de entrada que puede subir el usuario simular como se escucharía el sonido que reproduciría un teléfono de baquelita, un fonógrafo, un gramófono, un magnetófono de hilo, un manipulador morse o un equipo de radiocomunicaciones antiguo.

Figura 1. Captura de la web del Museo apartado audio.

2.2 Diferentes equipos de vídeo

En la página web del Museo podemos observar los diferentes equipos de vídeo. Incluyendo descripciones [2], así como un breve resumen de sus características principales.

En un futuro, la idea es ir actualizando la página web del museo de la universidad incorporando más equipos de vídeo.
Clicando en cada equipo de vídeo podemos acceder a una página donde encontraremos información más en profundidad acerca de este, como, por ejemplo, quién fue su creador, en qué época de la historia y como ha sido su evolución.

Figura 2. Descripción de la página web del museo.

Podemos destacar los siguientes equipos de vídeo, Société Pathé Frères el cual es un proyector de películas de 9,5 mm, un aparato de manejo sencillo y de uso doméstico que se popularizó entre los años 20 y 30. Su funcionamiento se resume en unos simples pasos colocación de la película, encendido de la bombilla, enfoque de la proyección y movimiento de la manivela.

Figura 3. Proyector Pathe Baby.

Otro equipo de video que se puede encontrar en el museo es la televisión.
Se originó con la invención del disco Nipkow en 1884, el cual consistía en un disco metálico que incorporaba una serie de agujeros cuadrangulares con forma de espiral, cuando el disco giraba por cada agujero penetraba la luz reflejada de la escena filmada. Evolucionó con el desarrollo de sistemas electromecánicos y electrónicos. En la década de 1930 se realizaron las primeras retransmisiones y en 1936 se retransmitieron las Olimpiadas de Berlín, lo que se considera la primera retransmisión en directo.
A mediados del siglo XX, la televisión se extendió por todo el mundo. La introducción del color, el uso de satélites y la digitalización fueron hitos significativos en su desarrollo. Hoy en día, se utilizan diferentes estándares de transmisión digital terrestre en diferentes partes del mundo. La televisión se ha convertido en una parte integral de nuestras vidas que nos informa, entretiene y se comunica con nosotros.

Figura 4. Televisiones

Capítulo 3. Efectos de vídeo

3.1 Efectos de vídeo en general

Los efectos de vídeo son herramientas y técnicas utilizadas en producciones con la finalidad de enriquecer su contenido.
Estos efectos se pueden generar mediante software o hardware, los efectos generados mediante software pueden ser utilizados en aplicaciones de vídeo avanzadas, programas de composición o animación 3D, en cambio, el hardware es utilizado en la creación de vídeo, se realiza mediante el uso de grandes computadoras con gran capacidad de procesamiento (GPU) y memoria RAM suficiente para manejar los cálculos. Aunque la combinación de ambos permite a los profesionales de la industria audiovisual crear efectos de alta calidad y realistas Estos recursos brindan la flexibilidad necesaria para experimentar, ajustar y refinar los efectos hasta obtener el resultado deseado.
Existen una gran variedad de efectos de vídeo, cada uno con sus características y sus diferentes aplicaciones, podemos destacar los siguientes:

- Efectos de transición, este efecto en general se aplicar para suavizar el paso de una escena a otra, para proporcionar una conexión entre los clips, pueden ser sutiles o llamativos.
- Efecto de composición, este efecto se utiliza para combinar múltiples elementos en una misma escena, puede ser utilizados para entrelazar objetos reales con objetos ficticios, puede resultar útil a la hora de crear paisajes o escenarios surrealistas o fantásticos.
- Efecto de distorsión, este efecto consiste en deformar o alterar la apariencia de un vídeo, existen diferentes tipos de distorsiones como pueden ser efectos de ondulación, estiramiento, espejo, mosaico, entre otros, brindando un aspecto visual único y llamativo.
- Efectos de partículas, este efecto se genera con partículas virtuales, como polvo, desprendimiento, humo. Añadiendo dinamismo y realismo creando composiciones visualmente impactantes contribuyendo a la inmersión del espectador.

- Efectos de color y corrección, estos efectos son utilizados a la hora de ajustar la tonalidad, el contraste, la saturación y otros parámetros de la imagen. En este caso los hemos utilizado para simular efectos vintage, blanco y negro lo que contribuye a crear una atmosfera visual coherente.
- Efectos de movimiento, estos efectos añaden movimiento a elementos estáticos o generar grandes animaciones. Puede incluir desde panorámicas, zoom, rotaciones, desenfoques de movimiento, seguimiento de objetos hasta efectos de cámara lenta o rápida. Estos efectos aportan gran dinamismo y energía a las escenas, capturando la atención del espectador.
- Efectos 3D, estos efectos permiten crear objetos tridimensionales virtuales, que interactúan con el entorno y generan una sensación de profundidad. Se utilizan en animaciones, modelado de personajes, o escenarios virtuales

3.2 Editores de vídeo

Los editores de vídeo son aplicaciones software diseñadas específicamente para la edición de vídeo, las cuales nos permiten importar, organizar, recortar, fusionar y ajustar clips de vídeo, así como agregar efectos visuales, transiciones, música, texto y otros elementos para crear una narrativa visual coherente.

Además, los editores de vídeo ofrecen herramientas para ajustar el color y brillo, corrección de color, estabilización de imagen, eliminación de ruido y mejora de calidad visual. Todo ello con la finalidad de refinar y mejorar la apariencia de los clips.

Los editores de vídeo también brindan opciones de edición de audio, lo que permite ajustar el volumen, agregar música de fondo, mezclar pistas de audio y realizar otras modificaciones relacionadas con el sonido.

Existen numerosos editores de vídeo, los cuales nos brindan herramientas básicas a la hora de aplicar efectos a nuestros vídeos, existen desde editores sencillos para

principiantes hasta editores de vídeo profesionales se utilizan en la industria cinematográfica, algunos de los principales editores pueden ser:

- Adobe Premiere Pro. es una herramienta de edición de vídeo profesional ampliamente utilizada en la industria del cine y la televisión. Ofrece una amplia gama de funciones y herramientas avanzadas para la edición y producción de vídeo.
- Final Cut Pro, es un editor de vídeo desarrollado por Apple para usuarios de Mac. Proporciona una interfaz intuitiva, potentes herramientas de edición y una integración fluida con otros productos de Apple
- DaVinci Resolve, es una herramienta de edición de vídeo y corrección de color utilizada por profesionales en la industria audiovisual. Ofrece funciones avanzadas de edición, corrección de color y efectos visuales en una sola aplicación.
- iMovie, es un editor de vídeo popular entre los usuarios de Mac. Es fácil de usar y brinda funciones básicas de edición, como recorte de clips, ajuste de color y añadir efectos.
- Sony Vegas Pro, es un editor de vídeo ampliamente utilizado que ofrece una amplia gama de funciones y herramientas de edición , así como opciones de efectos visuales y de audio.

3.3 Matlab como herramienta de trabajo.

Matlab es un entorno y un lenguaje de programación desarrollado por MathWorks. Especialmente utilizado en campos como pueden ser la ciencia, la ingeniera y la investigación.
Una de las principales razones para utilizar Matlab en el tratamiento de imagen es el número de funciones y herramientas que tienen desarrolladas para el procesado adecuado de imágenes. Que permiten desde el uso básico de operaciones como cargar, leer y escribir imágenes, hasta manipular pixeles y realizar operaciones aritméticas y lógicas en imágenes, estas funciones integradas nos permiten el procesado eficiente y preciso de la imagen.

Además de las funciones descritas antes también contamos con el uso de funciones más avanzadas para el procesado de imagen. Estas herramientas incluyen filtros, transformaciones geométricas, segmentación de imagen, detección de bordes, extracción de características entre otras.

Otra ventaja que ofrece Matlab es la capacidad que tiene para trabajar con vectores y matrices, lo que nos ayuda mucho a la hora de procesar imágenes, además que, Matlab ha sido diseñado para trabajar con datos multidimensionales, como son las imágenes y posee una sintaxis muy intuitiva para realizar operaciones con las matrices, esto simplifica gratamente el manejo de los datos de las imágenes y permite realizar operaciones sobre todos los pixeles.
Matlab ofrece una interfaz intuitiva y muy útil a la hora de realizar gráficas, visualizar imágenes y crear histogramas, lo que facilita la visualización y el análisis de los resultados del procesamiento de imágenes.

Por último, Matlab ofrece una fácil integración con diferentes herramientas y bibliotecas.
Se puede combinar con el uso de bibliotecas de procesado de imágenes externas como por ejemplo OpenCV, para ampliar las capacidades de procesado de imágenes. Además, Matlab se integra bien con otros lenguajes de programación.

Capítulo 4. Desarrollo de los efectos

4.1 Esquema realizado para el proceso de los efectos en Matlab.

Todos los efectos realizados en Matlab siguen el siguiente esqueleto y solo cambia el procesado realizado sobre cada frame. Cada efecto está formado por diferentes funciones, que serán explicadas más adelante.
El código esqueleto mostrado tiene como finalidad procesar y aplicar efectos visuales a cada fotograma, la explicación técnica de su funcionamiento es la siguiente:

Primero definimos un directorio de trabajo que representa el lugar en el cual se van a ir almacenando los frames procesados, con la finalidad de volver a incluirlos todos en único video de salida con el efecto realizado.

A continuación, creamos un objeto VideoReader que se utiliza para leer un archivo de video, el objeto proporciona funciones y propiedades para acceder de forma individual a cada fotograma del vídeo, por consiguiente, permite el procesamiento y análisis de cada cuadro de forma independiente.
Después, obtenemos las dimensiones del vídeo que nos van a ser útiles en el procesado más adelante.
Comenzamos con el tratamiendo de cada fotograma, se utiliza un bucle while para recorrer cada frame del vídeo donde podemos destacar que se realizan las siguientes acciones:

1. img = readFrame(VideoFrames); Esta línea lee el siguiente fotograma del vídeo y lo almacena en una variables de nombre ‘img’.
2. En el siguiente punto se puede aplicar el efecto a la variable ‘img’ ya sea con funciones creadas por ti, o con funciones propias de Matlab como pueden ser pasar a grises, ajustar brillo o contraste entre otras.
3. Una vez han sido aplicados los efectos deseados, procedemos a guardar cada fotograma procesado en un archivo de imagen utilizando la función imwrite(J,fullname); La variable J, es el nombre que tiene cada frame tras su modificación y es el que guardamos en el directorio de trabajo.

