Universidad de la República
Facultad de Ingeniería

Plataforma Abierta de Restauración de Películas

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Sebastián Bugna, Juan Andrés Friss de Kereki

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista, Ingeniero en Computación.

Tutores

Gregory Randall ..... Universidad de la República, IIE
Eduardo Fernández ..... Universidad de la República, INCO
Mauricio Delbracio ..... Universidad de la República, IIE

Tribunal

Isabel Wschebor ..... Archivo General de la Universidad, AGU
Ignacio Ramirez ..... Universidad de la República, IIE
Pablo Musé ..... Universidad de la República, IIE
Alvaro Martín ..... Universidad de la República, INCO
Libertad Tansini ..... Universidad de la República, INCO

Montevideo
viernes 28 de abril, 2017

Plataforma Abierta de Restauración de Películas
Sebastián Bugna, Juan Andrés Friss de Kereki

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 177 páginas.
Compilada el viernes 28 de abril, 2017.
http://iie.fing.edu.uy/

Resumen

La restauración digital de películas es un área de creciente interés. Muchas películas deterioradas son digitalizadas y remasterizadas en nuevos formatos de mayor calidad para su archivado, exhibición y distribución. Resulta de interés desarrollar métodos que faciliten el proceso de restauración mediante procesamiento de señales.

En Uruguay, el Archivo General de la Universidad (AGU) cuenta con decenas de películas nacionales de carácter histórico que fueron rescatadas en condiciones diversas de degradación y posteriormente digitalizadas, pero aún conservan desperfectos debido a su deterioro. Otras instituciones locales, como el Servicio Oficial de Difusión, Radiotelevisión y Espectáculos (SODRE) o la Facultad de Información y Comunicación (FIC), atraviesan situaciones similares, y en muchos casos el volumen de los acervos no amerita la inversión tecnológica implicada en la restauración. En el continente la situación se repite en todos los países.

El objetivo de este proyecto consiste en desarrollar una plataforma basada en software libre para la restauración de películas deterioradas previamente digitalizadas, que pueda ser útil en particular en los países del tercer mundo con dificultades de acceso a las herramientas de restauración comerciales, y también para permitir que grupos en otras universidades aporten otros módulos para restaurar diferentes desperfectos, enriqueciendo de esa manera la plataforma.

La plataforma desarrollada se denomina PARP, Plataforma Abierta de Restauración de Películas Deterioradas[1], utiliza la aplicación de postproducción Natron y la librería de *computer vision* OpenCV. Se incluye la implementación de algoritmos de postproducción de video para restaurar degradaciones específicas: detección semi-automática de cortes entre planos, corrección de *flicker* de luminancia, detección y restauración de *scratches*. Los algoritmos fueron diseñados para tener una interacción intuitiva y fluida con el usuario.

El estado actual del archivo y patrimonio audiovisual uruguayo impulsa a incursionar en nuevas ramas de investigación en el área. Archivólogos, bibliotecarios, ingenieros, cineastas, postproductores, programadores y catalogadores, entre otras especialidades, forman parte de una gran cadena de investigación y trabajo que distintas instituciones y archivos nacionales buscan profundizar a futuro.

[1]PARP se encuentra publicado en https://github.com/SebastianBugna/PARP

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

Aquellos nacidos en el siglo XX probablemente recuerdan los *videocassettes* o VHS. Sin embargo, hoy en día no nos encontraremos con ellos normalmente, ya que se acostumbra a filmar y almacenar videos y películas digitalmente, en nuevos formatos. La tendencia indica que se está dejando de usar el celuloide o material fílmico casi por completo. Reproducirlos genera una copia de menor calidad que la original, un problema que no afecta a los archivos digitales, donde una copia exacta es posible.

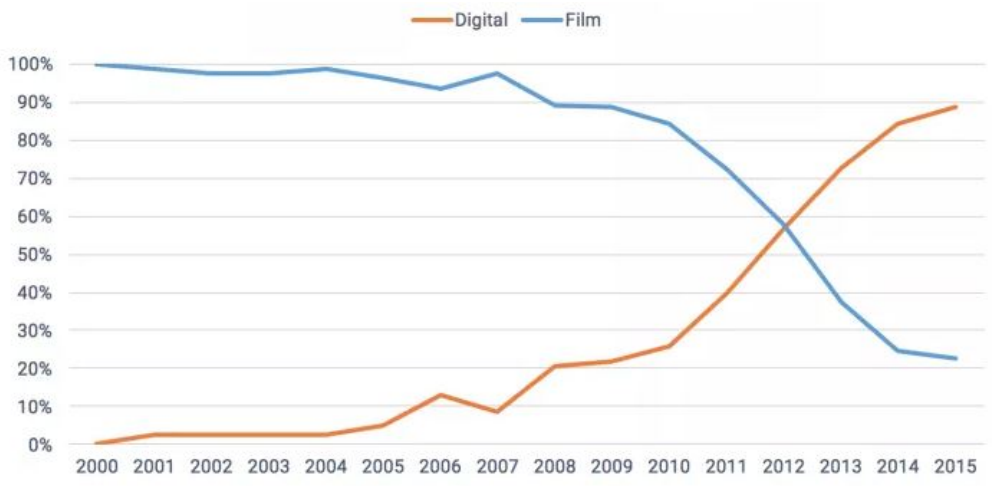


Figura 1.1: Formato de captura de las cien películas de mayor recaudación en Estados Unidos año a año. [1]

¿Pero qué sucede con las películas de celuloide con el paso del tiempo? El deterioro inevitable del celuloide provoca la pérdida de películas que son parte de una herencia cultural. Estas degradaciones incluyen suciedades que se adhieren al fílmico, roturas, *flicker*, rayones, pérdida de color y descomposición de la película, entre otros procesos irreversibles.

La restauración digital de películas es un área de creciente interés. Muchas películas deterioradas están siendo digitalizadas y remasterizadas en nuevos formatos de mayor calidad para su archivado, exhibición y distribución (HD, 2K, 4K, UHD, DCP, DPX, etc).

La restauración manual mediante técnicas de postproducción es tediosa y consume una inmensa cantidad de tiempo. A modo de ejemplo, "Viaje a la Luna"(1902) de Georges Méliès, cuya duración es de 15 minutos, tomó un año entero en ser restaurada con estas técnicas. Por esto, resulta de interés desarrollar métodos que faciliten el proceso de restauración de estos deterioros mediante procesamiento de señales.

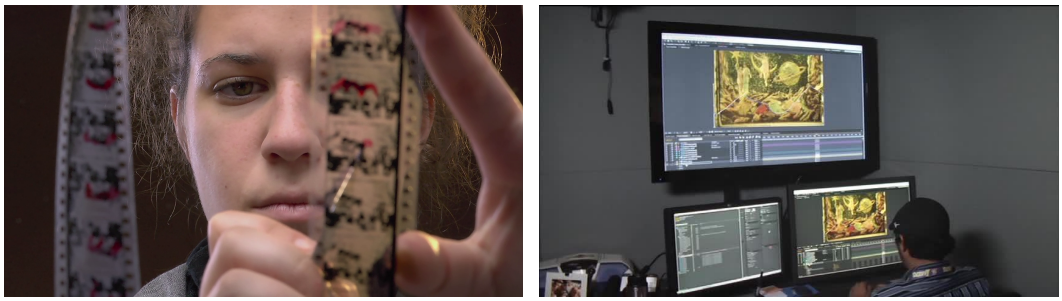


Figura 1.2: Fotografías del proceso de restauración de la película emblemática "Viaje a la luna" (1902), G. Méliès, con técnicas de restauración analógicas y digitales. Extraído del documental que registra dicho proceso"*Le Voyage Extraordinaire*" (S. Bromberg y E. Lange, 2011)[2].

En Uruguay, el Archivo General de la Universidad (AGU), cuenta con decenas de películas nacionales de carácter histórico que fueron rescatadas en condiciones diversas de degradación y posteriormente digitalizadas, pero aún conservan desperfectos debido a su deterioro [3].

Otras instituciones locales, como el Servicio Oficial de Difusión, Radiotelevisión y Espectáculos (SODRE) o la Facultad de Información y Comunicación (FIC), atraviesan situaciones similares, y en muchos casos el volumen de los acervos no amerita la inversión tecnológica implicada en la restauración. En el continente la situación se repite en todos los países.

El principal objetivo de este proyecto consistió en desarrollar una plataforma de restauración digital de películas basada en software libre, modular, que ayude a los restauradores en su proceso de trabajo. En particular, puede ser de gran ayuda en instituciones de países del tercer mundo con dificultades de acceso a las herramientas de restauración comerciales. Otro objetivo es permitir que grupos en otras universidades aporten otros módulos para restaurar diferentes desperfectos.

Nos proponemos realizar una búsqueda de diferentes métodos computacionales, así como explorar distintos algoritmos de restauración digital, con el objetivo de programar una interfaz gráfica abierta que permita visualizar, restaurar y exportar una película.

Se parte de una versión digital de la película, y se permite al usuario aplicar técnicas de procesamiento de video para corregir las distintas degradaciones. La interfaz gráfica facilita la toma de decisiones así como la visualización de los resultados intermedios en el proceso de restauración.

Las opciones de software de restauración digital en el mercado tienen licencias muy costosas. En general los costos resultan prohibitivos para la mayoría de las instituciones y archivos de la región. Por ello resulta de interés montar y dejar en funcionamiento una plataforma libre y abierta, con buenas condiciones de reusabilidad y evolucionabilidad, con una comunidad participativa y una documentación sólida que permitan el acercamiento de nuevos desarrolladores.

Como criterio de éxito en el proyecto, nos propusimos restaurar con la plataforma desarrollada al menos dos de los problemas típicos de degradación de películas, y poder instalar y utilizar la plataforma sobre computadoras que utilicen sistema operativo libre Linux.

El Archivo General de la Universidad (AGU) suministró películas deterioradas ya digitalizadas, que fueron elegidas por el equipo de trabajo. Dichas películas presentan otros desperfectos, además de los abordados en esta tesis.

Tempranamente el alcance de este proyecto implicaba la implementación de cuatro algoritmos de restauración. Este objetivo fue modificado al conocer las complejidades específicas de la integración entre tecnologías de video, así también como la escasez de implementaciones libres y disponibles de algoritmos de restauración digital de películas deterioradas.

Capítulo 2

Contexto y Marco Teórico

En esta sección se hace un repaso sobre diferentes recursos e informaciones relevantes en torno a la temática de restauración digital de películas deterioradas.

En cuanto al contenido, primero se presentan algunas características del material fílmico, así como los distintos estándares de formatos. Luego se repasan las principales degradaciones asociadas a estos formatos. También se presenta el contexto actual del archivo audiovisual nacional y los diferentes archivos existentes, y se señalan las características de las tecnologías de digitalización disponibles, con el objetivo de analizar la naturaleza de los archivos digitales que se desea restaurar.

Además, se plantea el punto de vista del Archivo General de la Universidad (AGU) con respecto a la restauración de películas conservadas en su acervo. También se presentan opciones, basadas en software libre y software propietario, disponibles para realizar diferentes tareas dentro de lo que comprende la restauración digital de películas. Finalmente se hace una conclusión sobre el contexto nacional y los recursos que existen en el mercado para realizar tareas de restauración.

2.1. Características de la película de celuloide

El celuloide es un material termo-plástico sintetizado, utilizado durante todo el siglo XX, principalmente por la industria cinematográfica como materia prima para crear material fílmico en formatos estándar. El uso del celuloide permite filmar, copiar, proyectar y distribuir las películas.

La película se comercializa en un rollo o tira de acetato transparente sobre el cual se adhiere, a través de un adhesivo, una emulsión gelatinosa que contiene cristales microscópicos de haluros de plata que son sensibles al impacto de la luz. Las características de estos cristales definen aspectos de la película como son la sensibilidad, el contraste y la resolución.

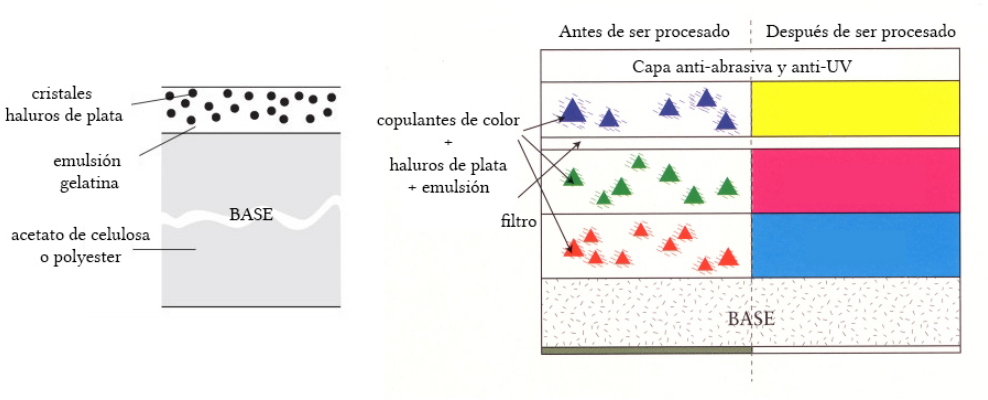


Figura 2.1: Estructura básica de la película de acetato de celulosa o poliéster. Izq. blanco y negro [4], Der. color [5].