4. Para poder ajustar el nombre y el formato de los archivos de imagen guardados cambiando la línea
   filename = [sprintf('%03d',ii) '.jpg'];.
   Por ejemplo, si deseas guardar con otro formato los archivos de imagen puedes cambiar el '.jpg' por la extensión '.png' .

Después de salir del bucle while, continuamos con las siguientes acciones
imageNames = dir(fullfile(workingDir,'*.jpg')); En este caso utilizamos la función 'dir' para obtener una lista de los nombres de archivo que coinciden con el patrón '.jpg' en el directorio de trabajo esto se almacena en una nueva variable. Cada elemento de la celda representa un nombre de archivo imagen.
A continuación, creamos un objeto 'VideoWriter' :
outputVideo = VideoWriter(fullfile(workingDir,'temblor')); con el nombre 'outputVideo' que se utiliza para escribir los fotogramas procesados en un nuevo vídeo, el nombre y la ubicación se especifican en la ruta del directorio de trabajo y el nombre, por ejemplo, en este caso 'temblor'.
La siguiente línea outputVideo.FrameRate = VideoFrames.FrameRate; establece la velocidad de los fotogramas del vídeo de salida igual a la velocidad de fotogramas del vídeo original.
De esta manera el vídeo de salida tendrá la misma velocidad de reproducción el de entrada, esto se irá modificando en función del efecto deseado, no todos los efectos tendrán la misma velocidad de reproducción.
Continuando, la línea open(outputVideo) abre el objeto 'outputVideo' para comenzar a escribir los fotogramas en el vídeo de salida.
Ya para terminar, entramos en el bucle 'for' el cual recorre cada uno de los elementos 'imageNames'.
Cada elemento representa el nombre del archivo imagen que se encuentra dentro de la ruta establecida y dentro de este bucle se realiza los siguiente:

1. En la línea img = imread(fullfile(workingDir,imageNames{ii})); se utiliza la función 'imread' para leer el archivo especifico correspondiente al índice 'ii' en el arreglo de imágenes 'imagesName', se utiliza 'fullfile' para construir la ruta completa combinando el directorio de trabajo con nombre del archivo de imagen.

2. Para terminar, utilizamos la función 'writeVideo' para escribir la imagen 'img' en el objeto creado anteriormente 'outputVideo' con esto conseguimos que cada frame se agregue al vídeo salida que se está creando writeVideo(outputVideo,img).

4.1.1 Esqueleto

```
workingDir= '';
VídeoFrames = VídeoReader('PRUEBA.mp4');
rows=VídeoFrames.Height;
columns=VideoFrames.Width;
ii = 1;

while hasFrame(VideoFrames)
   img = readFrame(VideoFrames);
   filename = [sprintf('%03d',ii) '.jpg'];
   fullname = fullfile(workingDir,filename);
   imwrite(J,fullname);
   ii=ii+1;

end

imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';
outputVideo = VideoWriter(fullfile(workingDir,'temblor'));
outputVideo.FrameRate = VideoFrames.FrameRate;
open(outputVideo)


for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));
   writeVideo(outputVideo,img)

end

close(outputVideo)
```

Código 1. esquelto.m

4.2 Efecto de película antigua con manchas por desprendimiento y efecto temblor.

4.2.1 Historia y características del efecto.

El efecto de película antigua es una técnica de postproducción de vídeo que buscar recrear la apariencia y la sensación de las antiguas películas.

Durante las primeras décadas del cine las películas eran mudas, es decir, sin diálogos grabados, donde podemos destacar directores como Charlie Chaplin.
Además, las películas estaban filmadas en blanco y negro debido a las limitaciones de la época, esto conllevo a que los directores tuviesen que jugar con las sombras.

Más adelante, a mediados del siglo XX, se desarrollaron diferentes técnicas para poder grabar y proyectar películas en color esto abrió las puertas para conseguir una estética más realistas y nuevas posibilidades creativas.

Las películas se grababan sobre un material llamado celuloide el cual era muy propenso a sufrir daños y defectos, como pueden ser manchas de polvo, desprendimiento, arañazos y desgaste. Estos defectos se debían en gran parte por el resultado de estar manipulando estos rollos de películas durante su proyección en salas de cine.

En la era digital, los artistas y cineastas comenzaron a experimentar con técnicas de postproducción para lograr el efecto de película antigua con el uso de software de edición de vídeo, se pudieron agregar digitalmente manchas por desprendimiento, arañazos o desgaste en las imágenes. El uso de todos estos elementos consiguió que las películas tuviesen un toque más envejecido, recordando la nostalgia de las películas antiguas.

Además del efecto de desprendimiento, otro elemento importante de este efecto es el efecto del temblor de la cámara. En las películas antiguas algunos de los movimientos de cámara no conseguían ser estables debido a que no eran tan sofisticados como los de

ahora. Esto conllevaba a movimientos irregulares y temblor en la imagen, este efecto se puede simular mediante el uso de editores digitales.

El efecto de película antigua se ha utilizado en numerosas producciones cinematográficas, comerciales y vídeos musicales con el fin de evocar esta estética vintage. Estas películas utilizan este efecto como una herramienta narrativa para transportar al espectador a una época pasada y recrear la atmosfera de las películas clásicas.

Figura 5. Cinematógrafo.

4.2.2 Código realizado.

Para la compresión del código se ha decidido realizar un diagrama de flujo como el que se muestra en la figura 7, el vídeo de entrada es el input de la rutina converterFrames.m. Es el vídeo sobre el que queremos realizar el efecto, durante esta rutina el vídeo se va a ir descomponiendo en frames. La función crear_vinyeta.m tiene crea una viñeta la cual se aplicará a todas las imágenes.

En proceso.m se aplica una serie de modificaciones a los frames para que tenga una estética más vintage.

La función vinyeta.m hace la multiplicación de la imagen resultado de proceso.m y la viñeta creada anteriormente, para terminar, una vez se ha aplicado los efectos de color y el de viñeta en la rutina principal a la imagen resultado pasa por otra función de nombre temblor, la cual proporciona una sensación de movimiento al vídeo.

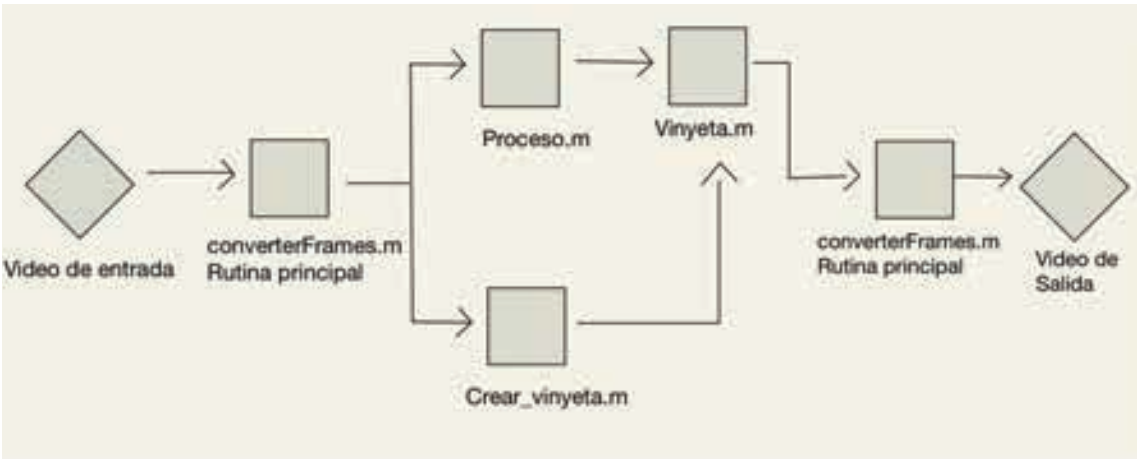

Figura 6. Diagrama de flujo del primer efecto.

Vamos a proceder con la explicación de cada función por separado y como se comunican entre ellas:

- converterFrames.m. Partiendo del esqueleto anteriormente descrito, debemos añadir una serie de líneas de código para la realización del efecto. Para simular la estética de las líneas en las pantallas y manchas de suciedad o desprendimiento debemos incorporar a nuestro directorio un vídeo en el que se aplique arañazos y manchas a un fondo negro.

Una vez tenemos cargado ese vídeo creamos un objeto videaReader para leer los fotogramas del vídeo 'defecto.mp4', además debemos obtener parámetros del vídeo como son las filas y las columnas que serán utilizados como parámetros de entrada de la función crear_vinyeta.m.

Ahora dentro del bucle 'while' procedemos a leer los frames del vídeo de entrada, además también se debe leer cada frame del vídeo defecto, y almacenamos el resultado en diferentes objetos.

Se debe ajustar el tamaño del vídeo defecto al tamaño del vídeo de entrada para que puedan combinarse utilizando la función 'imresize'.
Después de ajustar el tamaño debemos pasar defecto a blanco y negro con la función 'rgb2gray', y se crea una matriz tridimensional con el mismo tamaño que la anterior, DefectoG= rgb2gray(Defecto2);
defecto3=zeros([size(DefectoG), 3]);
A continuación, debemos asignar los valores de la imagen en escala de grises a cada canal de color RGB
defecto3(:,:,1)= DefectoG; defecto3(:,:,2)= DefectoG;
defecto3(:,:,3)= DefectoG;.

De esta manera los tres canales tendrán el mismo valor de intensidad. Ahora para terminar debemos superponer ambas imágenes
result=uint8(255 - ((255-double(im22)).*(255-double(defecto3)))/255);
Primero, se realiza una resta de las matrices 255 - double(im22) y 255 - double(defecto3);
Esto se hace para invertir los valores de intensidad de ambas imágenes.