Desde los orígenes del cine, los estándares de formatos y sus características físico-químicas han cambiado varias veces. La forma más usual de reconocerlos es por el ancho de la tira de celuloide, también relacionado a la resolución de la imagen proyectada, así como las medidas específicas de los fotogramas que constituyen la secuencia de imágenes. [1]

Las presentaciones más usuales se pueden ver en la Fig. 2.2. Una lista más exhaustiva de los formatos estándar se presenta a continuación:

- 70mm, 65mm - Formato con mayor resolución que el estándar más familiar 35mm. La película se filma en la cámara en 65 mm y luego se transfiere al formato de 70mm para su proyección, donde se agrega una banda de 5mm con la información de sonido. Es utilizado actualmente en el formato estándar de cine IMAX, pero ha estado presente desde los orígenes del cine [11].
- 35mm - Estándar desde 1909 para filmar, proyectar y archivar películas de cine.
- 28mm - 1923, abarata el 35mm para uso amateur. Fue sustituido rápidamente por el 16 mm.
- 16mm - 1923, Kodak. estándar en 1932. Abarata el 35mm para uso amateur.
- 9.5mm - 1922, Pathé, para copia, proyección de películas y uso doméstico.
- 8mm - 1930, Kodak. Opción de menor costo que 35mm para realización amateur.
- Super 8 - 1965, Kodak. Mantiene el tamaño 8mm pero tiene mayor resolución de imagen y usa perforaciones más pequeñas.

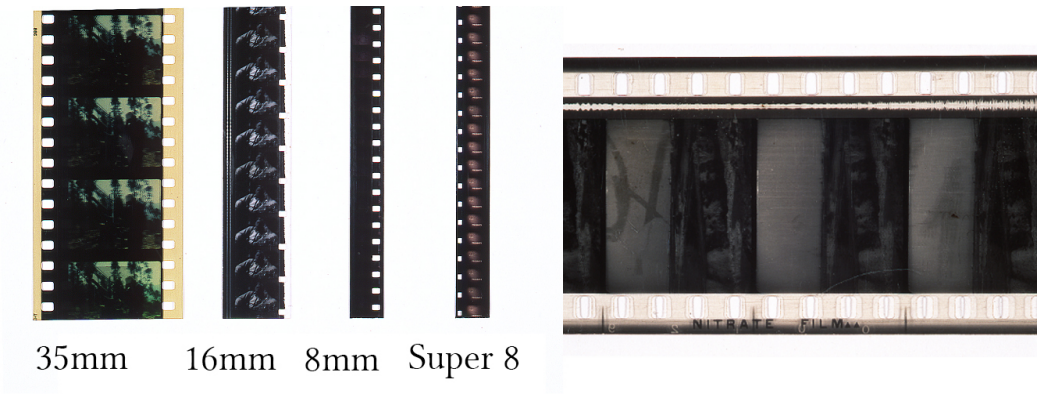


Figura 2.2: (a) Estándar de formatos película analógica (b) Detalle de tres fotogramas en formato 35mm [6].

Un segundo elemento importante para clasificar los formatos del material fílmico es el material utilizado para la base o sustrato transparente sobre el que se adhiere la emulsión con sales de plata sensibles a la luz. Hay tres tipos principales según el sustrato:

- Nitrato de celulosa - usado para producir: 70mm, 65mm, 35mm. Es autoinflamable, y fue prohibido en Europa y Estados Unidos entre 1940 y 1950.
- Acetato de celulosa (incluye di-acetato, tri-acetato) - usado para producir: 35mm, 16mm, 9,5mm, 8mm, Super 8. Sufre el síndrome del vinagre.
- Polyester (Mylar, Ester) - usado para producir: 35mm, 16mm, 8mm y Super 8.

2.2. Degradación de la película analógica

Un diccionario que releva todos los problemas posibles asociados a películas de celuloide deterioradas se puede encontrar en *"The Brava broadcast archive programme impairments dictionary"* creado por BRAVA[12] [2]. La información que se presenta a continuación sobre las distintas degradaciones se basa en dicho archivo y discusiones compartidas con el AGU.

Grano de película Superposición de los granos de la película (cristales de haluro de plata procesados) que se encuentran a diferente profundidad en la emulsión. Presentan patrones espaciales aleatorios, y pueden presentar pérdida del foco o nitidez de la imagen y errores en la densidad de la película. Este no es un deterioro que surge con el paso del tiempo, sino una característica del sistema de registrado y siempre está presente.

[1] Información más específica sobre identificación de formatos de películas analógicas puede encontrarse en los artículos de referencia [6], [7],[8], [9], [10].

[2] BRAVA es un proyecto de restauración de video que se presenta en la Sección 2.6.

Suciedades (polvo, manchas, pelos, ruido *"dirt & sparkles"*) Puede presentarse en la superficie, o adherirse o incrustarse en el fílmico. Puede haber sido impreso desde la película original o haberse agregado posteriormente en copias. En general este tipo de ruido dura un único fotograma, o al menos es impulsivo temporalmente.

Como se muestra en la Figura 2.3a, son artefactos localizados y relativamente pequeños que borran información de la película original. Pueden ser totalmente opacos o semitransparentes, conservando algo de la información original de la película.

Rayones o *scratches* Se presentan como líneas prácticamente verticales, y persistentes en varios fotogramas consecutivos. Pueden ser brillantes u oscuros o afectar el color, según la profundidad del *scratch* y si éste se encuentra en la emulsión o en el sustrato del formato analógico[13].

Existen diversas causas: pueden ser generados por la fricción con piezas mecánicas en los mecanismos de reproducción de la película, o bien porque la película es arrastrada contra sí misma y cualquier irregularidad o suciedad en el carrete raya la cinta (ver Fig.2.3b).

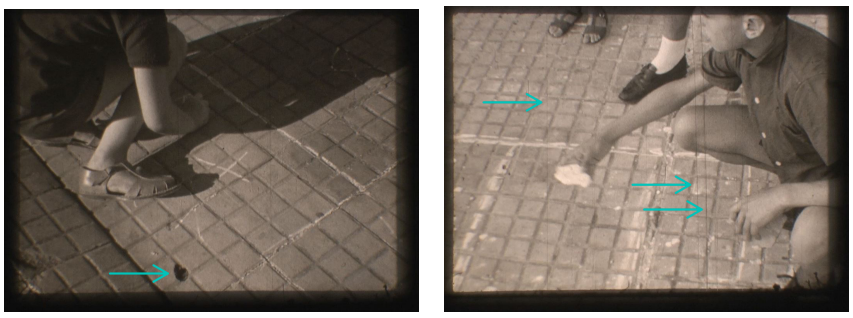
Degradaciones de color Incluyen variaciones no naturales del color, desaparición y/o desvanecimiento del color original de la película, entre otros. Estos desperfectos pueden ser no uniformes con el tiempo, y pueden ser locales espacialmente (solo aparecen en algunas regiones de la imagen) o abarcar gran parte del material fílmico.

Vibración de la imagen Las desestabilizaciones presentes en películas deterioradas digitalizadas se deben a razones mecánicas en los sistemas de digitalización (*e.g.* un mecanismo de arrastre poco preciso). Se puede presentar como vibraciones en la posición y/o rotación de la imagen, así también como deformaciones no uniformes temporalmente y localizadas espacialmente.

Flicker de iluminación Variaciones no naturales en la intensidad de luminosidad de la secuencia. Puede ser provocada por la degradación no uniforme de la película. También puede originarse durante la digitalización. En este casos el desperfecto puede presentar periodicidad temporal *e.g.* debido a una mala operación de la velocidad de obturación de la cámara con la cual se digitaliza.

2.2.1. Síndrome del vinagre

Consiste en un proceso de deterioro propio de las películas de acetato de celulosa en malas condiciones de humedad y calor. La película libera ácido acético que la descompone y le da su característico olor avinagrado. Es altamente tóxico, irreversible y el proceso se disemina rápidamente entre películas a través del aire.



(a) (b)

Figura 2.3: Fotograma de "Juegos y Rondas", 1968, Rodolfo V. Tálice. (AGU)
Señalados en (a) *Blotch* (b) *Scratches*

El acetato de celulosa, o *safety film*, sustituyó la película de nitrato de celulosa, que era auto-inflamable. Sin embargo debido a su defecto químico el acetato de celulosa fue posteriormente sustituido por la película de poliéster.

El proceso presenta distintos grados e instancias progresivas de deterioro. En un principio la película libera ácido acético; más adelante el acetato plástico de la película se vuelve quebradizo y pierde su flexibilidad y elasticidad. El acetato se contrae y se separa de la emulsión de la película. En grados más avanzados la emulsión se cristaliza y libera burbujas líquidas de aditivos de la base plástica de la película, y en las películas de color se tiñe la película hacia el color rosado o el azul. En estos casos, la película ya no puede ser proyectada por medios mecánicos.

La humedad y el calor, así como los medios anaeróbicos, son los principales factores que provocan este proceso. Por esto a su vez las películas deben ser preservadas en condiciones controladas y estables de temperatura y humedad. En caso de estar muy deterioradas son apartadas o descartadas para no desencadenar el proceso en otras películas. Actualmente no existen medios para detener ni revertir el curso del deterioro.

Figura 2.4: Acetato de celulosa deteriorado por la liberación de ácido acético [14].

2.3. Conservación de materiales audiovisuales

En el documento "Buenas prácticas en archivos históricos de servicios universitarios"del AGU[15], se marcan pautas para la conservación de materiales audiovisuales, en torno a la investigación histórica del acervo del Instituto de Cinematografía de la Universidad (ICUR). Dichas pautas pueden ser aplicadas en cualquier institución que custodie este tipo de documentación.

La política de conservación propuesta apuesta a un tratamiento integral del acervo. Se respetan las recomendaciones de la FIAF (*International Federation of Film Archives*) [16] en cuanto a conservación y custodia de materiales audiovisuales, y la norma *ISAD-G* [17] en el área de investigación histórica.

El procedimiento se divide en tres etapas:

1. Inspección y acondicionamiento físico de las películas
2. Identificación y descripción de características fisico-químicas
3. Política de conservación preventiva

Esta última etapa implica la limpieza y control de los deterioros, control de las condiciones de proyección de los materiales, el almacenamiento, y la climatización y control ambiental de un depósito. Finalmente la preservación consiste en la preparación y acondicionamiento de los materiales para su digitalización, que en este caso se realizó por medio del sistema de digitalización *telecine*[3].

2.4. Archivo y patrimonio audiovisual en Uruguay

2.4.1. Contexto actual

En 2011 se llevó a cabo una consultoría por parte de Isabel Wschebor, Julieta Keldijan y Ana Laura Cirio, para el Instituto del Cine y Audiovisual Uruguayo (ICAU), donde se obtuvo un primer diagnóstico de la situación del patrimonio audiovisual nacional[18].

La investigación buscó obtener un mapa de las instituciones públicas y privadas, y actores que tuvieran bajo su custodia patrimonio audiovisual. Las autoras relevaron la situación de preservación del patrimonio fílmico nacional y se investigaron, en contraste, el contexto en diversos países de Iberoamérica.

[3]En la Sección 2.5.3 se describe el proceso de telecinado.

Los resultados del informe expresan una precariedad extrema en las condiciones de conservación general de los archivos nacionales. Con excepción de Cinemateca, las instituciones no cuentan con depósitos climatizados y con control ambiental adecuados.

Tampoco existen formaciones académicas específicas sobre conservación de fotografías y películas y/o archivos sonoros, ni políticas de gestión de recursos humanos. Esto muestra como dificultad un panorama heterogéneo de actores y personas a cargo del acervo patrimonial, con conocimientos y metodologías desparejas.

En muchos casos, el volumen de los acervos no amerita la inversión implicada en las tecnologías necesarias, a pesar de su valor histórico. Es necesaria la formación de redes interinstitucionales en este ámbito para la socialización de conocimiento y conceptos, y además para obtener el acceso a tecnologías de transferencia y reproducción del archivo audiovisual.

En 2014 se crea el "Compromiso Audiovisual 2015-2020"[19], bajo la coordinación general de ICAU, la Oficina de Locaciones Montevideanas (OLM), la Asociación de Productores y Realizadores de Cine del Uruguay (ASOPROD) y con la participación de más de 300 personas vinculadas al sector audiovisual, donde se aborda como uno de los ejes estratégicos el patrimonio audiovisual. Se propone trabajar sobre el concepto de un Sistema Nacional de Archivos Audiovisuales, en el que tengan cabida tanto los archivos públicos como los privados, de cine y televisión. Se propone la instrumentación de un sistema integrado de rescate, conservación y acopio del patrimonio nacional audiovisual para garantizar su preservación y acceso. Esto se enmarcó en el reciente Sistema Nacional de Archivos, creado mediante la Ley de Archivos No. 18220 en 2007 [20].

Varios artículos y algunas entrevistas recientes a diferentes actores en estas temáticas están disponibles en la publicación del Cine Universitario "Revista Film"[21] y [22].

2.4.2. Archivos nacionales

A continuación se resumen los diferentes archivos nacionales. Esta información fue recabada de la publicación del Cine Universitario "Revista Film"[23], que profundiza en este tema. En el caso del AGU, se utilizó como referencia su libro "Buenas prácticas en archivos históricos de servicios universitarios"[15]. La siguiente lista no es exhaustiva; existen diferentes archivos personales, privados y exclusivos que seguramente no estén considerados en este sondeo.