Luego, se multiplica elemento por elemento y se divide por 255 para normalizar los valores. Finalmente, se convierte la matriz resultante a uint8.

- Crear_vinyeta.m, esta función tiene como parámetros de entrada las filas y las columnas del vídeo, se crea una máscara que tiene que ser aplicada a la imagen

con la finalidad de oscurecer gradualmente los bordes, generando que la atención del espectador sea dirigida hacia el centro de la imagen.

Básicamente el funcionamiento del código es el siguiente:

1. Calculamos el centro de la imagen a partir de los parámetros de entrada, se calcula el centro de la viñeta como la mitad de las dimensiones de los fotogramas más un desplazamiento de 0.5 debido a que Matlab comienza en 1 en lugar de 0.
2. Se inician dos bucles 'for' anidados, uno para recorrer cada fila de la imagen viñeta y otro para recorrer cada columna de la imagen.
3. En cada iteración del bucle se calcula de distancia euclidiana desde cada píxel de la imagen hasta el centro de la viñeta.
4. Debemos normalizar la matriz vignette dividiendo todos sus valores por el valor máximo de la matriz, esto nos asegura que sus valores se encuentren en el rango de 0 a 1.
5. Ahora debemos invertir la matriz vignette, esto se realiza para que los valores más cercanos al centro tengan un valor más alto que los pixeles más alejados.
6. A continuación, creamos una versión RGB de la matriz vignette aplicando tres copias de la misma, esto se hace para que tenga tres capas de color y pueda ser aplicada como máscara.

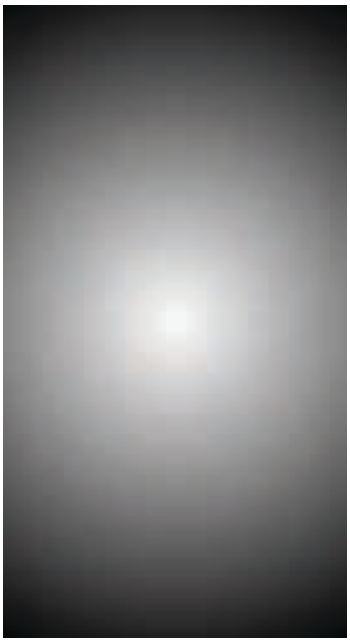

Figura 7. Máscara de viñeta.

```
function vignette= crear_vinyeta(rows, columns);
        Center = [rows/2, columns/2] +.5;
        for row=1:rows
                for col=1:columns
                vignette(row, col) = sqrt((row-Center(1))^2+(col-
                Center(2))^2);
                end
        end
        vignette = vignette / max(vignette(:));
        vignette = 1 - vignette;
        vignette = cat(3, vignette, vignette, vignette);
end
```

Código 2. Crear_vinyeta.m

- Proceso.m,, esta función realiza las siguientes transformaciones a la imagen, primero se le añade ruido de sal y pimienta el cual consiste en la adición de puntos blancos y negros en la imagen con una densidad del 2%.
  `im1= imnoise(img,'salt & pepper',0.02);.`
  Después, dependiendo del efecto que quieras realizar, porque debes seleccionar si quieres que el efecto se visualice en blanco y negro o en color sepia, si el efecto se tiene que realizar en escala de grises existe una función en Matlab 'rgb2gray' con la cual obtenemos el resultado deseado, en contra, si queremos aplicar un efecto sepia Matlab no incorpora ninguna función, pero se ha creado una que consiste en definir una matriz que contenga los factores de transformación para cada canal de color RGB, estos factores determinan cómo se combinan los valores de los canales rojo, verde y azul para obtener los nuevos valores de los canales rojo, verde y azul de la imagen con efecto sepia. Para concluir se combinan los tres canales RGB en una nueva imagen utilizando la función 'cat'.

- Vinyeta.m, en esta función simplemente se combinan la imagen resultante de la función 'proceso.m' con la viñeta creada el comienzo de la rutina principal.
  VinColor = uint8(double(img) .* vignette);

- Temblor.m, el efecto temblor se realiza recortando y redimensionando una imagen, para ello se genera un vector de nombre 'targetSize' el cual está compuesto por dos filas que representan secuencias de recorte parametrizadas en función del tamaño de la imagen de entrada, es decir, será una matriz de dos 2 filas donde cada columna representa un par de coordenadas ancho y alto de recorte. Después, debemos utilizar la función 'imcrop' para recortar la imagen, el corte se realiza utilizando las dimensiones del 'targetSize', además de añadirle un desplazamiento aleatorio entre 0 y 8.
  J = imcrop(result,[round(rand(1)*8) round(rand(1)*8) targetSize(:,ii)']),
  a continuación, debemos utilizar la funcion 'imresize' de la imagen para ajustar el tamaño del recorte al tamaño original de la imagen.

4.3 Efecto VHS

4.3.1 Historia y características del efecto

El VHS es un formato de cinta magnética utilizada para la grabación y reproducción de vídeo entre las décadas 1980 y 1990, se convirtió rápidamente en el estándar dominante para el entretenimiento en el hogar, el VHS tenía como principal objetivo competir contra el formato desarrollado por la marca Sony de nombre Betamax.

VHS ofrecía una forma accesible de grabar y reproducir programas de televisión, películas y todo tipo de contenido en cintas magnéticas, esto conllevo que las personas podían disfrutar del contenido audiovisual desde la comodidad de sus hogares, en contra posición de tener que asistir a las salas de cine o de depender de lo que programado en las emisoras.

Sin embargo, con el paso del tiempo la tecnología fue avanzando, y el VHS comenzó a mostrar sus limitaciones en términos de calidad de imagen en comparación con los formatos más modernos como en ese tiempo podía ser el DVD.

El VHS presentaba una resolución de imagen baja y una reproducción del color menos precisa, además este tipo de cintas eran propensas a deformaciones, desgate y degradación con el tiempo lo que podría comprometer la calidad del contenido grabado.

A pesar de sus limitaciones, el VHS se convirtió en un formato icónico, la estética visual se convirtió en un elemento distintivo de la época y se ha mantenido como un símbolo nostálgico de la época.

El efecto VHS se caracteriza por varios elementos visuales como pueden ser los siguientes:

- Ruido, el ruido es un símbolo de las cintas de VHS y se debe a la interferencia electromagnética. Estos elementos visuales se manifestaban como pixeles distorsionados, puntos brillantes o lineales horizontales.
- Desvanecimiento de color, con el paso del tiempo, es frecuente que las cintas VHS comiencen a perder calidad y precisión al momento de la reproducción del color, esto se puede interpretar como una apariencia descolorida, tonos más apagados y una tendencia hacia tonos más cálidos, como lo pueden ser el amarillo o el naranja.
- Defectos de seguimiento, era muy frecuente que los reproductores VHS tuviesen dificultades para mantener una reproducción estable, esto provocaba saltos y movimientos bruscos, lo que se puede traducir como, una apariencia inestable y sacudida de la imagen.
- Líneas de cinta y distorsiones en la imagen, las cintas VHS sufrían un gran desgaste con el paso del tiempo, lo que resultaba en líneas verticales u horizontales en la imagen, así como distorsiones de forma de ondas.

Figura 8. Cinta VHS.

4.3.2 Código realizado

En este caso realizamos un procesado de vídeo y aplicamos un efecto VHS a cada frame, para ello partiendo del esqueleto explicado anteriormente, debemos implementar una función para procesar cada imagen. La función tiene el nombre ‘effect_VHS’ y recibe como parámetro de entrada cada ‘frame’, y aplica el efecto de la siguiente manera:

1. Primero debemos extraer las componentes roja, verde y azul de la imagen, de la siguiente manera

   roja = img(:,:,1);

   verde = img(:,:,2);

   azul = img(:,:,3);

2. Aplicamos una operación de translación para cada componente la roja, la verde y la azul. Para conseguir este efecto debemos usar la función 'imtranslate' que se utiliza para desplazar los pixeles de la imagen según un vector de desplazamiento especificado. Para este caso se han desplazado la componente roja en 8 pixeles hacia la derecha y 8 hacia abajo, la componente verde 5 pixeles hacia la derecha y 10 hacia abajo, y por último la azul 7 pixeles a la derecha y 15 hacia abajo.

   nueva_imagen_rojo = imtranslate(roja, [8, 8]);

   nueva_imagen_verde = imtranslate(verde, [5, 10]);

   nueva_imagen_azul = imtranslate(azul, [7, 15]);

3. A continuación, creamos una matriz que va a ser la encargada de almacenar las componentes roja, verde y azul desplazadas de la siguiente manera:

   B(:,:,1) = nueva_imagen_rojo;

   B(:,:,2) = nueva_imagen_verde;

   B(:,:,3) = nueva_imagen_azul;

4. Por último, creamos un filtro de promediado de la imagen y se lo aplicamos a los canales de color de la imagen, además de añadir ruido a la imagen.

   filtro_promediado = fspecial('disk', 3);
   B = imfilter(B, factor_desvanecimiento * filtro_ promediado);
   B= imnoise(B,'salt & pepper',0.01);

La explicación anterior hace referencia a la función 'effect_VHS' pero una vez la imagen ha sido procesada, en la rutina principal debemos realizar un recorte, debido a que, al desplazar las componentes de color en las esquinas se pueden apreciar el color de cada componente, esto afecta a la hora de visualizar el vídeo resultado ya que puede resultar incluso un poco confuso.

Para solucionar el nuevo problema debemos utilizar la función ‘imcrop’ disponible en Matlab, que lo que haces es recortar una región de la imagen la cual nos resulte de interés en este caso como queremos hacer un recorte de la imagen eliminando donde las componentes están solas, no donde están solapadas debemos emplear el siguiente código:

```
im1=imcrop(im1,[20 20 columns rows ]);
```

Las dos primeras coordenadas corresponden al punto superior izquierdo y las dos últimas a la altura y anchura restando 20 pixeles para obtener el recorte.
Y por último nos quedaría utilizar la función ‘imresize’ para volver al tamaño original del vídeo.

```
im1=imresize(im1,[rows columns]);
```

Como se puede apreciar en las imágenes al no realizar el recorte parece un poco raro, hay que añadir que las imágenes son las mismas, es decir, es el mismo frame, el numero 7 concretamente, se puede apreciar en la segunda imagen como hay alguna componente que sobre sale por arriba de la imagen.

Se puede apreciar que es la imagen de la derecha la misma ya que los destellos de luz están las mismas posiciones en las imágenes.

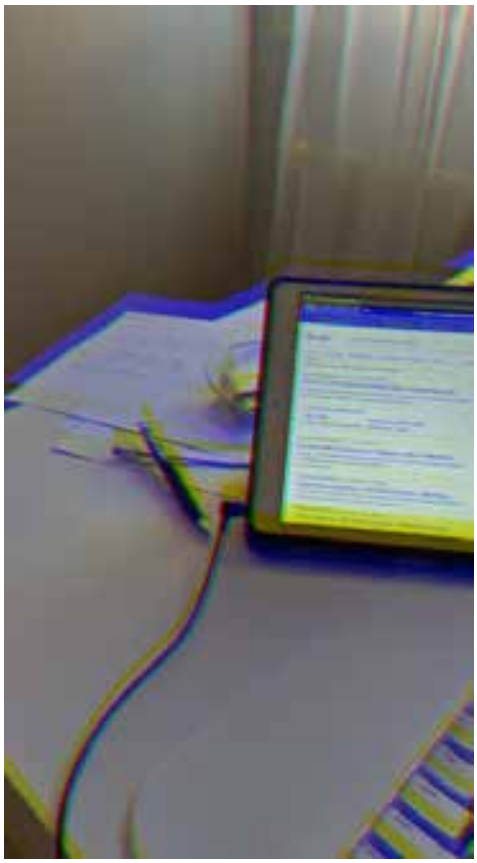

Figura 9.a Comparación frame 7.

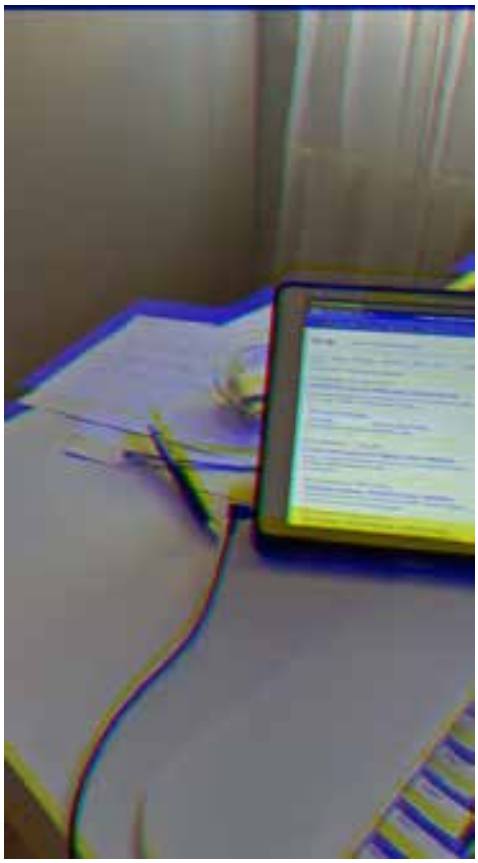

Figura 9.b Comparación frame 7.

4.4 Efecto de televisión antigua.

4.4.1 Historia y características del efecto.

El efecto de televisión antigua con líneas en la pantalla es un fenómeno icónico y característico de los televisores de rayos catódicos que se utilizaron en el pasado, este efecto se originaba debido a las limitaciones que existían.

La imagen era generada mediante un haz de electrones que golpeaba la cubierta de la pantalla recubierta con fósforo. Este haz de electrones recorría la pantalla en una seria de líneas horizontales, desde arriba hacia abajo, y de izquierda a derecha iluminando los pixeles de la imagen. Pero existía un problema debido a la velocidad finita del haz de electrones, que provocaba un retraso, esto producía la aparición de las líneas horizontales en la pantalla.

Otras variables también podían influir en la presencia de las líneas en la pantalla. Interferencias electromagnéticas, mala calidad de la señal de transmisión, ajustes incorrectos de la imagen y problemas con los componentes electrónicos podían agravar el efecto.

A pesar de ser considerado un "defecto" en aquel entonces, con el paso del tiempo, el efecto de las líneas en la pantalla se convirtió en un símbolo nostálgico de la era de la televisión analógica. Incluso en la actualidad, el efecto se recrea intencionalmente para evocar una estética retro en el cine, la fotografía y los medios digitales.

Figura 10. TV con líneas horizontales.

4.4.2 Código realizado.

Este efecto en concreto se divide en tres partes, la primera parte consiste en recrear el encendido de una televisión antigua.
Después, el correspondiente procesado de los frames para la realización del efecto en cuestión y por último el apagado de la televisión.

Comenzando por el encendido de la televisión, esta función que recibe el nombre de cabecera tiene como parámetros de entrada las filas y las columnas del vídeo, el efecto que deseamos conseguir es que recrear una secuencia de imágenes con patrones de ruido que aumenten gradualmente en tamaño hasta que coincida con el de la matriz de fondo. Para ello debemos crear una matriz de un determinado tamaño y otra con un tamaño mucho más pequeño en el centro de esta.

Los pasos que se han seguido han sido los siguientes:

1. Se ha definido un directorio de trabajo en el que se van a almacenar el conjunto de imágenes que forman la cabecera.
2. Se crea una matriz de zeros con el tamaño de las filas y las columnas, matriz que se utilizará de fondo.
   ```
   A = zeros(n, l);
   ```

3. Iniciamos la matriz de ruido con tamaño de 1, establecemos el número de pasos en el que queremos que la matriz de ruido crezca y el factor de crecimiento con el que queremos que crezca.
4. Creamos un bucle while con la siguiente condición, mientras que el tamaño de la matriz de ruido sea menor que el número de filas de la matriz de fondo y el contador que hemos definido sea mayor o igual al número de pasos que hemos establecido antes. Donde calculamos los rangos entre los que va a ir creciendo la matriz de ruido. Estos rangos reciben el nombre de rango_inicio y rango_fin.
   rango_inicio = floor((n - m) / 2) + 1;
   rango_fin = rango_inicio + m - 1;
   Con estas líneas de código calculamos el índice de inicio y de fin el cual a continuación se asignará la matriz de ruido y a la matriz inicial.

5. Creación de la matriz de ruido,
   B = randn(m, l); de esta forma generamos la matriz de ruido del tamaño establecido anteriormente y con valores aleatorios, por eso empleamos la función 'randn'.

A continuación, asignamos la matriz de ruido en la región correspondiente de la matriz inicial, es decir, se asignan los valores de 'B' en las filas desde rango_inicio hasta rango_fin y en todas las columnas.
A(rango_inicio:rango_fin, :) = B;

6. Ahora se debe indicar el directorio en el cual queremos guardar esta secuencia de imágenes que más adelante utilizaremos como cabecera de nuestro efecto.
7. Además, en el mismo bucle while la matriz de ruido va a ir aumentando por un factor que recibe el nombre de 'factorCrecimiento' con esto conseguimos que la matriz de ruido vaya creciendo en cada iteración del bucle.
8. Una vez se ha finalizado el bucle while, se encuentra la siguiente condición
   if m > n
   el tamaño m que es el tamaño inicial de la matriz de ruido, cuando sea mayor a n que es el número de filas de la matriz inicial.
   A=B;
   Asignamos B a la matriz inicial(A) para que este ocupe la pantalla completa.

9. Ahora vamos a ver la secuencia de crecimiento de las matrices como hemos establecido un numero de pasos de 15, vamos a tener 15 frames de crecimiento de la matriz de ruido, pero para que no sea redundante se van a poner menos imágenes.

Figura 11. Crecimiento matriz de ruido paso 1.

Figura 12. Crecimiento matriz de ruido paso 6.

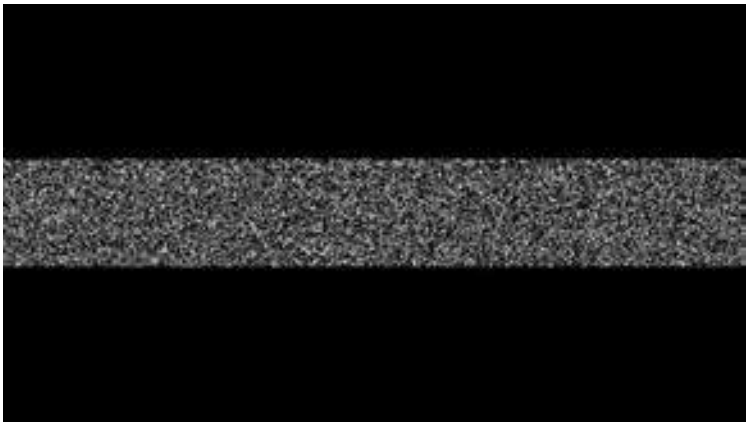

Figura 13 .Crecimiento matriz de ruido paso 11.

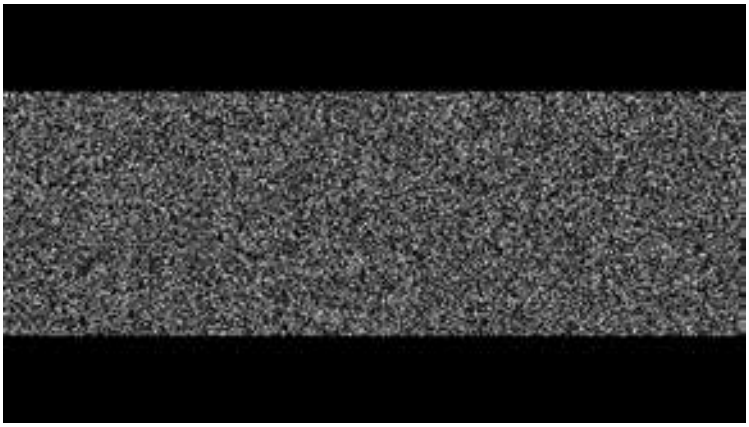

Figura 14. Crecimiento matriz de ruido paso 13.