Cinemateca Uruguaya

Cinemateca fue fundada en 1952 como Asociación Civil sin fines de lucro. Su objetivo es contribuir al desarrollo de la cultura cinematográfica y artística en general. Forma parte de la FIAF. Es fundador de la Coordinadora Latinoamericana de Archivos de Imágenes en Movimiento (CLAIM) y miembro asesor del ICAU. Su archivo audiovisual fue declarado monumento histórico nacional, por iniciativa de la Comisión de Patrimonio y del ICAU.

La creación temprana del archivo permitió la adquisición de copias propias de películas internacionales y gran parte de la producción nacional, que facilitaron la difusión de películas y mayor relacionamiento con diferentes cinematecas del mundo.

El archivo contiene aproximadamente 22 mil títulos en total en los formatos 35mm, 16mm, Super-8, poliéster, Betacam [24], VHS (*Video Home System*) [25], U-Matic [26] y DVD (*Digital Versatile Disc*) [27], y un acervo de 3.000 piezas como cortometrajes, documentales y noticieros nacionales. El esquema de trabajo del archivo se basa en la búsqueda, clasificación, catalogación y duplicación de los materiales.

SODRE, Archivo nacional de la Imagen y la Palabra [28]

Dependencia del Ministerio de Educación y Cultura (MEC) y del Servicio Oficial de Difusión, Radiotelevisión y Espectáculos (SODRE). Nace en 1943 como cineteca, cuya función es documentar y salvaguardar películas nacionales y extranjeras. Tras el incendio del SODRE en 1972, se pierde gran parte del material.

El acervo actual está conformado por 5 mil títulos extranjeros y nacionales en 35mm y 16mm; 2 mil videos en U-Matic, VHS y DVD, además de grabaciones y negativos fotográficos.

AGU. Archivo General de la Universidad de la República [3]

El AGU cuenta con un acervo de casi 800 películas, 12 mil fotografías, y una biblioteca especializada en cine y fotografía científica. Desde el año 2007 custodia la producción del ICUR (Instituto de Cinematografía de la Universidad), que funcionó entre 1950 y 1973. [4]

Universidad Católica del Uruguay. Archivo audiovisual "Prof. Dina Pintos" [29]

Se trabaja siguiendo normas y estándares de preservación, con la limitante de los recursos disponibles. La bóveda donde se almacena el acervo está deshumificada pero no climatizada y no se realizan controles de temperatura y humedad.

[4] Se presenta específicamente el trabajo del AGU en la siguiente sección.

IM. Archivo de Prensa y Comunicación de la Intendencia de Montevideo

Se crea en 1990 por la necesidad de registrar actividades que apoyen la gestión y sean parte de la memoria audiovisual de Montevideo. El archivo está constituido por 6 mil videos (en formatos digitales y magnéticos) de duración aproximada de siete minutos que registran gestiones de los gobiernos departamentales, convenios, conferencias, obras públicas, entre otros.

AGADU (Asociación General de Autores del Uruguay), Museo y Centro de Documentación

En su reapertura en 2008 comienza la organización de su colección audiovisual. Consiste de 150 DVDs y 27 VHS digitalizados y clasificados que tienen relación con actividad autoral en nuestro país (películas, documentales, conciertos, teatro, entre otros).

FAC. Archivo de la Fundación de Arte Contemporáneo

La Fundación de Arte Contemporáneo es un colectivo autogestionado de artistas formado en 1999, que realiza acciones vinculadas al *videoarte* y la *videoinstalación*. El laboratorio de Cine FAC tiene como centro el archivo, la conservación y preservación, apelando al trabajo conjunto de los artistas. El acervo se compone de varias piezas de *videoarte* histórico, *performances, videoinstalaciones, video escultura,* entre otros, en formatos VHS, U-matic, DVD, 16mm y Super-8.

Tevé CIUDAD

Surge con la fundación del canal en 1996, la necesidad de sistematizar y conservar los contenidos propios del canal. Además el archivo cuenta con contenidos que devienen de las líneas de fomento audiovisual nacional (ICAU, FONA, etc). Contiene en su acervo más de 9 mil horas de archivo que han sido declaradas Patrimonio Cultural de la Nación por el Ministerio de Educación y Cultura.

Canal 10

El archivo del canal 10 tiene origen en los años 80 con los primeros formatos magnéticos de video (U-matic, Betacam, VHS, DV). Gran parte del archivo se digitalizó en cintas LTO (*Linear Tape-Open*) generación 4 y 5 [30], y el resto se ha descartado o reciclado. La tecnología LTO ofrece cintas magnéticas de alta capacidad que logran almacenar en el orden de *terabytes* por cinta (50 a 60 horas de video). El material se encuentra en condiciones controladas de temperatura y humedad. Está catalogado y se puede visualizar a través de una interfaz gráfica de usuario.

Teledoce

El archivo surge de la necesidad del canal de conservar contenidos emitidos para realizar nuevas ediciones. Se conservan algunos originales en condiciones de humedad y temperatura controladas. Además de la producción nacional, el acervo del canal incluye producciones extranjeras.

2.4.3. Archivo General de la Universidad de la República

El Archivo General de la Universidad de la República (AGU) trabaja interdisciplinariamente en la preservación de materiales audiovisuales. Desde el 2007 el AGU custodia el acervo del ICUR que funcionó desde 1950 hasta su intervención en 1973.

Este archivo incluye casi 800 películas y 12 mil fotografías, así como una biblioteca especializada en cine y fotografía científica. Se trata de un acervo único para estudiar el papel de la Universidad de la República en la historia del arte, los medios, la ciencia y la tecnología, dado que ICUR fue la primer institución abocada a la producción de cine científico y pedagógico, y fue un espacio de innovación a nivel nacional en lo que hace al cine documental de interés social y político.

Las cintas del ICUR no fueron conservadas ni reproducidas en condiciones ambientales y tecnológicas favorables, y presentan todo tipo de degradaciones. Esto implicó un debate sobre los criterios técnicos y las líneas de trabajo que posibilitarían la conservación y digitalización de las películas, por parte del AGU.

Se constató el mal estado de una gran porción de las películas, separando dicha porción del resto de la colección, y descartando los documentos irrecuperables. Se realizaron procedimientos de limpieza manual para acondicionar el material, además de mantenerlo en condiciones ambientales estables y controladas.

Desde 2011 se incorporó al AGU un equipo de trabajo abocado específicamente a las tareas de conservación y digitalización del laboratorio, lo que permitió la creación de una cadena de trabajo orientada a la conservación y restauración integral del archivo.

En esta etapa se estabilizaron los materiales fílmicos, restaurando su soporte analógico. También se realizó un telecinado de las películas, lo que permite el acceso y archivado digital de las cintas.

Para el enriquecimiento de este trabajo por parte del AGU, es imprescindible la colaboración interdisciplinaria que desarrolle líneas de investigación que trasciendan los procedimientos técnicos e incursionen en diferentes dimensiones asociados a los aspectos archivísticos, históricos y tecnológicos de las imágenes.

Por más información véase el documento "Buenas prácticas en archivos históricos de servicios universitarios"del AGU [15].

2.5. Digitalización

Una de las etapas más importantes en la tarea de recuperar un archivo fílmico es su conversión analógico-digital. El soporte foto-químico de la película se convierte en imagen electrónica. Esto permite conservar una copia digital de la película para poder archivarla, restaurarla y exhibirla.

Existen distintas tecnologías para digitalizar material fílmico. La digitalización es costosa e involucra diversos factores de decisión. En general se evalúan los compromisos entre el costo de las inversiones y el volumen y/o la trascendencia del material que se desea digitalizar.

Hoy en día la producción audiovisual es esencialmente digital. Por ende las tecnologías de conversión analógico-digital disponibles son más escasas. Sin embargo, como se menciona a continuación, algunas opciones siguen en activo desarrollo.

Vale la pena mencionar que la digitalización es en sí misma objeto de debate para especialistas en conservación de material fílmico. Algunos expertos plantean que el único método aceptable para preservar el material es la transferencia fílmico a fílmico, debido principalmente a la inestabilidad de los formatos digitales actuales [31].

2.5.1. Breve repaso histórico

A causa de la enorme popularidad de la televisión entre 1940 y 1950, surgió la necesidad de retransmitir y realizar copias rápidas de las emisiones que eran, hasta entonces, únicamente en vivo. Esto dio origen al *Kinescope* [32], una versión pionera del *telecine*. Una cámara filmaba la emisión directamente desde la pantalla de un monitor y esto permitía gestionar las copias.

Desde entonces las opciones tecnológicas de telecinado se mantuvieron evolucionando y adecuando a las necesidades de la industria, y el principio básico se mantiene.

Por otro lado, en la década de los 90 se hace popular en producciones cinematográficas, el flujo de trabajo *digital intermediate* (DI), proceso mediante el cual se realiza la digitalización con *scanner* de la película analógica para aplicar efectos digitales, CGI (*Computer Generated Imagery*), composición y retoque de color, y luego se vuelve la película a su formato analógico original, típicamente 35mm. Hasta mediados del 2000, debido a su éxito, la demanda de DI creció rápidamente, y se produjeron *scanners* de alta calidad que pueden incluso realizar tareas de postproducción digital.

2.5.2. Estándares de captura digital

Los proyectos de digitalización de archivo audiovisual normalmente implican requerimientos esenciales sobre la calidad de imagen. A su vez se deben respetar protocolos determinados que favorezcan a los distintos puntos de la cadena de trabajo.

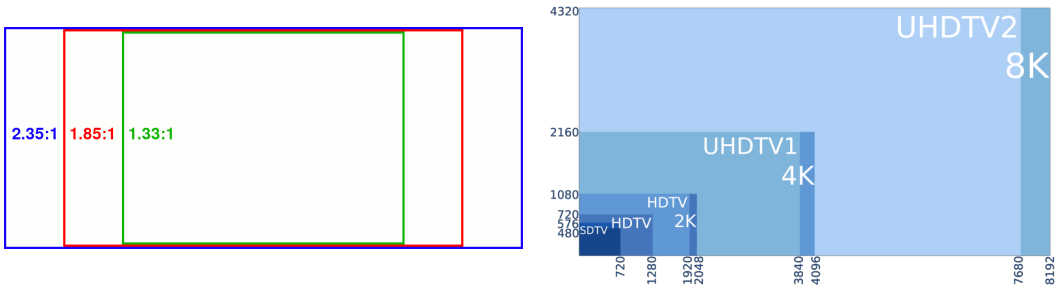
Se recomienda por ejemplo digitalizar a la máxima resolución y calidad disponible de acuerdo al formato analógico, para evitar redigitalizar y volver a manipular la película en el futuro, entre otras cosas (ver [33] y [34]).

Resulta imprescindible tener en cuenta estos aspectos técnicos durante la etapa de restauración digital, posterior al digitalizado. Algunos de ellos se resumen a continuación.

Relación de aspecto

Se define como el ancho de la pantalla en relación al alto de la misma, y se utilizan estándares de exhibición y captura. Existes las normas ISO y ASA que definen diferentes relaciones de aspecto nominales estándares[35].

El estándar 16:9 propuesto por el SMPTE *(Society of Motion Picture and Television Engineers)* se ha convertido en la relación de aspecto por defecto de los videos de formato HD o alta definición, porque permite integrar la relación de aspecto estándar 4:3 (televisión analógica) y formatos de cine "anamórficos" como 2:35:1 (*CinemaScope*) en la misma pantalla, agregando barras negras a los lados de la imagen, como se muestra en la Figura 2.5.



(a) (b)

Figura 2.5: Algunos formatos estándares de películas (a) Ejemplos de relaciones de aspecto [36], (b) Ejemplos de resoluciones, las medidas están dadas en píxeles [37]

Resolución

La resolución SD *(Standard Definition)* es el estándar de transmisión de video con relación de aspecto 4:3. Sus dimensiones son 720×486 para la norma NTSC y 768×576 para PAL y SECAM. En general en estos formatos la imagen se conforma por dos campos (par e impar) entrelazados y superpuestos que logran la sensación de movimiento adecuada durante la reproducción de un video. Estos formatos producen, en general, diversos desperfectos durante el proceso de restauración digital debido al escaneo entrelazado de la imagen. Actualmente esta técnica es sustituida por el escaneo progresivo, donde la imagen se lee secuencialmente en líneas horizontales y se compensan los defectos del entrelazado.

El desarrollo de las tecnologías de cámaras digitales permite actualmente resoluciones mayores a Full HD (1080×1920). Esto modifica a su vez los formatos de exhibición.

El formato 1080p (el índice *"p"* por escaneo progresivo) o Full HD, con relación de aspecto 16:9, es un nuevo estándar de alta definición, aunque actualmente está siendo sustituido por el formato 2K (2048×1080) en televisión digital y producciones cinematográficas.

Los formatos UHD (*Ultra HD*) también son utilizados en el campo profesional, incluyendo las resoluciones 4K (4096×2160) y 8K (7680×4320).

En los próximos años se estima que las tecnologías trabajen en resoluciones desde los 6K hasta los 16K (ver Fig. 2.5). Los archivólogos en general acuerdan en que 4K es suficiente para propósitos de archivado y reutilización[38].

Profundidad de color (*bitdepth*) y espacio de color

El *bitdepth* o profundidad de color de una imagen, determina la cantidad de memoria destinada por cada píxel para cada canal de color de la imagen. Por ende determina la precisión en la información de color.

Los formatos de video digital actuales utilizan un mínimo de 8-bits por canal de color, es decir 256 valores (2^8) por cada canal RGB, o sea que es posible representar en una imagen más de 16 millones de colores ($256^3 = 16{,}777{,}216$ colores). Actualmente se utilizan también capturas mayores, de 10, 12 y 16 bits.

En los detalles de un modelo de *scanner* comercial se especifican comúnmente (entre otros) estos dos valores: el *bitdepth* y el espacio de color *i.e. 10-bit log, 16-bit linear*. El espacio logarítmico (*log*) se utiliza con ciertos negativos porque provee una mejor respuesta a las zonas oscuras de la imagen y un mayor rango dinámico [39].

Submuestreo de crominancia (*Chroma subsampling*)

Esta técnica de compresión se realiza utilizando los espacios de color Y'CbCr [40]. Estos espacios utilizan tres canales para representar la información de color: un canal de luminancia (Y') y dos canales de crominancia (Cb *blue difference*, y Cr *red difference*).

Nuestra percepción visual tiene mayor sensibilidad a la luminosidad que a los colores, debido a una mayor cantidad de células cono que de células bastones en la retina del ojo humano. Por ello se submuestrea información de color Cb y Cr, y se mantiene la información en el canal de luminancia Y', de esta forma se comprime información de la imagen con resultados imperceptibles para el ojo humano[41]. El indicador *chroma subsampling ratio* relaciona el índice de muestreo de luminancia (Y') en referencia a los canales Cb y Cr respectivamente.

Los *scanners* de calidad superior trabajan en 4:4:4. Esto indica que no hay un submuestreo entre el espacio RGB y Y'CbCr, por ende no hay pérdidas de información. También existen equipos que trabajan en 4:2:2 y 4:2:0. No hay submuestreo en luminancia (Y'), y por cada cuatro píxeles de luminancia hay dos muestras de Cb, Cr o dos muestras de Cb y ninguna muestra de Cr, respectivamente.

Formatos de captura utilizados

Las tecnologías de escaneo almacenan la captura como secuencia de imágenes fijas sin sonido. En general ofrecen gran flexibilidad en el almacenamiento de la información de color y el manejo de los espacios de color, lo que optimiza el flujo de trabajo. A su vez permite ingresar información variada de *metadata* a los archivos. Utilizan formatos adecuados para realizar copias digitales *masters* a partir de los originales en formatos analógicos. Los formatos típicos son:

- ***DPX (Digital Moving Picture Exchange)*** [42]
  Estándar en la industria cinematográfica utilizado por la mayoría de los *scanners* desde 2004. Creado por SMPTE, miembro de ANSI *(American National Standards Institute)*. No comprime la imagen capturada por el *scanner*, y resulta ventajoso para no perder información durante el proceso.

- ***TIFF (Tagged Image File Format)*** [43]
  De Adobe Systems desde 2009. Es compatible con la mayoría de las aplicaciones de *image processing* y postproducción de video. Permite capturar imágenes sin compresión de datos.

- ***Cineon*** [44]
  Similar al formato DPX, pero la estructura del formato está diseñado para video digital y no es tan flexible como DPX. Creado por *Eastman Kodak* para el *scanner Cineon* en 1993. No continúa en desarrollo, pero aún es elegido por algunas aplicaciones de efectos visuales. Fue utilizado en 1993 en la restauración del clásico de Disney "Blanca nieve y los siete enanitos"de

1937. Esta fue la primera película digitalizada con *scanner* en 4K y 10-bit. La versión remasterizada se distribuyó en 35mm, DVD y BlueRay [45].

Por otro lado, las tecnologías de telecinado trabajan con archivos de video en los formatos estándares de captura de las cámaras utilizadas. Actualmente se pueden utilizar formatos de alta definición (FullHD, 2K, UHD) con distintas opciones de compresión.

Los archivos de video sin compresión ni procesamiento (*RAW*, sin compresión) pueden ser demasiado voluminosos si se va a digitalizar un volumen considerable de películas, y esto implica una mayor inversión en tecnologías de almacenamiento y postproducción. Por ello se utilizan compresiones sin pérdidas que acotan el peso del archivo y minimizan la pérdida de información *e.g. MPEG-2/D10, Apple ProRes, DVCPro, H264* [46].

2.5.3. *Telecine*

Gran parte de los dispositivos de *telecine* funcionan proyectando luz a través de la película analógica. Ésta es dirigida, a través de espejos, hacia el objetivo de una cámara. En general la salida es en definición estándar (SD) y algunos ofrecen video de alta definición. Esta tecnología fue desarrollada en la década de los 70 y hasta comienzos del 2000, y por ende hoy en día debe ser reconfigurada y rediseñada para ser compatible con las cámaras digitales actuales.

Un aspecto complejo del proceso de telecinado es la sincronización de la tasa de fotogramas por segundo (fps) o *framerate* mecánico de proyección, con la captura de la señal de video digital. Se tienen mayores complicaciones cuando el *framerate* de salida del archivo digital es necesariamente distinto al *framerate* original de la película analógica. Esto era común en la retransferencia entre televisión (25 fps, 29.97 fps) y cine (24 fps). Se generan diversos desperfectos *e.g. screen tearing*[5]. Muchas técnicas fueron desarrolladas para evitar los desperfectos producidos en el proceso de telecinado, entre ellas las conocidas como *pulldowns* [47].

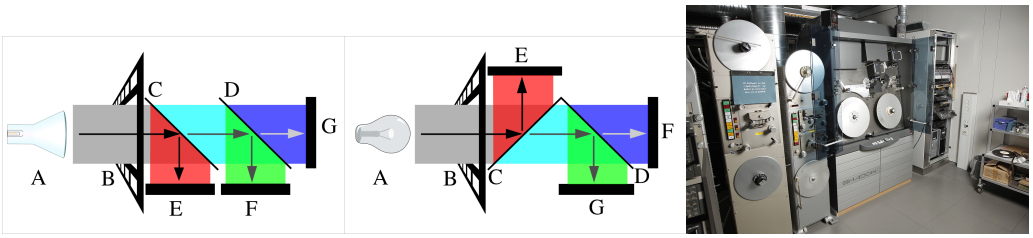
Otra desventaja asociada a algunos *telecines* es la deformación de la imagen. Esto se debe a que la película deteriorada puede no estar completamente tensa en la ventanilla que la sujeta. A su vez, el sistema de espejos es sensible a las perturbaciones de movimiento, y esto deviene en inestabilidades y vibración en la secuencia capturada.

[5]Esto ocurre cuando en cierto fotograma se muestra información de otros fotogramas.

Existen diversas tecnologías de hardware disponibles para telecinado:

- *Flying spot scanner* (FSS): utiliza una fuente de luz CRT (tubo de rayos catódicos) y espejos dicroicos para capturar los canales de color por separado.
- Arreglo lineal de CCD: utiliza luz blanca de xenón y proyecta la película por un prisma que separa la imagen en canales RGB. Cada canal es capturado por un sensor CCD diferente, como se muestra en la Figura 2.6.

La diferencia fundamental entre ambos es que la luz de rayos catódicos CRT es colimada, sus rayos son paralelos, y esto enfatiza la textura de los *scratches* y las suciedades durante la captura. La luz de la tecnología CCD, en cambio, es difusa y reduce el ruido y los *scratches* en cierta medida [38].



(a) (b) (c)

Figura 2.6: Esquemas de caminos ópticos (a) CRT o FFS *telecine* con espejos dicroicos y foto-multiplicadores (b) CCD *telecine* con prisma o espejos dicroicos y sensores CCD [48] (c) Shadow CCD *telecine* [49]

2.5.4. Datacine

Tecnología híbrida entre *scanner* y *telecine* introducida en 1996. Puede reproducir películas en estándares SD y HD en tiempo real (24, 25, 29.97 fps) progesivos y entrelazados. Tiene salida de datos DPX, similar a un *scanner*. Utiliza un arreglo de sensores CCD para la captura, se diferencia del *telecine* dado que la separación en canales de color se realiza en el mismo sensor CCD. Ofrece prestaciones de corrección de color interna, importación de EDL (*Edit Decision List*[50]), procesamiento de imágenes, y en general tasas de transferencia de datos rápidas ($\sim$ 6 fps a 2K).

Fue líder en la industria de sistemas de trasferencia de películas facilitando el desarrollo del *digital intermediate* (DI), y tiene asociados costos en equipamiento y servicios muy elevados, por encima de las tecnologías de escaneo. Algunos ejemplos desarrollados por la compañía Cintel (actualmente adquirida por Blackmagic) son C-Reality, DSX, Millennium, y por parte de las compañías Phillips y Technicolor son Spirit, Spirit2K, Spirit4K, Shadow, entre otros.

2.5.5. *Scanner*

A diferencia de los aparatos de *telecine*, el *scanner* es un dispositivo de adquisición de datos, pero no para reproducción de video. La transferencia no alcanza a ser en tiempo real, pero la velocidad de escaneo puede llegar a valores elevados de hasta 25 fps a 2K. Admite todo tipo de formatos analógicos, así como resoluciones de salida (SD, HD, 2K, UHD) y manejo de distintos espacios de color en formatos DPX, TIFF con información diversa de *metadata*.

La utilización de *scanner* presenta algunas ventajas con respecto a los otros métodos de digitalización. La captura es extremadamente estable, a diferencia del *telecine*, y es posible escanear películas con roturas en el sistema de perforaciones. También se evitan los problemas asociados al sincronismo de la tasa de fotogramas por segundo.

Algunos ejemplos son Cintel diTTo, dataMill, y Klone, Blackmagic Cintel [51], Filmlight Northlight, Imagica ImagerXE y HSX, ARRI Arriscan [52], *Digital Vision GoldenEye* II, III, entre otros (ver Figura 2.7).

Algunos modelos actuales son portátiles, lo cual ofrece la gran ventaja de que el archivo fílmico no debe alejarse de los depósitos especiales. En general los precios por equipamiento mínimos de estas tecnologías rondan los 40 mil dólares, sin tomar en cuenta los servicios asociados.

(a) (b) (c)

Figura 2.7: (a) Blackmagic Cintel Scanner [51] (b) y (c) Arriscan scanner [52]

Existen diversos mecanismos tecnológicos de arrastre de la película disponibles:

- Intermitente: se arrastra la película hasta el siguiente fotograma por medio de engranajes. Cada fotograma toma aproximadamente un segundo en ser capturado.
- Continuo: se trasporta la película a velocidad constante sin utilizar las perforaciones de la película. Permite escanear las perforaciones de la película, lo cual aporta información al archivo digital sobre el estado de la película. Se utilizan detección óptica de las perforaciones para estabilizar las secuencias.

- *Wet Gate*: la ventanilla de captura se sumerge en un material líquido con índice de refracción similar al del sustrato del material fílmico. Esto rellena gran parte de los *scratches* y roturas en el sustrato y disminuye notoriamente los desperfectos durante la digitalización. Se utiliza un disolvente percloretileno o PERC, de acceso restringido debido a su alta toxicidad (ver Fig. 2.8).

(a) (b)

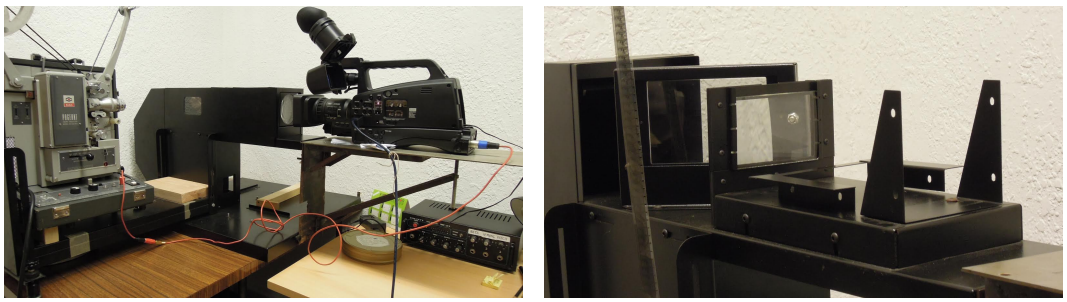
Figura 2.8: Ejemplo de restauración con *ARRISCAN* [53] (a) Con ventanilla estándar seca (b) *Wet Gate*, ventanilla sumergida en *PERC* que elimina *scratches* y desperfectos.

2.5.6. Experiencia con *telecine* del AGU

El equipo del AGU trabajó en la digitalización de su acervo de material fílmico con un sistema de *telecine* [15]. El sistema incluye un proyector 16mm con una lámpara de 60 Watts, un sistema de espejos internos, una cámara digital Full HD, una tarjeta digitalizadora, almacenamiento externo y una interfaz sencilla para el usuario.

Para crear el *master* digital original se utiliza el formato Full HD con compresión Apple ProRes HQ. Estas copias se utilizan en la restauración digital, investigación y archivado. Se hacen copias más livianas para visionado y difusión en Internet, con compresión H264 y reducción de la resolución a SD.