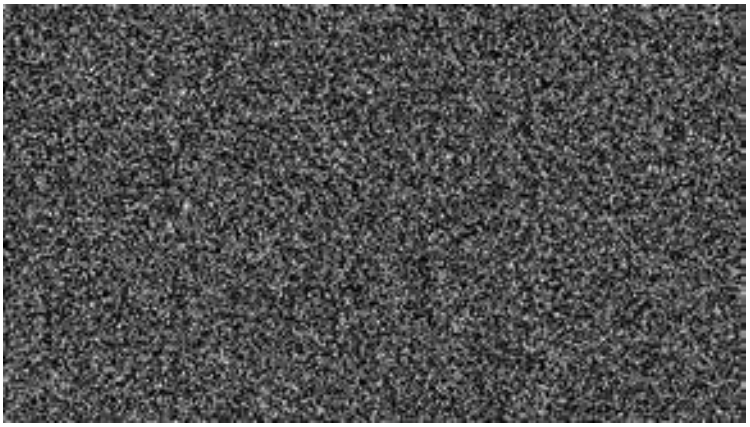

Figura 15. Crecimiento matriz de ruido paso final.

Como se ha mencionado anteriormente, ahora continuamos con la segunda parte de este efecto, el procesado de los frames del vídeo, para ello, se ha implementado una función 'tvANT.m' la cual es invocada en el esqueleto al comienzo del bucle while.

Como se ha explicado anteriormente, una de las principales características de las televisiones antiguas consistía en la aparición de líneas horizontales de diferentes intensidades, además de la visualización en blanco y negro de las imágenes.

Esta función se ha implementado de la siguiente manera:

1. Comenzamos convirtiendo la imagen de entrada a escala de grises con la función 'rgb2gray'.
2. Se crea un vector 'y' que contiene las posiciones verticales de cada línea y además establece cada cuanto pixeles se debe colocar cada línea.
   y = 1:8:size(imagen, 1);
   En este caso hemos dejado de espacio entre líneas 8 pixeles.
3. Además, se debe crear un vector de intensidades con el mismo tamaño que 'y' para aplicar líneas de mayor o menor intensidad. Incluso es necesario crear otro vector de nombre grosor el cual va a genera valores aleatorios entre 1 y 2 y del mismo tamaño que 'y' este vector servirá para determinar el grosor de cada línea.
   intensidad = rand(size(y));
   grosor = randi([1, 2], size(y));

4. A continuación, se inicia un bucle 'for' para recorrer cada posición 'y' de la imagen, en el for se va a obtener los valores de posición inicial y posición final de la línea. Para ello obtenemos las siguientes expresiones:

   start_row = max(1, y(i)-floor(grosor(i)/2));
   end_row = min(size(imagen, 1), y(i)+ceil(grosor(i)/2));

   'Start_row' se calcula como el máximo entre 1 y la posición y(i) (posición vertical de la línea) menos la mitad del grosor de la línea, redondeado hacia abajo. Esto asegura que 'start_row' no sea menor que 1 y evita que la línea se extienda fuera del límite inferior de la imagen.

   'End_row' se calcula como el mínimo entre el tamaño de la imagen en el eje vertical (size(imagen, 1)) y la posición y(i) más la mitad del grosor de la línea, redondeado hacia arriba. Esto garantiza que 'end_row' no exceda el tamaño de la imagen y evita que la línea se extienda más allá del límite superior de la imagen.

5. Por último, debemos aplicar el vector de intensidad creado anteriormente para cada línea, básicamente ajusta la intensidad de los pixeles dentro de la región de la línea multiplicándolos por la intensidad.
   imagen(start_row:end_row, :) = imagen(start_row:end_row, :) * intensidad(i);

Una vez hemos conseguido añadir las líneas horizontales a la pantalla, debemos añadir dos bordes a izquierda y derecha de la imagen, para ello se han definido las siguientes variables:
tamanio_bordes = round(porcentaje_bordes * tamano_original);

imagen_con_bordes = [zeros(rows, tamanio_bordes, size(im1, 3)), im1, zeros(rows, tamanio_bordes, size(im1, 3))];

La nueva imagen se construye concatenando tres partes: los bordes negros a la izquierda, la imagen original y los bordes negros a la derecha.
Los tamaños de los bordes negros se determinan por el número de filas de la imagen original (rows), el tamaño de los bordes (tamanio_bordes) y el número de canales de color de la imagen original (size(im1, 3)).

Ahora debemos continuar con la última parte del efecto, esta última parte del efecto es muy similar a la primera, pero debe hacerse a la inversa, es decir, en el primer caso queríamos replicar el encendido de una televisión antigua, ahora queremos replicar el apagado, antes al comienzo teníamos una matriz de zeros('negra') y otra matriz de ruido que iba creciendo, ahora debe ser al contario debemos tener una matriz de ruido aleatorio que ocupe toda la pantalla y vaya disminuyendo de la misma manera que antes iba a aumentando hasta acabar totalmente en negro.
La función recibe el nombre de cabecerarev.m y los pasos que se han seguido para la creación de esta son los siguientes:

1. Declaramos el directorio donde queremos que se guarden las imágenes de salida, añadir que esta función como la anterior tiene como parámetros de entrada las filas y las columnas y el número de frames que tiene el vídeo.

2. Después se debe establecer el número de pasos en el que tiene que decrecer, además del factor por el que tiene que decrecer.

3. Continuamos con la creación del bucle 'while' el cual tiene las siguientes condiciones:
   while m >= 1 && ii <= numPasos + jj
   siendo el m el tamaño de la matriz de ruido, el cual empezará siendo máxima e irá disminuyendo en función avance el bucle y la otra condición hace referencia, siendo jj el número de frames del vídeo y 'numPasos' el número de pasos en los que debe disminuir.

4. Ahora debemos determinar las coordenadas de inicio y fin para centrar la matriz de ruido en la matriz de ceros, recordad que al principio tienen el mismo tamaño,

por tanto, solo se verá una que será la de ruido. Ya que asignamos la matriz de ruido a la matriz de ceros.

```
rango_inicio = floor((n - m) / 2) + 1;
rango_fin = rango_inicio + m - 1;
A(rango_inicio:rango_fin, :) = B;
```

5. Para finalizar, el tamaño de 'm' que indica el tamaño de la matriz de ruido se va multiplicando por un factor de decrecimiento lo que conlleva que su tamaño se vaya viendo considerablemente reducido hasta llegar a cero.

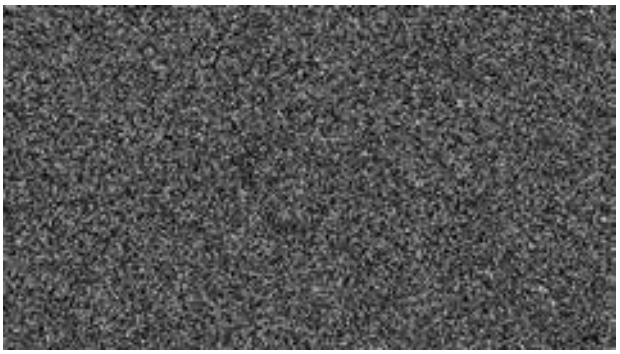

Figura 16. Decrecimiento matriz de ruido paso inicial.

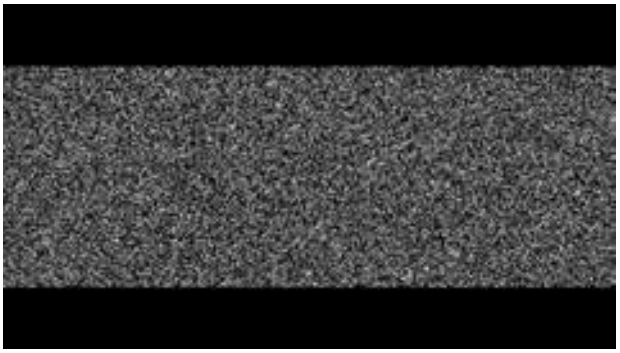

Figura 17. Decrecimiento matriz de ruido paso 4.

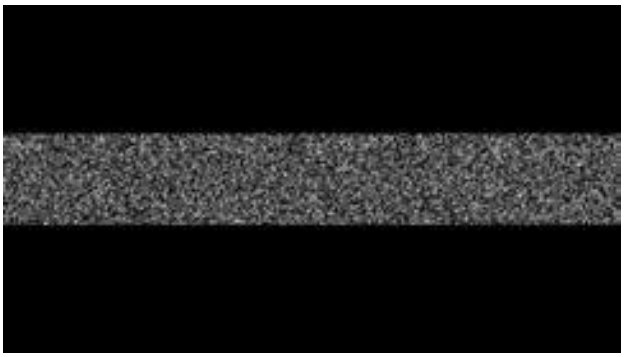

Figura 18. Decrecimiento matriz de ruido paso 8.

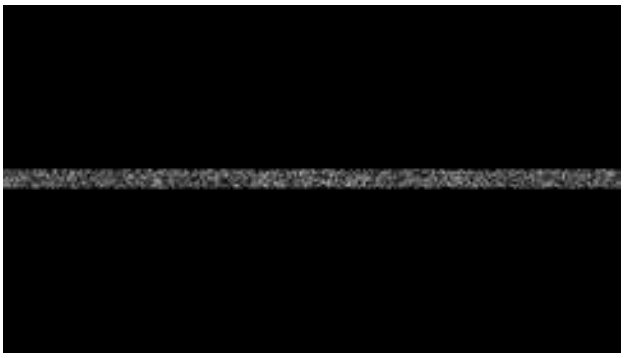

Figura 19. Decrecimiento matriz de ruido paso 13.

Figura 20. Decrecimiento matriz de ruido paso final.