El resultado es aceptable pero incluye algunos desperfectos propios del sistema de telecinado, asociados a la falta de sincronismo entre el dispositivo mecánico y la captura digital. Además los materiales fueron hallados en condiciones de deterioro avanzadas, y presentan una gran cantidad de manchas, suciedades y *scratches*.



(a) (b)

Figura 2.9: *Telecine* del AGU (a) vista general del sistema con proyector y cámara digital de captura (b) abierto con vista a los espejos internos

El archivo audiovisual del AGU cuenta con casi 800 películas. Un archivo de este tamaño implica la realización previa de planes de digitalización donde se establezcan cuáles son los materiales prioritarios a ser digitalizados de acuerdo a los recursos disponibles.

Actualmente un grupo de especialistas trabaja en conjunto con el AGU, en un nuevo proyecto de telecinado a partir del modelo de plataforma libre para *telecine/scanner* Kinograph [54] (ver Sección 2.7.3). Se trata de un proyecto que combina tecnologías de telecine profesional en desuso, con sistemas de captura digital en RAW y 4K. Dichas digitalizaciones presentarán desperfectos particulares y nuevas posibilidades de remasterización, y podrían analizarse y restaurarse mediante las herramientas desarrolladas en este proyecto. [6]

2.6. Proyectos de preservación y restauración digital

Debido a la necesidad de crear herramientas para la restauración digital del vasto archivo de películas deterioradas, distintas organizaciones han creado programas para sus acervos. A continuación se mencionan algunos ejemplos relevantes, con el fin de analizar los objetivos particulares y cometidos detrás de estos proyectos.

The Film Foundation (USA) [55]

Fundada en 1990 en los Estados Unidos por Martin Scorsese y otros directores destacados. Es una organización sin fines de lucro dedicada a la preservación, difusión y restauración de películas clásicas. Ha colaborado en la restauración de más de 700 películas que son accesibles al público general a través de museos, festivales y diferentes planes de formación educativa. Lleva a cabo el plan educativo interdisciplinario *"The Story of Movies"*, que introdujo a más de 10 millones de estudiantes en el lenguaje cinematográfico clásico, su significado cultural, histórico y artístico [56].

[6] Esto se propone en la Sección de Trabajo Futuro 10.2.

The National Film Preservation Foundation (USA) [57]

Se crea con el apoyo de la *Academy of Motion Picture Arts and Sciences (AMPAS)* y de The Film Foundation [55]. Es una organización sin fines de lucro fundada en 1997. Colaboran con distintas instituciones para garantizar la preservación de películas y hacerlas accesibles para estudiantes, investigadores y productores cinematográficos. Estas se distribuyen en la librería del congreso de los Estados Unidos *"Library of Congress"* [58].

PRESTO Space (Inglaterra) [59]

El objetivo del proyecto es proveer soluciones técnicas y sistemas integrados para una completa preservación y restauración digital de todo tipo de colecciones audiovisuales en Europa. La UNESCO estima que existen 200 millones de horas de material audiovisual valioso, y 50 millones en Europa. Todas las grabaciones, video en formatos magnéticos y películas analógicas corren peligro de deteriorarse por completo en los próximos 20 años. PRESTO señala el escenario heterogéneo del patrimonio europeo, y establece que la preservación es un desafío para los archivos nacionales y locales, pero también para las universidades, librerías, museos, empresas y colecciones personales. Se busca centralizar las políticas, el desarrollo tecnológico y el conocimiento en estas temáticas.

AURORA y BRAVA (Unión Europea)

Es un proyecto fundado en 1995 por el programa ACTS de la comisión europea y continuado en 2002 como proyecto BRAVA. AURORA es un acrónimo de *"AUtomated Restoration of ORiginal and video Archives"*. El proyecto, de 3 años de duración, con el objetivo de crear algoritmos con resultados del estado del arte para restaurar un gran volumen de películas y material de televisión [12]. El instituto de investigación de sistemas de información *"Joanneum"* publicó algunos artículos en torno a estos proyectos ([60] [61], [62]).

UCLA Film & Television Archive (USA) [63]

A partir de este proyecto se han restaurado cientos de películas y programas de televisión. Se estima que el 50 por ciento de las películas producidas en Estados Unidos antes de 1950 se han perdido. Casi la totalidad del material fílmico que no ha sido restaurado o retransferido se encuentra en condiciones avanzadas de deterioro. Especialistas de la UCLA sugieren que debido a la rápida obsolescencia e inestabilidad de los sistemas digitales, las películas deben ser retransferidas a película de poliéster [64].

ABS-CBN (Filipinas) [65]

Proyecto de restauración digital de películas realizado por ABS-CBN en conjunto con *Central Digital Lab*. El objetivo es digitalizar, restaurar y remasterizar cerca de 2.400 películas filipinas del acervo del ABS-CBN. Desde el comienzo del

proyecto en 2011, ha culminado exitosamente la remasterización de 100 películas. El formato de digitalización elegido es el DCP (*Digital Cinema Package*), con resolución Full HD a 24 fps progresivo. La distribución se realiza por televisión, Internet, DVDs, BlueRays y salas de cine.

Opciones de formación académica

Existe una gran variedad de escuelas de cine, institutos que ofrecen cursos académicos en áreas de investigación específicas, carreras y maestrías en las temáticas de preservación y restauración de materiales audiovisuales (ver, por ejemplo, [66], [67], [68] y [69]). Muchos de estos cursos se dictan en los Estados Unidos o en el Reino Unido.

2.7. Herramientas de software de restauración digital de películas

2.7.1. Tipos de licencias de software

Antes de presentar las diferentes opciones existentes en el mercado para restaurar películas deterioradas, se presentan las diferentes licencias de software que se utilizan para distribuir y utilizar dichas herramientas. Esto es de suma importancia dado que establece los pactos acordados entre el creador del software, quien posee el derecho de autor y propiedad intelectual, y aquellas personas que lo utilizan, determinando las obligaciones y los derechos de ambos.

En la Figura 2.10 se muestran las distintas categorías de licenciamiento existentes y cómo se vinculan y jerarquizan entre sí.

La siguiente información fue recabada de los sitios *web* de *Free Software Foundation* [71], Sistema Operativo GNU [72], *Open Source Initiative* [73] y *The Debian Free Software Guidelines* (DFSG)[74]. Estas son algunas organizaciones que definen y promueven diferentes tipos de licencia de software, en particular licencias libres y/o *open source*.

Software libre

Se suministra el software con autorización para que cualquiera pueda usarlo, copiarlo y/o distribuirlo, ya sea con o sin modificaciones, gratuitamente o mediante pago. En particular, esto significa que el código fuente debe estar disponible[7]. Cabe mencionar que el software libre es una cuestión de libertad, no de precio. Pero las empresas de software privativo usan habitualmente el término "*free software*"[8]. BSD, GNU y *Free Software Foundation* promueven este tipo de licencias.

[7]Esta es una definición simplificada, la definición completa se puede consultar en el sitio *web* de GNU [75].

[8]En inglés la palabra *free* tiene dos significados: "libre" y "gratuito".

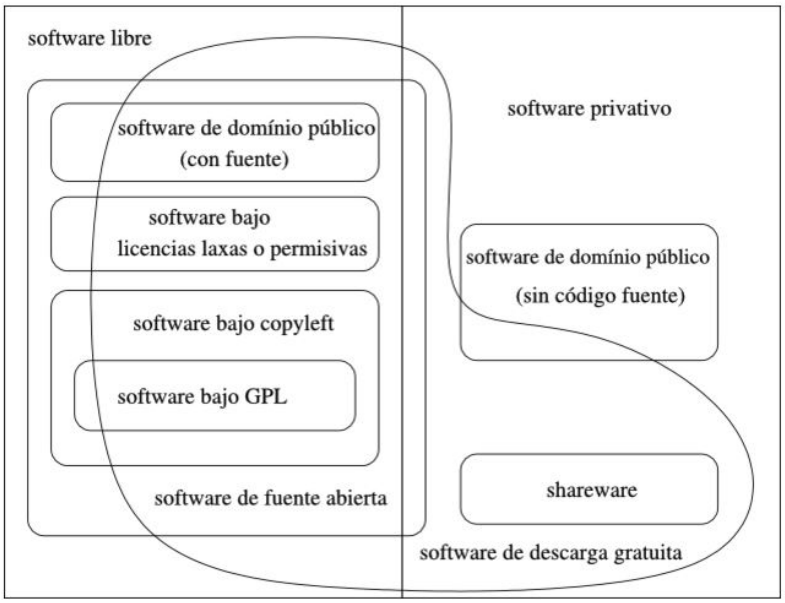


Figura 2.10: En el diagrama se muestran las distintos categorías de licencia de software existentes y sus vinculaciones [70].

Software de código abierto (*open source*)

El proyecto *Open Source Initiative*, que promueve las licencias *open source*, establece que software de código abierto no implica únicamente el acceso al código. Además se deben cumplir una lista de requerimientos específicos sobre el código fuente disponible y el mismo se debe poder redistribuir libremente (aunque sea con fines lucrativos), entre otros requerimientos [76].

Existen discusiones sobre las diferencias entre los conceptos de software *open source* y software libre [77]. Algunas personas utilizan la expresión software de "código abierto" para referirse más o menos a la misma categoría a la que pertenece el software libre. Sin embargo, no son exactamente el mismo tipo de software porque algunas licencias libres no son compatibles con licencias *open source*, y viceversa.

Software de dominio público

Se suministra sin derechos de autor. Si el código fuente es de dominio público, se trata de un caso especial de software libre sin *copyleft*, lo que significa que algunas copias o versiones modificadas pueden no ser libres en absoluto. En algunos casos, un programa ejecutable puede ser de dominio público pero no disponer libremente del código fuente, por lo que no se trata de software libre.

Software con *copyleft*

El software con *copyleft* es software libre cuyos términos de distribución garantizan que todas las copias de todas las versiones tengan aproximadamente los mismos términos de distribución. Esto significa, por ejemplo, que las licencias *copyleft* generalmente no permiten que terceros le agreguen requisitos adicionales al software y exigen que el código fuente esté disponible. Esto tutela el programa y sus versiones modificadas contra algunas de las formas más comunes de convertirlo en software privativo.

Software libre sin *copyleft*

Los programas publicados sin *copyleft* cuentan con permiso de redistribución y modificación, como así también con el permiso de agregarle restricciones. Si un programa es libre pero no tiene *copyleft*, es posible que algunas copias o modificaciones no sean libres en absoluto. Una empresa de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como software privativo.

Software con licencia permisiva o laxa

Estas licencias permiten utilizar el código de cualquier manera, inclusive la distribución de binarios privativos con o sin modificaciones del software libre. Se incluyen en esta categorías las licencias BSD [73].

Software con licencia GPL

La Licencia Pública General de GNU (*General Public License* - GNU GPL) [78] consiste en un conjunto específico de cláusulas de distribución para publicar programas con *copyleft*. Equiparar el software libre con software cubierto por la licencia GPL es por lo tanto un error.

Sin embargo, existen licencias de software libre compatibles con GPL, como por ejemplo lo son las versiones 2 y 3 de la licencia GPL [79].

Software privativo

Incluye todas las categorías de licencia de software que no son libres.

Freeware

El término *freeware* no tiene una definición claramente aceptada, pero se usa generalmente para referirse a paquetes en los cuales se permite la redistribución pero no la modificación (y su código fuente no está disponible). Estos paquetes no son software libre.

Shareware

El término *shareware* se refiere al software del que se permite redistribuir copias, pero quien continúa a utilizar una copia debe pagar para obtener la licencia. El software *shareware* no es software libre.

Software privado

El software privado o software personalizado es aquel que ha sido desarrollado para un usuario (generalmente una organización o una empresa). El usuario lo mantiene y utiliza, y no lo publica, ni como código fuente ni como binarios. En particular, si el usuario tiene todos los derechos sobre el programa privado, el programa es libre. Sin embargo, si el usuario distribuye copias sin otorgar dichas libertades para las mismas, esas copias no son libres.

Software comercial

El software comercial es aquel desarrollado por una empresa como parte de su actividad comercial. La mayoría del software comercial es privativo, pero también existe software libre que es comercial, y software privativo que no es comercial (ver Figura 2.10).

2.7.2. Tipos de software de postproducción digital

Existen diferentes tipos de aplicaciones utilizadas en el proceso de postproducción de una película[80]. La restauración de una película deteriorada podría implicar utilizar varias de estas herramientas, dependiendo las tareas que se planee abordar.

Software de edición permite realizar el montaje (edición) de una secuencia de video. Se trabaja con línea de tiempo para agregar, cortar o mover puntos de entrada y salida de los clips. Es una herramienta adecuada para trabajar con secuencias largas, por ejemplo con la duración típica de una película. Comúnmente se pueden aplicar efectos, transiciones, etc.

Todas las aplicaciones actuales utilizan sistemas de edición no lineal (NLE). Las aplicaciones NLE permiten realizar la edición de forma no destructiva, accediendo a cualquier fotograma de la secuencia en cualquier orden. Esto se define en contraposición a los sistemas anteriores de edición lineal o analógica, que no permitían editar con tal flexibilidad.