4.5 Efecto de super 8.

4.5.1 Historia y características del efecto 8.

La historia de las cámaras Super 8 se remonta a la década de 1960, cuando la compañía Eastman Kodak lanzó el formato Super 8 como un desarrollo del formato de película anterior de 8 milímetros. Introducido en 1965, Super 8 se convirtió rápidamente en una opción popular entre los entusiastas de las películas caseras y los cineastas independientes debido a su facilidad de uso y precio asequible.

El formato Super 8 usaba película de 8 mm en cartuchos fáciles de cargar, lo que permitía grabar y reproducir películas sin procesamiento profesional. Esto hizo que el cine en casa fuera mucho más accesible y popular entre los fanáticos.

Las cámaras super 8 tenían unas características bastante mejores comparadas con el anterior formato de 8 milímetros. Tenían lentes intercambiables, enfoque y control de exposición, esto permitía que el usuario tuviese un mayor control sobre la calidad de imagen que capturaban.

Algunas de las principales características del efecto super 8 son las siguientes:

1. Aspecto granulado. Debido al tamaño del negativo utilizado el efecto super 8 tenía un aspecto granulado añadiendo textura y ruido a la imagen.
2. Colores desaturados, el efecto super 8 reduce la saturación de los colores y le da un tono más suave, estás cámaras tenían una reproducción de color característica con tonos desaturados y una apariencia retro.
3. Vignette, el efecto super 8 recrea este viñeteo añadiendo una graduación de brillo en los bordes de las imágenes, provocando bordes más oscuros o desvanecidos.
4. Parpadeos y destellos, las películas super 8, a veces presentaban destellos de luz y parpadeo debido a las características de la cámara y la película utilizada. También el efecto super 8 suele tener cambios de exposición simulados para imitar este aspecto.

5. Efecto de desenfoque, debido a las limitaciones técnicas de las cámaras y a las lentes utilizadas a menudo se producía un desenfoque.

4.5.2 Explicación del código realizado.

Comenzaremos procesando cada frame en una nueva función con nombre 'efectosuper8', esta función tiene como parámetro de entrada cada imagen del vídeo y como salida la imagen procesada para la realización del efecto.

Para la realización del efecto seguimos los siguientes pasos:

1. Primero se debe ajustar el contraste de la imagen, para ello se va a emplear la función 'imadjust', esta función permite ajustar el rango de intensidad de una imagen, en este caso no especificamos los límites de los valores de entrada.
   imagenImperfecta = imadjust(imagenImperfecta, [], [], 0.5);
2. Se debe crear una matriz de ceros con el mismo tamaño que la imagen de entrada, la cual representa la máscara de arañazos. Además, también se debe ajustar el número de arañazos que debe aparecer en cada frame, en este caso para este efecto, se ha determinado que debe ser un numero aleatorio de arañazos entre diez y quince.
3. A continuación, se crea un bucle 'for' desde 1 hasta el número de arañazos, en el cual se calcula la longitud del arañazo, que es otro valor aleatorio entre veinte y sesenta.

   Ahora debemos saber la dirección de este arañazo para ello creamos una variable aleatoria entre 0 y 1, tomando 0 para la dirección horizontal y 1 para la dirección vertical. Dependiendo de la dirección del arañazo se ajustan las coordenadas 'x' e 'y' y se aplica la máscara la imagen correspondiente.
4. Después de crear la máscara de arañazos debemos crear la máscara de manchas, en la cual se establece los pixeles que van a ser establecidos como manchas
5. El siguientes paso es suavizar esas manchas para que no tengan un aspecto 'poco realista' para ello se aplica un filtro de convolución a la máscara para suavizar estas manchas, al aplicar el filtro de convolución se realizada un promedio

ponderado de los pixeles vecinos para cada pixel en la máscara de manchas, se ha optado por un filtro 7x7, ya que tiene mayor capacidad de eliminar detalles pequeños y además agilizamos los cálculos ya que al ser un filtro pequeño requiere menos operaciones, se debe dividir este filtro entre 25 ya que el tamaño del filtro al ser 7x7 contiene 49 pixeles, por tanto dividir por 25 normaliza el filtro para la suma de todos sus valores sea 1. Además, ajustamos el filtro con la opción 'same' para asegurar que la imagen resultado tenga el mismo tamaño que la original, así conservamos las dimensiones espaciales.

```
mascaraManchas = conv2(double(mascaraManchas), ones(7) / 25, 'same') > 0.25;
```

6. Ahora se debe ajustar la intensidad de los arañazos y de las manchas de la siguiente manera

```
valoresAranazos = rand(filas, columnas) * 0.4;
valoresManchas = rand(filas, columnas) * 0.2;
```

7. Una vez tenemos todo lo necesario acerca de los arañazos y las manchas debemos aplicárselo a nuestra imagen de la siguiente manera:

```
imagenImperfecta = imagenImperfecta + repmat(mascaraAranazos, [1, 1, size(imagenImperfecta, 3)]) .* repmat(valoresAranazos, [1, 1, size(imagenImperfecta, 3)]);
```

   La máscara arañazos es una matriz binaria donde los pixeles con valor 1, representan la ubicación de los arañazos, mientras que los 0 indican áreas sin arañazos. A continuación debemos aplicar en las posiciones donde hay arañazos la intensidad que hemos determinado antes, además debemos tener en cuenta que se apliquen en todas las componentes de la imagen.

8. Después ajustamos la imagen para que se encuentre en el rango [0 1] para ajustar valores por encima y por debajo. Y aplicamos un desplazamiento de las componentes de colores, generamos tres números aleatorios entre [-5,5] estos números representan los desplazamientos que se harán.

```
imagenImperfecta = max(0, imagenImperfecta);
imagenImperfecta = min(1, imagenImperfecta);
desplazamientoColor = randi([-5, 5], 1, 3);
```

9. Ahora aplicamos el desplazamiento de color para cada canal, para ello utilizamos la función ‘circshift’, la cual se encarga de desplazar los elementos de la matriz en la dirección especificada en desplazamientoColor, dependiendo de cada componente.
10. Para finalizar con la función, declaramos una variable aleatoria entre [1,2], este parámetro va a determinar la cantidad de desenfoque que se va a aplicar en la imagen, una vez se ha generado ese valor, se utiliza la función ‘imgaussfilt’ que aplica un filtro de desenfoque gaussiano a la imagen, donde el valor aleatorio anterior especifica la desviación estándar del filtro. El resultado es una imagen con un efecto de desenfoque suavizado.

Figura 21. Máscaras de arañazos y manchas suavizadas.

Una vez la imagen ha pasado por la función ‘efectosuper8’ vuelve la rutina principal y se le aplica una viñeta para oscurecer los bordes y esquinas de la imagen y se le añade un texto a la imagen en el cual aparece la fecha, el nombre del efecto y la palabra ‘play’.

Capítulo 5. Análisis de resultados.

5.1 Efecto película antigua.

Para ver ayudar a entender el efecto realizado vamos a hacer una comparativa entre diferentes frames, con la finalidad de tener una visión mejorada del prototipo.

El vídeo sobre el que se va a aplicar el efecto es un vídeo que se encuentra en su formato original, sin ninguna alteración visual ni cambios estéticos. El color y la nitidez del vídeo no han sido alterados previamente, tampoco se incluirán efectos de distorsión, ni ruido, ni ningún tipo de deteriorado al vídeo original.

Se van a seleccionar los mismos frames para poder apreciar las diferencias con mayor claridad.

Figura 22.a Frames 86, 87 y 88 del vídeo sin aplicar el efecto.

Figura 22.b Frames 86, 87 y 88 del vídeo con el efecto aplicado.

Como se muestra en las ilustraciones 22 a y 22 b, la secuencia de imágenes de la ilustración 22 b, sufre una seria de modificaciones relevantes, como puede ser, el efecto viñeta. Como se puede apreciar las esquinas de las imágenes, han sido oscurecidas provocando atraer la atención al centro de la imagen. También se puede apreciar claramente la aparición de manchas de polvo y algunos arañazos ayudando a reforzar esta estética vintage. Y, para terminar, podemos destacar sutilmente, en la ilustración 22b, se puede apreciar que las cabezas no se encuentran alineadas exactamente en la misma posición en las tres imágenes, esto es provocado por el efecto temblor. Aunque no se aprecie claramente, en la secuencia de imágenes existe un efecto de movimiento entre los diferentes frames provocando conseguir simular el efecto provocado por los motores de las cámaras de cine.

5.2 Efecto VHS.

Al igual que antes, el vídeo original no ha sido alterado de ninguna forma ya que, si no, no tendría sentido realizar este experimento.

Figura 23.a Frame 8, del vídeo original.

Figura 23.b Frame 8, del vídeo con el efecto.

Como se puede apreciar, en la ilustración 26a, se pueden observar los siguientes efectos a diferencia de la ilustración 26b, comenzando con el desplazamiento de las componentes de color, provocando que la imagen tenga una estética más vintage, además, se aprecia que ha habido un pequeño desvanecimiento de la intensidad de la imagen provocando que se visualice la imagen más borrosa, además de añadir un pequeño ruido a cada frame.

5.3 Efecto TV antigua.

Al igual que en el resto de los casos, sobre el vídeo al cual vamos a aplicar el efecto no ha sufrido ninguna modificación previa. Además, este efecto está compuesto de tres partes como he comentado anteriormente, una parte de cabecera que tiene como finalidad simular el encendido de una televisión, la siguiente parte se aplica al vídeo de entrada para simular el efecto de tv antigua y por último la parte de apagado de televisión.

Figura 24.a Frame 92 aplicado el efecto.

Figura 24.b Frame 92 aplicando el efecto.

Como se puede apreciar en la ilustración 24 b, se han añadido dos bordes, uno a la izquierda y otro a la derecha de la imagen sin variar el tamaño de esta, además se han aplicados efectos para simular las líneas antiguas que aparecían en la televisión y generaban distorsiones, ya que estas líneas tienen un grosor y una intensidad distintos cada una. Y, para terminar, se aplica un filtro de desvanecimiento de imagen consiguiendo una estética vintage.