Software de postproducción permite trabajar en secuencias de video aplicando diferentes videos, imágenes y efectos. A diferencia de las aplicaciones de edición, no poseen demasiadas herramientas de montaje sino que se especializan en el área de efectos digitales.

Algunas aplicaciones de postproducción se denominan compositores. En estos casos el trabajo se realiza dentro de un único plano o escena, y no resulta adecuado para trabajar en secuencias de larga duración. Los programas de postproducción de video se dividen en dos formas distintas de trabajar:

- Composición basada en capas
- Composición basada en nodos

Dependiendo del tipo de proyecto se debe elegir una u otra opción, dentro de las opciones disponibles en el mercado. A continuación se describen ambas modalidades.

Composición basada en capas

Se trabaja colocando distintas capas de video sobre una línea de tiempo. En cualquier momento es posible agregar o quitar capas, así como reordenar fácilmente distintos fragmentos de video, hasta realizar la exportación final (*render*). Algunas aplicaciones de composición que utilizan esta forma de trabajo son Adobe After Effects[81] y HitFilm[82].

La mayoría de los editores no lineales (NLE) actuales también trabajan de esta forma. Algunas de estas aplicaciones son Adobe Premiere[83], Final Cut Pro[84], Vegas Pro[85] y DaVinci Resolve[86].

Composición basada en nodos

Un nodo es la herramienta básica mediante la cual se ejecuta cualquier acción o modificación sobre una señal de entrada y a partir de ésta se genera una señal de salida. En la mayoría de los casos se tendrá como entrada y salida información de video. Los nodos se utilizan tanto para la lectura y escritura de video, como para aplicar distintos efectos.

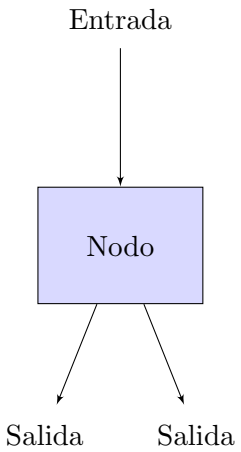


Figura 2.11: Ejemplo de nodo con una entrada y dos salidas.

La composición de video se representa a través de una estructura de árbol, conectando diferentes nodos (videos, imágenes, efectos, etc.) en un mapa jerárquico estructurado desde ciertos nodos fuente hasta la salida final de video (ver Fig. 2.12). Esta interfaz de composición presenta una enorme flexibilidad para el usuario, pudiendo modificar parámetros de una etapa anterior de procesamiento mientras que se observa el resultado final, entre otras características.

Las aplicaciones que trabajan con nodos, en general, carecen de una línea de tiempo donde se puede modificar el video, a

diferencias de las aplicaciones NLE. Dentro de las principales aplicaciones que incorporan composición basada en nodos se incluyen The Foundry Nuke [87], Natron [88], PFTrack[89], Blender [90] y Fusion [91].

Plugins de postproducción componentes de software, que agregan alguna funcionalidad específica a una aplicación de postproducción. Algunos ejemplos son los *plugins* RevisionFX [92] o el eliminador de ruido NeatVideo [93].

2.7.3. Herramientas de postproducción/restauración libres

Una plataforma libre consiste en un sistema de software basado y desarrollado sobre estándares libres, como pueden ser aplicaciones externas o interfaces de programación de aplicaciones (API) ya publicadas y documentadas, que permitan reutilizar el software con distintos objetivos. Se pueden incluir en la plataforma módulos de código libre y/o *open source* que sean compatibles con licencias libres.

En el marco de este proyecto se optó por una plataforma completamente libre (ver Sección 5.3), con una documentación completa y con una comunidad activa. Esto garantiza la posibilidad de participación por parte de otros desarrolladores que deseen y puedan continuar perfeccionando la herramienta y agregando nuevos módulos para restaurar otros desperfectos específicos. A continuación se listan algunas de estas plataformas.

Natron [88]

Natron es un software libre para composición digital de video basado en una estructura de nodos, disponible para los sistemas operativos Windows, MacOS X y Linux. Soporta *plugins* desarrollados de acuerdo a la API OpenFX 1.4 [94]. La mayoría de los *plugins* OFX desarrollados en la industria de efectos visuales, tanto libres como propietarios, son compatibles con Natron.

Fue creado en 2012 por Alexandre Gauthier y su primera versión fue desarrollada en el "*French Institute for Research in Computer Science and Automation*" (INRIA) [95] durante 2013. Se trata del primer software de composición libre con herramientas robustas y eficientes para proyectos audiovisuales profesionales.

CinePaint [97]

CinePaint es un software libre de edición de imágenes, usado principalmente para retoque y restauración de secuencias de imágenes. Trabaja con imágenes de alta calidad como DPX, 16-bit TIFF y OpenEXR [98], además de formatos convencionales como JPEG y PNG. Se basa en el software de edición de imágenes GIMP [99]. Probablemente es una de las herramientas libres más utilizadas en producciones cinematográficas. Un tutorial se encuentra en [100]. Antes se lo conocía

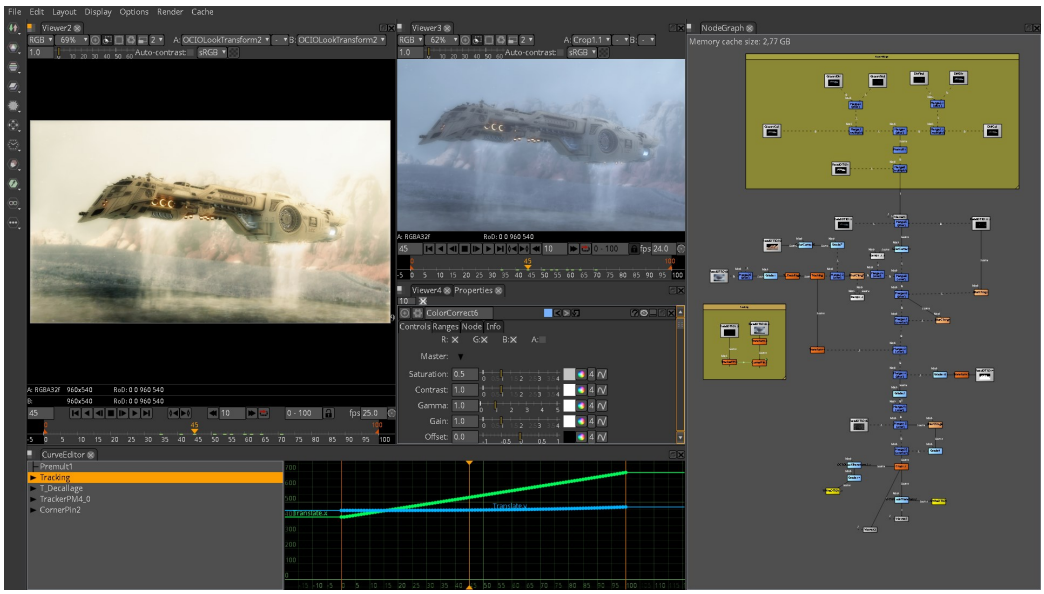


Figura 2.12: Ejemplo de interfaz de Natron [96].

como Film GIMP. Se distribuye con licencia GNU (*General Public License*[101]).

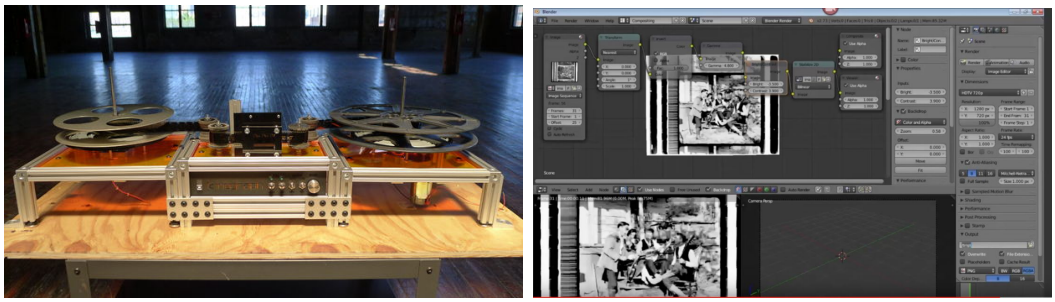
Se encuentra disponible para los sistemas operativos Linux y MacOS. Actualmente no está en desarrollo y las últimas versiones no cuentan con una documentación completa, ni existe una comunidad activa. Por estas razones no fue elegido para desarrollar la plataforma PARP.

Kinograph [54]

Plataforma libre y en código abierto de digitalización con *scanner* y/o *telecine*, compatible con cualquier formato analógico. Actualmente soporta 35mm y 16mm, y se está desarrollando soporte para 8mm. Tiene dependencias *Control P5 library* para *Processing*[102], y una librería OpenCV modificada.

Se recomienda realizar algunas tareas de restauración digital como estabilización y archivado de *masters* en Blender y Fusion 7 [103], [104]. Sin embargo Fusion 7 no es libre (ver Fig. 2.13).

El proyecto incluye información sobre las herramientas de hardware necesarias y tutoriales DIY ("*do it yourself*", "hágalo usted mismo") para reproducir el montaje de una máquina de digitalización. La sección de software incluye código para Arduino [105], y un interfaz de postproducción digital para realizar la detección de las perforaciones de la película. Esto permite la extracción de los cuadros de la película a partir de la captura en bruto.



(a) (b)

Figura 2.13: Kinograph (a) Montaje de la máquina de digitalización v0.1 (b) Estabilización y tratamiento digital en Blender [103]

AVISynth [106]

Plataforma libre con *plugins* de postproducción de video y efectos visuales. Tiene pocos algoritmos específicos de restauración. Los filtros se aplican mediante línea de comando únicamente, no existe una interfaz gráfica de usuario (GUI) para el usuario.

En 2012 un usuario desarrolló con AVISynth una herramienta específica para restaurar películas en 8mm obteniendo buenos resultados (ver [107] y [108]). La herramienta está discontinuada y su documentación es incompleta.

VirtualDub [109]

Herramienta libre de captura y procesamiento de video para Windows. Tiene algunos algoritmos de restauración básicos como *sharpen*, *deinterlace* y *blur*, entre otros. Su desarrollo se detuvo en 2004 pero existe una versión estable para versiones actuales de Windows en 64-bits.

OFX, The Open Effects Association [110]

OFX (también denominado OpenFX) es un estándar libre y abierto para realizar *plugins* de efectos visuales o procesamiento de imágenes [111]. Su licencia actualmente pertenece a *The Open Effects Association* [110]. Permite a los *plugins* escritos según este estándar trabajar en cualquier aplicación o *host* que admita el estándar. Esto evita que el desarrollo y soporte de un *plugin* que realiza cierta tarea sea distinto para cada aplicación o *host* para el cual se desarrolle.

Entre las metas de OFX se encuentran el desarrollo de un estándar:

- para escribir *plugins* de efectos visuales o procesamiento de imágenes
- con amplio consenso en la industria
- independiente del sistema operativo

- suficientemente flexible para soportar un amplio rango de efectos visuales, desde efectos de edición hasta composición.

También se destacan metas menores, como:

- que dicho estándar sea expandible a otras áreas en el futuro, como *plugins* de sonido o *input/output* de imágenes
- que toda expansión del estándar sea realizada de tal modo que todos los *hosts* puedan soportar los *plugins* aunque tengan diferentes versiones de la API
- que los *plugins* escritos siguiendo el estándar puedan ser ajustados a cualquier *host*
- que los *hosts* puedan proveer de funcionalidades extras al *plugin* que lo use.

Tuttle OFX [112]

Consiste en un conjunto de *plugins* libres desarrollados bajo el estándar OpenFX, y que pueden utilizarse en diferentes *hosts* como Nuke y Fusion. Dentro de los *plugins* se incluyen herramientas para realizar procesamiento de imágenes en general, como correciones de color, trasnformaciones de la imagen, conversiones entre diferentes espacios de color, filtros y efectos de imagen como *blur* y reducción de ruido mediante promediados no locales [113].

Incluye la posibilidad de manipular secuencias de video mediante línea de comandos y automatizar diferentes operaciones de procesamiento, como exportar y convertir videos utilizando diferentes formatos.

Posee una documentación completa y una serie de tutoriales explicativos, para utilizar la librería se requieren conocimientos avanzados sobre OpenFX.

OpenCV [114]

OpenCV es una biblioteca libre de *computer vision*. Es multiplataforma, disponible para Windows, Linux y MacOS X. Contiene una gran cantidad de funciones implementadas y optimizadas en el área de procesamiento de imágenes y video que resultan de gran utilidad para implementar algoritmos de restauración. La interfaz utiliza los lenguajes C y C++.