5.4 Efecto SUPER 8.

Como se puede apreciar las diferencias entre, la ilustración 25 a y 25 b son notables, destacando la presencia de manchas y arañazos dando una estética desgastada y con errores. También se ha aplicado una distorsión de color provocando que la imagen pierda intensidad y luminosidad, además de añadir un efecto de desenfoque en algunos de los frames. Por último, hay que indicar las esquinas de la imagen se han vuelto a oscurecer para captar la atención en el centro de esta.

Figura 25.a Imagen original frame 14.

Figura 25.b Imagen después de aplicar el efecto frame 14.

Capítulo 6. Conclusiones y líneas futuras

Hemos realizado el prototipo y hemos visto su viabilidad. Se han desarrollado en Matlab todos los efectos que nos propusimos como objetivo al principio de este trabajo de fin de grado, ahora se debería incorporar a la página web del museo de la universidad, para ello se debería estudiar que lenguaje de programación es el más apropiado.

Una vez el script se puede ejecutar en el servidor de la universidad, se debería desarrollar en wordpress o en html una pequeña página que contenga los diferentes equipos, con una breve descripción de estos, un botón para poder cargar los vídeos a los cuales le vamos a aplicar el efecto y el visualizador del vídeo resultado.

Además, el museo de la universidad politécnica de Valencia tiene pensado incluir nuevos equipos de grabación antiguos, esto conllevaría un nuevo estudio de los efectos y características principales de estos, y más adelante la implementación en la página web.

Capítulo 7. Bibliografía

[1] Museo de la Telecomunicación Vicente Miralles Segarra. Último acceso abril (2023), https://museotelecomvlc.webs.upv.es/

[2] Old Film Effect - Adobe Premiere Pro, Shabir Khaled . Último acceso abril (2023), https://www.youtube.com/watch?v=bIR9RnI3poM

[3] Efecto VHS en Adobe Premiere Pro, Adobe Premier Pro. Último acceso abril (2023), https://www.youtube.com/watch?v=-YNjHi6fWLE

[4]Documentación tratamiento de imagen en Matlab, MathWorks. Último acceso junio (2023), https://es.mathworks.com/help/matlab/images_btfntr_-1.html

[5]Documentación tratamiento de imagen en Adobe Premier Pro, Adobe Premier Pro. Último acceso junio (2023), https://helpx.adobe.com/es/pdf/premiere_pro_reference.pdf

[6] Efecto de TV Encendiendo y apagando (Premiere Pro Tutorial), Adobe Premier Pro. Último acceso junio (2023 https://www.youtube.com/watch?v=RrzpW74kRGk

[7] Efecto Vintage SUPER 8mm Film (Tutorial Premiere Pro), Matias Toledo. Último acceso junio (2023 https://www.youtube.com/watch?v=pwHyPjW8Kck

[8] Historia de la televisión. Último acceso julio (2023) https://www.ipp.edu.pe/blog/historia-de-la-television/

[9] Historia del VHS. Último acceso julio (2023) https://vhsadvd.wordpress.com/historia-del-vhs/

[10] Historia del formato super 8. Último acceso julio (2023) https://bwfotovideo.es/historia-del-super8-y-super16-su-uso-en-el-ambito-domestico-y-profesional/

Capítulo 8. Anexo código implementado.

8.1 Efecto película antigua:

```
workingDir= '/Users /FRAMES';
VideoFrames = VideoReader('VIDEO_PELICULA_ANTI.mp4');
DefectoFrames= VideoReader('defecto.mp4');
%Creacion secuencia de imagenes
rows=VideoFrames.Height;
columns=VideoFrames.Width;
vignette=crear_vinyeta(rows, columns);
targetSizeX=[rows-18:-1:rows-50, rows-51:rows-29,rows-18:-1:rows-50, rows-51:rows-29,612:-1:590, rows-18:-1:rows-50, rows-
51:rows-29, rows-18:-1:rows-50, rows-51:rows-29, rows-18:-1:rows-50, rows-51:rows-29, rows-51:rows-29];
    targetSizeY=round(targetSizeX./2);
   targetSize=[targetSizeY;targetSizeX];
ii = 1;
jj=1;
while hasFrame(VideoFrames)
  img = readFrame(VideoFrames);
   if jj < 100
  jj=jj+1;
  Defecto= read(DefectoFrames,jj,'default');
  Defecto2=imresize(Defecto,[rows columns]);
   else
      jj=1;
      Defecto= read(DefectoFrames,jj,'default');
  Defecto2=imresize(Defecto,[rows columns]);
   end
  filename = [sprintf('%03d',ii) '.jpg'];
  fullname = fullfile(workingDir,filename);
  VectorAleatorio = round(rand(1,8)*1);
  im11= proceso(img);% pasa a sepia
  im22= vinyeta(im11,vignette);%añade viñeta
  DefectoG= rgb2gray(Defecto2);
  defecto3=zeros([size(DefectoG), 3]);
  defecto3(:,:,1)= DefectoG;
  defecto3(:,:,2)= DefectoG;
  defecto3(:,:,3)= DefectoG;
  % result = imfuse(Defecto,im22,'blend','Scaling','joint');
   defecto3=uint8(defecto3);
   result=uint8(255 - ((255-double(im22)).*(255-double(defecto3)))/255);

%EFECTO TEMBLOR
   J = imcrop(result,[round(rand(1)*8) round(rand(1)*8) targetSize(:,ii)']);
   J= imresize(J,[rows columns]);
   T=length(targetSize);
   tt=mod(ii,T)+1;
   if (tt>T)
      tt=0;
   end
    tt=tt+1;
      imwrite(J,fullname);    % (img1.jpg, img2.jpg, etc.)

  if ii > 100, break, end
  ii=ii+1
end



%Encontrar carpeta frames PARA CONVERTIR EL CONJUNTO DE IMAGENES EN ARREGLO DE
%CELDAS

imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';

%para crear nuevos videos con la secuencia de imagenes


outputVideo = VideoWriter(fullfile(workingDir,'temblor'));
outputVideo.FrameRate = 12;%cambiar el framerate del video
%outputVideo.FrameRate = VideoFrames.FrameRate;
```

```
open(outputVideo)


for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));

   writeVideo(outputVideo,img)

end

close(outputVideo)
```

Función crear viñeta:

```
function vignette= crear_vinyeta(rows, columns);
vignette = zeros(rows, columns);

Center = [rows/2, columns/2] +.5;
for row=1:rows
    for col=1:columns
        vignette(row, col) = sqrt((row-Center(1))^2+(col-Center(2))^2);
    end
end

vignette = vignette / max(vignette(:));
vignette = 1 - vignette;

vignette = cat(3, vignette, vignette, vignette);

end
```

Función sepia:

```
function imout=sepia(X)
    tRGB = [0.393  0.769  0.189;   0.349  0.686  0.168; 0.272  0.534  0.131];

    R=double(X(:,:,1));
    G=double(X(:,:,2));
    B=double(X(:,:,3));

    %para la capa R
    tr=round(R*tRGB(1,1)+G*tRGB(1,2)+B*tRGB(1,3));
    tr(tr>255)=255;

    %capa G
    tg=round(R*tRGB(2,1)+G*tRGB(2,2)+B*tRGB(2,3));
    tg(tg>255)=255;

    %capa B
    tb=round(R*tRGB(3,1)+G*tRGB(3,2)+B*tRGB(3,3));
    tg(tg>255)=255;

    result=uint8(cat(3,tr,tg,tb));
    %imshow(result);
    imout=result;

end
```

Función aplicar viñeta:

```
function imout= vinyeta(img, vignette);

VinColor = uint8(double(img) .* vignette);

imout=VinColor;

end
```

8.2 Efecto VHS

```
workingDir= '/Users/ FRAMES/frames_VHS';
VideoFrames = VideoReader('video_vhs2.mp4');
rows=VideoFrames.Height;
columns=VideoFrames.Width;

ii = 1;


while hasFrame(VideoFrames)
   img = readFrame(VideoFrames);
   filename = [sprintf('%03d',ii) '.jpg'];
   fullname = fullfile(workingDir,filename);
   im1= effect_VHS(img);% efecto VHS
   im1=imcrop(im1,[20 20 columns-20 rows-20 ]);
   im1=imresize(im1,[rows columns]);
       imwrite(im1,fullname);    % (img1.jpg, img2.jpg, etc.)


   if ii > 100, break, end
   ii=ii+1
end



imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';
outputVideo = VideoWriter(fullfile(workingDir,'VHS2'));
outputVideo.FrameRate = VideoFrames.FrameRate;
open(outputVideo)


for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));
   writeVideo(outputVideo,img)

end
close(outputVideo)
```

Función efecto VHS:

```
function imout= effect_VHS(img)


% Obtener las componentes roja, verde y azul de la imagen
roja = img(:,:,1);
verde = img(:,:,2);
azul = img(:,:,3);
```

```
nueva_imagen_rojo = imtranslate(roja, [0, 0]);
nueva_imagen_verde = imtranslate(verde, [15, 18]);
nueva_imagen_azul = imtranslate(azul, [7, 10]);

B(:,:,1) = nueva_imagen_rojo;
B(:,:,2) = nueva_imagen_verde;
B(:,:,3) = nueva_imagen_azul;
factor_desvanecimiento = 0.9;

% Define un filtro de promedio
filtro_promedio = fspecial('disk', 3);

% Aplica el filtro a los canales de color de la imagen
B = imfilter(B, factor_desvanecimiento * filtro_promedio);
B= imnoise(B,'salt & pepper',0.01);
imout=B;
end
```