CImg [115]

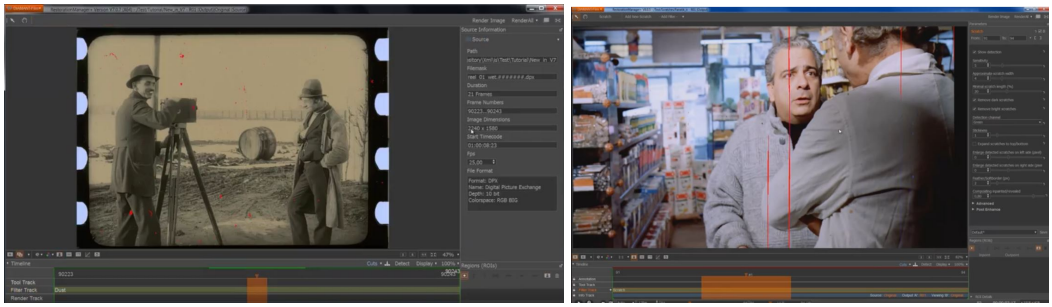
CImg es una librería libre, liviana escrita en C++ para procesamiento de imágenes. Incluye varias funcionalidades, entre otras se destacan cargar y guardar imágenes en varios formatos, filtros, análisis estadísticos, o interacciones entre el usuario y la imagen. Permite trabajar con imágenes de hasta cuatro canales de información. Es autocontenida y portable (funciona en sistemas Unix, Windows, MacOS X).

2.7.4. Herramientas de restauración propietarias

Hs-Art Diamant y DustBuster+ [116], [117]

La aplicación para restauración de películas *Diamant* consiste en una serie de herramientas para restauración digital automática, semi-automática e interactiva. Desde 2001 es utilizada exitosamente por muchos grupos de restauración de películas, laboratorios y casas de postproducción. Incluye herramientas eficientes para corregir la mayoría de los desperfectos típicos en digitalizaciones de material fílmico y un diseño avanzado de interacción con el usuario. Se estima un costo mínimo de 30 mil dólares anuales.

La versión de distribución *DustBuster+* es más económica pero no incluye los filtros automáticos y muchos de los algoritmos de restauración específicos de mayor interés (ver Fig. 2.14).



(a) (b)

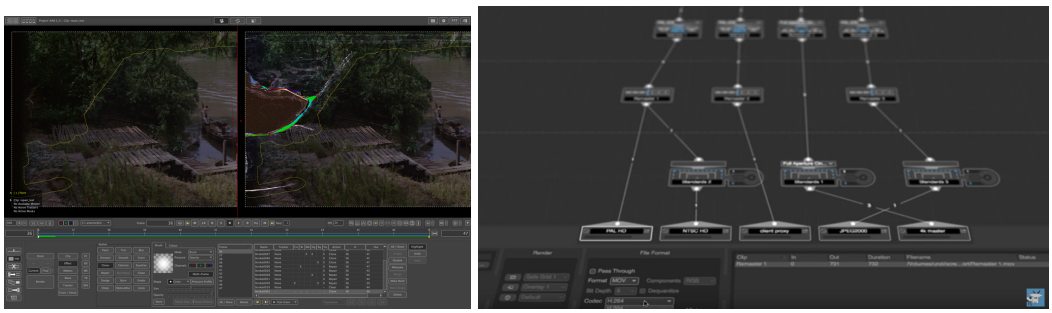
Figura 2.14: Hs-Art Diamant (a) Detección de suciedades (b) Detección de *scratches* [116]

The Pixel Farm - PFClean [118]

Es la aplicación más utilizada para restauración de video digital. Es desarrollada por *Pixelfarm* en Reino Unido. Su costo es de aproximadamente 10 mil dólares. Tiene algunos filtros de restauración digital pero no es la herramienta más completa. Utiliza una interfaz con nodos para el flujo de trabajo del proyecto y las exportaciones (ver Fig. 2.15).

DaVinci Revival [119]

Esta herramienta se encuentra discontinuada. Ahora solo existe la opción *DaVinci Resolve* de *Blackmagic* que es una herramienta poderosa de corrección de color utilizada en Hoollywood. *DaVinci Revival* ofrecía una gran cantidad de filtros profesionales de corrección para la mayoría de los desperfectos en películas deterioradas. Existe una versión gratuita de *DaVinci Resolve* con prestaciones limitadas, pero de todas formas es una aplicación privada y costosa para uso profesional. El precio ronda entre los 10 mil y 30 mil dólares anuales.



(a) (b)

Figura 2.15: Pixelfarm PFClean (a) Interfaz de restauración (b) *Workflow* del proyecto con nodos [118]

Phoenix Film Restoration [120]

Ofrece herramientas profesionales y una interfaz completa para realizar tareas de restauración digital. Tiene herramientas de eliminación de polvo y suciedades, reducción de ruido y corrección de color. El precio no se encuentra disponible en la web, y se debe consultar.

Film Fix

Herramienta discontinuada. Es un *plugin* de restauración distribuido por *Red Giant* para *Adobe After Effects CS3*. Contenía algoritmos de estabilización de imagen, y también de corrección de los problemas de *dirt & sparkles* y roturas. Antes su precio llegaba a los 20 mil dólares pero hoy se encuentra disponible por 125 dólares.

NeatVideo [121]

Es un *plugin* de eliminación de ruido y manchas. Se encuentra disponible para las aplicaciones Adobe Premiere, After Effects, Sony Vegas y Virtualdub. Fácil de usar y arroja muy buenos resultados. Los tiempos de procesamiento en general triplican el tiempo real de reproducción del video. Cuesta solo 50 dólares.

2.8. Conclusiones

Los proyectos de restauración internacionales no acuerdan completamente en cuál es el formato de captura ideal para la preservación en el futuro. Sin embargo, remasterizar y archivar las películas restauradas en formatos con resoluciones Full HD, 2K y UHD con compresiones Apple Pro Res, DPX o TIFF, son estándares adoptados por varias instituciones y archivos a nivel internacional. Esto es porque estas características satisfacen las cantidades de información necesarias para poder restaurar y conservar los archivos.

No existen herramientas de restauración digital libres, que sean profesionales y que cuenten con una comunidad activa de usuarios y desarrolladores. Esto verifica la necesidad de gestar una plataforma libre y abierta, con sus ventajas asociadas, para diferentes usuarios. Por otra parte, la oferta de aplicaciones propietarias es reducida y muy costosa.

Capítulo 3

Estado del Arte: Restauración de Videos

A continuación se exponen y repasan algoritmos y técnicas comprendidas dentro de la literatura de restauración digital de video, con particular énfasis en aquellos que ayudan a restaurar las degradaciones más comunes presentes en material fílmico deteriorado y digitalizado [1].

3.1. Detección de Cortes

La detección de cortes entre escenas o planos en películas permite identificar los cortes presentes en la edición original de la película. Es la forma natural de segmentar una película en partes más chicas y manejables, y por ello es en general el primer paso en algoritmos que realizan tareas de restauración de video. Un plano se puede definir como una serie de imágenes consecutivas y contiguas tomadas por una única cámara representando una acción continua en tiempo y espacio. La detección manual de cortes es una tarea tediosa, y por eso la automatización mediante un algoritmo ahorra una enorme cantidad de tiempo[122].

Existen muchos tipos de cortes, principalmente divididos según dos clasificaciones[123]: por un lado los cortes duros o transiciones abruptas, y por el otro los cortes graduales. Los cortes duros son el corte clásico donde un fotograma corresponde a un plano, y el siguiente corresponde a otro. Los cortes graduales se pueden dividir en varios tipos pero la gran mayoría caen dentro de una de las siguientes tres categorías:

1. Barrido: Esto ocurre cuando una "línea virtual" recorre la pantalla, barriendo la escena anterior y mostrando la nueva. Sucede sobre varios fotogramas.

2. Fundido: Puede ser fundido de salida, donde el plano saliente va desapareciendo hacia una imagen vacía, blanca o negra; o fundido de entrada, donde surge una imagen a partir de una imagen vacía. Dura unos pocos fotogramas.

[1] Las degradaciones se presentaron en la Sección 2.2.

3. Fundido encadenado: En este caso, las últimas imágenes de un plano se superponen con las primeras del siguiente plano. Durante este solapamiento, la intensidad del plano anterior va disminuyendo, mientras aumenta la del plano entrante.

Existen muchos algoritmos para detectar cortes, que funcionan con rendimiento variable según el tipo de corte. Entre ellos se encuentran[124]:

1. **Suma de diferencias absolutas** Este método es el más intuitivo. Se comparan imágenes consecutivas píxel a píxel, y se van sumando las diferencias absolutas entre píxeles correspondientes. Cuanto mayor es la suma, mayor es la probabilidad de que haya ocurrido un cambio de escena. Reacciona sensiblemente a pequeños cambios dentro de una escena, y casos como una explosión o movimiento rápido de la cámara pueden generar falsos positivos. En cambio, casi no reacciona ante cortes graduales.

2. **Diferencia entre histogramas** Este método es similar al anterior; la diferencia es que se computa la distancia entre los histogramas de dos imágenes consecutivas. A diferencia del método anterior, no es sensible a cambios pequeños y rápidos en una imagen, por lo que produce menos falsos positivos. Un problema que puede tener este método es que dos imágenes totalmente distintas podrían tener el mismo histograma o muy similar. Este método no ofrece la misma garantía que el anterior en cuanto a detectar con seguridad los cortes duros.

3. **Relación de cambio de bordes** (*Edge Change Ratio*) Este método intenta comparar el contenido real de las imágenes. Para ello, detecta los bordes en un cuadro, los dilata y luego computa la probabilidad de que el siguiente cuadro contenga los mismos elementos. Es uno de los algoritmos que mejores resultados obtiene, ya que detecta con alto acierto los cortes duros, y se puede adaptar para reconocer también los graduales.

3.2. Corrección de *flicker* de luminancia

En la captura de las cámaras digitales actuales muchas veces se presentan problemas de parpadeo o fluctuación de luminosidad (*flicker*) cuando la velocidad de obturación de la cámara no está en sincronía con la frecuencia de la red eléctrica (en Uruguay 50 Hz) o a la frecuencia de emisión de alguna fuente de luz artificial [125].

El efecto de *flicker* de luminancia presente en películas de celuloide se caracteriza por la fluctuación no natural y cambios en la intensidad de luminancia entre fotogramas consecutivos. Algunos de estos cambios de contraste no son globales y pueden estar localizados solo en algunas regiones de la imagen. En general su origen está asociado a que el proceso de degradación es desigual para distintas partes de la película, como también por ejemplo la operación con la velocidad de obturación durante el proceso de digitalización de la película.

Reducir esta degradación mejora la calidad visual de la película. Además puede ser esencial como primer tratamiento para luego aplicar otros algoritmos de restauración, *e.g.* métodos de estimación de movimiento. Los algoritmos que abordan este tema pueden clasificarse en dos grandes categorías: aquellos que consideran al *flicker* como un fenómeno global, y aquellos que lo consideran como un fenómeno local, actuando de forma no uniforme en las imágenes [126].

Los algoritmos globales, en general, modelan al *flicker* como una transformación afín y global del contraste general de la imagen. Dentro de esta categoría se encuentran los siguientes dos algoritmos:

- Ecualización intermedia entre dos histogramas [127].
- STE (*Scale-time correction for global flicker*). Extensión del anterior a secuencias [128].

Los algoritmos mencionados trabajan sobre toda la imagen. En la práctica, el *flicker* puede afectar algunas regiones específicas de la imagen y aún peor, estas regiones pueden ser variantes en el tiempo. En estos casos puede ser conveniente aplicar algoritmos locales.

En varios de los métodos locales propuestos en el estado del arte (*e.g.* [129], [130], [131] y [132]), el dominio de la imagen se divide en una grilla fija de bloques o *patches*. Estos bloques pueden estar superpuestos. La corrección de contraste es aplicada localmente a cada uno de los bloques, independientemente de los otros.

Se pueden aplicar condiciones de rechazo en *patches* que contengan manchas o suciedades; es un desafío para estos algoritmos no confundir manchas o suciedades con problemas de *flicker*.

3.3. Detección y restauración de manchas y ruido *dirt & sparkles*

Uno de los desperfectos más comunes en películas deterioradas y mal conservadas, son las manchas y suciedades, conocido en la literatura como ruido de tipo *"dirt & sparkles"* ó *"blotches"*. Este ruido se origina por roturas en el celuloide, o suciedades y cuerpos pequeños que son atraídos y se adhieren al material fílmico debido a su electrostática (ver Sección 2.2).

Una de las principales características espacio-temporales del ruido *dirt & sparkles* es su impersistencia temporal y su cambio abrupto de posición en la imagen. Un *blotch*, en general, dura un único fotograma en el cuadro. Entre un fotograma y el siguiente, los *blotches* en general cambian su configuración totalmente. Esto resulta útil para su detección.

La metodología clásica para abordar el problema de los *blotches* en secuencias de imágenes, implica las siguientes tres etapas[133]:

1. Estimar el movimiento de la escena.
2. Establecer un criterio de detección.
3. Restaurar los *blotches* detectados.

Estas tres etapas se resuelven de forma muy variada en la literatura disponible.

La estimación de movimiento busca hallar correspondencias entre los píxeles de un fotograma y los píxeles de otros fotogramas anteriores y/o posteriores en la secuencia. Los métodos existentes se clasifican en directos e indirectos. Los indirectos utilizan detección de características distintivas en la imagen, generalmente con métodos estadísticos [134]. Los métodos directos, en cambio, calculan los parámetros del movimiento a partir de cuantificadores que se obtienen utilizando la información en los píxeles de la imagen. Dentro de estas técnicas se destacan: *block matching*, métodos en el dominio de las frecuencias, y *optical flow*[135]. En general estos algoritmos fallan ante oclusiones de objetos en la imagen o movimientos inestables de cámara. Es un tema activo de investigación y desarrollo en el área.