8.3 Efecto TV antigua.

```
workingDir= '/Users/FRAMES/frames_TV';
VideoFrames = VideoReader('videoTV2.mp4');
rows=VideoFrames.Height;
columns=VideoFrames.Width;

ii = 20;
porcentaje_bordes = 0.2; % Ajusta el porcentaje según tus preferencias
 im1= cabecera(rows,columns,porcentaje_bordes);% efecto VHS
 tamano_original = columns;
while hasFrame(VideoFrames)
   img = readFrame(VideoFrames);

   filename = [sprintf('%03d',ii) '.jpg'];
   fullname = fullfile(workingDir,filename);

   im1= tvANT(img);% efecto TV


tamanio_bordes = round(porcentaje_bordes * tamano_original);

% Ajusta el tamaño de la imagen con los bordes negros
imagen_con_bordes = [zeros(rows, tamanio_bordes, size(im1, 3)), im1, zeros(rows, tamanio_bordes,
size(im1, 3))];



      imwrite(imagen_con_bordes,fullname);    % (img1.jpg, img2.jpg, etc.)


   if ii > 100, break, end
   ii=ii+1
end
im1= cabecerarev(rows,columns,ii,porcentaje_bordes);% efecto VHS

imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';

%para crear nuevos videos con la secuencia de imagenes


outputVideo = VideoWriter(fullfile(workingDir,'TV'));
```

```
outputVideo.FrameRate = 12;%cambiar el framerate del video
%outputVideo.FrameRate = VideoFrames.FrameRate;

open(outputVideo)


for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));

   writeVideo(outputVideo,img)

end

close(outputVideo)
```

Función cabecera:

```
function cabecera = cabecera(rows, columns,porcentaje_bordes)
   workingDir = '/Users/nachocampillorodriguez/FRAMES/frames_TV';
   n = rows;
   l = columns;

   % Crear una matriz de ceros
   A = zeros(n, l);

   % Tamaño inicial de la matriz de ruido
   m = 2;

   ii = 1;

   % Número de pasos deseados
   numPasos = 15;

   % Factor de crecimiento ajustable
   factorCrecimiento = 1.5;

   while m < n && ii <= numPasos
      rango_inicio = floor((n - m) / 2) + 1;
      rango_fin = rango_inicio + m - 1;

      % Generar una matriz de ruido del tamaño adecuado
      B = randn(m, l);

      % Asignar la matriz de ruido en la región correspondiente de A
      A(rango_inicio:rango_fin, :) = B;

      filename = [sprintf('%03d', ii) '.jpg'];
      fullname = fullfile(workingDir, filename);

      % Redimensionar A al tamaño deseado y guardar la imagen
      A_resized = imresize(A, [rows, round(columns+(2*columns*porcentaje_bordes))]);
      imwrite(A_resized, fullname);

      m = round(m * factorCrecimiento);  % Aumentar el tamaño de B con un factor de crecimiento

      ii = ii + 1;

      imshow(A_resized);
   end

   if m > n || numPasos==ii-1
      A = B;
      A_resized = imresize(A, [rows, round(columns+(2*columns*porcentaje_bordes))]);
      imwrite(A_resized, fullname);
      imshow(A_resized);
   end

   cabecera = A_resized;
end
```

Función cabecera al final:

```
function cabecerarev = cabecerarev(rows, columns,ii,porcentaje_bordes)
    workingDir = '/Users/nachocampillorodriguez/FRAMES/frames_TV';
    n = rows;
    l = columns;
    % Tamaño inicial de la matriz de ruido
    m = n;
    jj=ii;
    ii = ii+1;

    % Número de pasos deseados
    numPasos = 15;

    % Factor de decrecimiento ajustable
    factorDecrecimiento = 0.8;

    while m >= 1 && ii <= numPasos + jj
        % Crear una matriz de ruido del tamaño adecuado
        B = randn(m, l);

        % Crear una matriz de ceros del tamaño completo
        A = zeros(n, l);

        % Calcular las coordenadas de inicio y fin para centrar la matriz de ruido
        rango_inicio = floor((n - m) / 2) + 1;
        rango_fin = rango_inicio + m - 1;

        % Asignar la matriz de ruido en la región centrada correspondiente de A
        A(rango_inicio:rango_fin, :) = B;

        filename = [sprintf('%03d', ii) '.jpg'];
        fullname = fullfile(workingDir, filename);

        % Redimensionar A al tamaño deseado y guardar la imagen
        A_resized = imresize(A, [rows, round(columns+(2*columns*porcentaje_bordes))]);

        imwrite(A_resized, fullname);

        m = round(m * factorDecrecimiento);  % Disminuir el tamaño de m

        ii = ii + 1;

        imshow(A_resized);
    end

    % Crear una matriz de ceros del tamaño completo (negro total)
    A = zeros(n, l);
    A_resized = imresize(A, [rows, round(columns+(2*columns*porcentaje_bordes))]);

    imwrite(A_resized, fullname);
    imshow(A_resized);

    cabecerarev = A_resized;
end
```

Función efecto de TV:

```
function imout = tvANT(img)
    imagen = rgb2gray(img);

    y = 1:9:size(imagen, 1); % Posiciones y de las líneas
```

```
    intensidad = (rand(size(y)))*3; % Valores de intensidad aleatorios entre 0 y 1
    grosor = randi([1, 2], size(y)); % Valores de grosor aleatorios entre 1 y 2

    for i = 1:length(y)
        start_row = max(1, y(i)-floor(grosor(i)/2));
        end_row = min(size(imagen, 1), y(i)+ceil(grosor(i)/2));
        imagen(start_row:end_row, :) = imagen(start_row:end_row, :) * intensidad(i);
    end

    imout = imagen;
end
```

8.4 Efecto Super 8.

```
workingDir= '/Users/nachocampillorodriguez/FRAMES/frames_super8';
VideoFrames = VideoReader('SUPER8_4.mp4');


rows=VideoFrames.Height;
columns=VideoFrames.Width;

ii = 1;
vignette=crear_vinyeta(rows, columns);

while hasFrame(VideoFrames)
   img = readFrame(VideoFrames);
   im1= efectosuper8(img);% efecto VHS
   im1=vinyeta(im1,vignette);

   % Insertar texto en la imagen
    im1 = insertText(im1, [10 10], 'PLAY', 'FontSize', 100, 'BoxOpacity', 0, 'TextColor', 'white', 'AnchorPoint', 'LeftTop');
    im1 = insertText(im1, [10 size(im1, 1) - 80], 'Super 8', 'FontSize', 68, 'BoxOpacity', 0, 'TextColor', 'white', 'AnchorPoint',
'LeftBottom');
    im1 = insertText(im1, [10 size(im1, 1) - 10], '18/10/2000', 'FontSize', 58, 'BoxOpacity', 0, 'TextColor', 'white', 'AnchorPoint',
'LeftBottom');



   filename = [sprintf('%03d',ii) '.jpg'];
   fullname = fullfile(workingDir,filename);
       imwrite(im1,fullname);    % (img1.jpg, img2.jpg, etc.)


   if ii > 200, break, end
   ii=ii+1
end


imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';



outputVideo = VideoWriter(fullfile(workingDir,'super8effectFIN_V6'));
outputVideo.FrameRate = 18;%cambiar el framerate del video
%outputVideo.FrameRate = VideoFrames.FrameRate;

open(outputVideo)


for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));

   writeVideo(outputVideo,img)

end
```

```
close(outputVideo)
```

Efecto Super 8:

```
function imagenImperfecta = efectosuper8(imagenOriginal)
    % Convertir la imagen a tipo double
    imagenImperfecta = im2double(imagenOriginal);

    % Ajustar el contraste de la imagen
    imagenImperfecta = imadjust(imagenImperfecta, [], [], 0.5);

    % Tamaño de la imagen
    [filas, columnas, ~] = size(imagenImperfecta);

    % Generar máscara de arañazos
    mascaraAranazos = zeros(filas, columnas); % Máscara de arañazos
    numAranazos = randi([10, 15]); % Número aleatorio de arañazos
    for i = 1:numAranazos
        longitud = randi([20, 60]); % Longitud aleatoria del arañazo
        direccion = randi([0, 1]); % Dirección aleatoria del arañazo (0: horizontal, 1: vertical)
        if direccion == 0
            x = randi([1, columnas-longitud+1]); % Posición aleatoria en el eje x
            y = randi([1, filas]); % Posición aleatoria en el eje y
            mascaraAranazos(y, x:x+longitud-1) = 1;
        else
            x = randi([1, columnas]); % Posición aleatoria en el eje x
            y = randi([1, filas-longitud+1]); % Posición aleatoria en el eje y
            mascaraAranazos(y:y+longitud-1, x) = 1;
        end
    end

    % Generar máscara de manchas
    mascaraManchas = rand(filas, columnas) < 0.05; % Máscara de manchas
    % Aplicar filtro de convolución para suavizar las manchas
    mascaraManchas = conv2(double(mascaraManchas), ones(7) / 25, 'same') > 0.25;

    % Generar valores aleatorios para las imperfecciones
    valoresAranazos = rand(filas, columnas) * 0.7; % Valores de los arañazos
    valoresManchas = rand(filas, columnas) * 0.6; % Valores de las manchas

    % Aplicar las imperfecciones a la imagen
    imagenImperfecta = imagenImperfecta + repmat(mascaraAranazos, [1, 1, size(imagenImperfecta, 3)]) .* repmat(valoresAranazos,
[1, 1, size(imagenImperfecta, 3)]);
    imagenImperfecta = imagenImperfecta + repmat(mascaraManchas, [1, 1, size(imagenImperfecta, 3)]) .* repmat(valoresManchas,
[1, 1, size(imagenImperfecta, 3)]);



    % Ajustar los valores para que estén dentro del rango [0, 1]
    imagenImperfecta = max(0, imagenImperfecta);
    imagenImperfecta = min(1, imagenImperfecta);


    % Efecto de distorsión de color
    desplazamientoColor = randi([-7, 5], 1, 3); % Desplazamiento aleatorio para cada canal de color
    imagenImperfecta(:, :, 1) = circshift(imagenImperfecta(:, :, 1), desplazamientoColor(1));
    imagenImperfecta(:, :, 2) = circshift(imagenImperfecta(:, :, 2), desplazamientoColor(2));
    imagenImperfecta(:, :, 3) = circshift(imagenImperfecta(:, :, 3), desplazamientoColor(3));

    % Efecto de desenfoque
    sigmaDesenfoque = randi([1, 3]); % Valor aleatorio para el parámetro sigma del filtro de desenfoque
    imagenImperfecta = imgaussfilt(imagenImperfecta, sigmaDesenfoque);


end
```