Los algoritmos de *block matching* son de gran relevancia en el área de compresión de video, y por ello existen implementaciones muy eficientes (ver [136], [137] y [138]).

Una vez que se conoce el movimiento de la escena, existen distintos métodos que permiten detectar los *blotches*. Estos se basan en la umbralización de descriptores de coherencia temporal en el nivel de intensidad de un píxel dado. Estos métodos incluyen los criterios *SDIa, SDIp (Spike Detection Index), ROD, sROD* (Simplified Rank Ordered Difference), etc. Dentro de los criterios mencionados, *sROD*[139] es conocido por su eficiencia [133].

Luego de la detección, se debe restaurar los píxeles donde se detectan *blotches*. En la literatura se propone para dicha tarea utilizar técnicas de *inpainting*, síntesis de textura, modelos auto-regresivos, medianas temporales y métodos Bayesianos, entre otros (ver Sección 3.6) .

3.4. Detección automática de *scratches*

Los rayones o *scratches* son una degradación usual presente en películas de celuloide, ocasionada por la fricción del fílmico contra piezas mecánicas que rotan [2].

Mientras que muchas de las degradaciones comunes en películas deterioradas son temporalmente impulsivas (manchas, *blotches*, *dirt* & *sparkles*, etc), los *scratches* pueden ser persistentes durante varios fotogramas en la misma posición espacial de la secuencia. Por ello los algoritmos deben trabajar de forma específica para automatizar correctamente las detecciones.

Los algoritmos de detección de *scratches* se pueden dividir en dos categorías, los espaciales y los temporales. Ambas técnicas son complementarias.

Algoritmos espaciales de detección de *scratches*

Kokaram [140] fue el primero en introducir un modelo espacial para la detección de *scratches*. Actualmente el modelo sigue siendo ampliamente utilizado, y es considerado el más eficiente [141]. El perfil horizontal del *scratch* se modela como una sinusoide amortiguada y mediante estimación Bayesiana se determina cuando un perfil observado en una imagen corresponde a un *scratch* o no (ver Fig. 3.1).

Otros proponen detectar *scratches* realizando descomposiciones en el dominio de *wavelets* (ver [143], [144], [145]). También se han propuesto redes neuronales para establecer la textura de los *scratches* antes de aplicar filtrado con algoritmos de morfología [146]. Sin embargo, todas estas técnicas presentan varias debilidades en cuanto a su eficiencia [147].

[2]Las características principales de los *scratches* fueron presentados en la Sección 2.2.

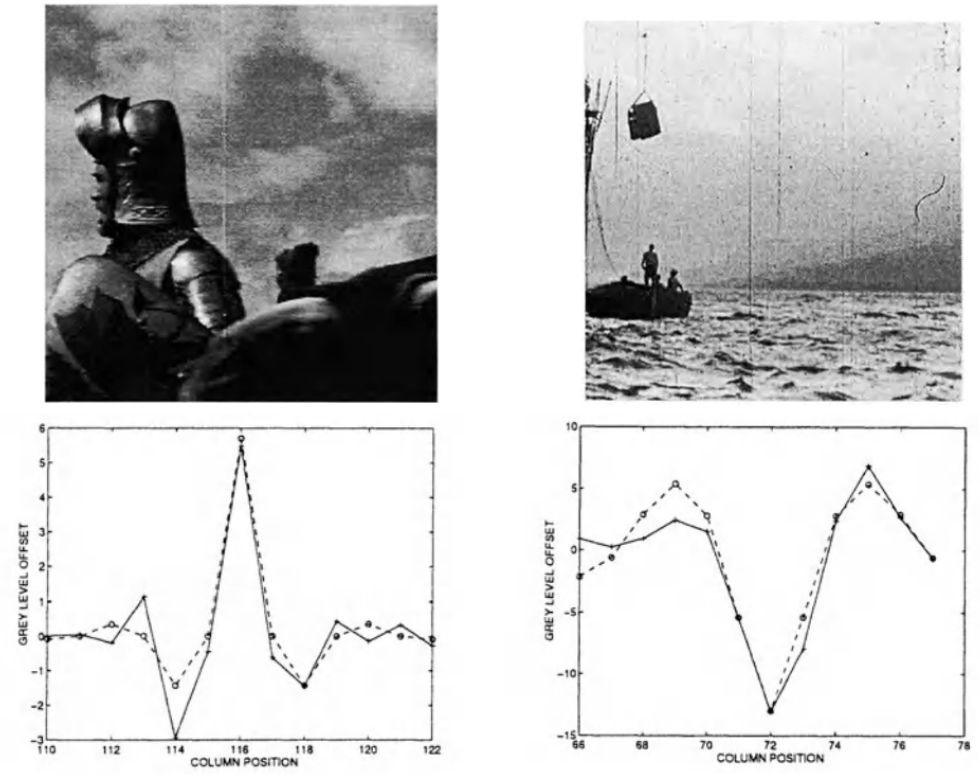


Figura 3.1: Arriba: fotogramas de prueba que presentan *scratches* en las columnas 116 y 72 respectivamente. Abajo: perfiles horizontales del nivel de gris de dichos *scratches* y en línea punteada el modelo de ajuste propuesto por Kokaram en [142].

Además, la detección de características en películas deterioradas puede ser desafiante por la presencia de ruido de grano, suciedades y *dirt & sparkles*. Otro desafío es que muchas estructuras visibles en las secuencias son similares espacio-temporalmente a los *scratches*.

Otros algoritmos en el estado del arte, si bien son eficientes, no son robustos ante el ruido propio de las películas deterioradas y a regiones con textura muy cargada, ya que no se adaptan a estas situaciones [148].

El algoritmo "*Robust Automatic Line Scratch Detection in Films*" propuesto por A. Almansa y colegas [149], en 2014, propone aplicar una metodología *a contrario* que se adapta a la textura de la imagen, para detectar los *scratches*. Esta técnica proporciona resultados robustos al ruido y a la propia textura de la imagen, obteniendo precisión a escala del píxel. Además, es más eficiente y precisa que otras técnicas propuestas anteriormente [150].

Algunos de los artículos mencionados anteriormente consideran al *scratch* como una línea vertical en el fotograma, y por eso son mucho menos precisos (ver

[140] y [151]). El algoritmo que utiliza la metodología *a contrario* mencionado, sin embargo, modela al *scratch* como la unión de diferentes segmentos de recta con distintas pendientes. Esto permite detectar una variedad mucho mayor de *scratches* y ajustarse mejor a los casos reales.

Algoritmos de filtrado temporal

En la literatura se proponen métodos de detección realizando filtrado temporal del video, con el fin de reducir las falsas detecciones que producen, en general, los algoritmos espaciales:

- *Coherencia de movimiento global* de la escena [147]:
  La dinámica de los *scratches* no tiene relación con el movimiento propio de la escena. De esta forma las detecciones que sigan el movimiento global de la escena se toman como falsas detecciones. (Este método no es útil para planos sin movimientos de cámara)
- En base a detección de cortes entre planos de la película [149]:
  Se propone descartar aquellas detecciones cuyos principios y finales distan menos de un cierto umbral temporal τ_c de los cortes entre planos. (*e.g.* τ_c=4 fotogramas)
- Seguimiento (*tracking*) mediante el filtro de Kalman [148]:
  Debido a que el origen del scratch suele estar relacionado al frotamiento de la película contra piezas mecánicas que rotan con cierta periodicidad, se modela la trayectoria del scratch $x(t)$ en cada fotograma, como una suma de tres componentes sinusoidales. Tomando ésta hipótesis se puede hacer seguimiento de los *scratches* mediante un filtro de Kalman y descartar aquellos que no respeten dicha dinámica.

3.5. Eliminación de borrosidad (*deblurring*)

Las películas analógicas comúnmente pierden nitidez con el paso del tiempo. También podrían generarse vibraciones y desenfoque en el video digitalizado con determinadas técnicas de digitalización (ver Sección 2.2). Una técnica sencilla y usual en postproducción para compensar este desenfoque consiste en aplicar un filtrado gaussiano que reduzca el ruido de grano de la película y luego un filtro de *sharpening* para recuperar los detalles desenfocados en la imagen. En general el resultado es una imagen más nítida, pero se realzan los artefactos como *dirt & sparkles* y *scratches*.

Los algoritmos actuales de *deblurring* dentro del estado del arte [152], en general atacan el problema del *motion-blur* ocasionado por el movimiento de la cámara "en mano". Eliminar este tipo de *blur* es uno de los problemas más complejos en restauración de video, dado que el núcleo (*kernel*) de desenfoque no es constante, si no que es variante espacial y temporalmente[153].

Algunos autores observan que en las secuencias con *motion-blur*, no todos los fotogramas están igualmente desenfocados[154]. En base a esta observación, se propone un método que detecta zonas con mayor nitidez *(sharpness)* y con esta información se restaura la información de las regiones desenfocadas en los fotogramas vecinos. Se utilizan técnicas de síntesis de *patches*, que garantizan la coherencia espacial y temporal de la secuencia.

Por otro lado, en el artículo "*Hand-held Video Deblurring via Efficient Fourier Aggregation*"[155], de M. Delbracio y colegas (2015), se propone combinar información en el dominio de Fourier de los fotogramas más nítidos con fotogramas vecinos, obteniendo resultados satisfactorios y más rápidos que los algoritmos presentados anteriormente.

3.6. *Video inpainting*

La técnica de *video inpainting* busca reconstruir información que se ha perdido en alguna región de la imagen. Estas regiones suelen ser desperfectos, *scratches*, manchas, u objetos en general que ocluyen la imagen de interés.

La terminología *"inpainting"* [3] fue adoptada de la denominación usada en restauración de arte, para denotar la modificación de una pintura de forma tal que no sea detectable por un observador que no conoce la obra de arte original. Esto se ilustra en la Fig. 3.2.

Cabe mencionar que el *video inpating* es una extensión en los métodos de *image inpainting* para imágenes fijas, que trabajan fotograma por fotograma, utilizando únicamente información espacial de la imagen. Para trabajar con secuencias de imágenes es necesario utilizar también información temporal del video. De lo contrario se pueden ocasionar incoherencias temporales en el resultado de la restauración. En restauración de películas deterioradas, el *video inpainting* es útil para restaurar *scratches*, manchas y ruido de tipo *dirt & sparkles*.

Todos los algoritmos de *inpainting* existentes requieren, además de la secuencia de imágenes a restaurar, otra secuencia que indique los píxeles donde se debe realizar el rellenado [4].

[3] La terminología fue propuesta en el artículo original "*Image Inpainting*", de M. Bertalmío y colegas [156]

[4] Una revisión exhaustiva del estado del arte del *video inpating* puede accederse en el libro *"Image Processing for Cinema"*, de Marcelo Bertalmío [157].

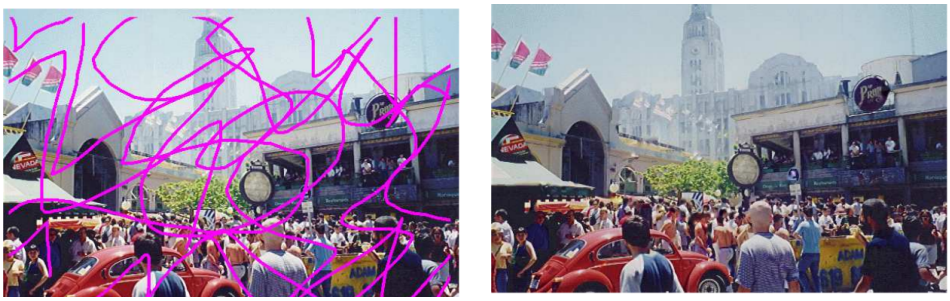


Figura 3.2: Ejemplo de *image inpainting* extraído del artículo "*Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting*"de M. Bertalmío y colegas [158].

Dentro del estado del arte se destacan:

- Métodos geométricos:

  Se utilizan ecuaciones diferenciales parciales (*PDEs*) y métodos variacionales. Las propuestas existentes rellenan las regiones necesarias imponiendo condiciones de borde y de suavidad en las curvas de niveles de la imagen (ver [158], [159] y [156]) y minimizando cierto funcional de energía de regularización (ver [160] y [161]). Básicamente, estos métodos interpolan los valores de intensidad de los píxeles para rellenar las regiones deseadas [162].

  Un ejemplo de esta técnica se muestra en la Figura 3.2, donde se utilizaron las ecuaciones de fluidodinámica *Navier-Stokes* para interpolar los píxeles de la imagen dentro de las regiones rayadas. Alternativamente, otros autores proponen una simplificación de estas técnicas mediante el método de "*fast marching squares*"[163]. Los resultados no son tan precisos como en los otros artículos, pero son menos costosos computacionalmente.

- Síntesis de textura:

  Se buscan *patches* en la imagen para ser copiados en la zona a rellenar [164]. Este método tiene diversas variantes e inspiró el algoritmo de eliminación de ruido *Non-Local Means* de Buades y colegas[113], relevante dentro del estado del arte.

  A su vez existe la posibilidad de rellenar la imagen a través de una representación compacta de la información utilizando elementos de un diccionario de imágenes (ver por ejemplo *e.g.* [165], [166], [167], [168]).

- Técnicas mixtas:

  Se combinan todas las técnicas de *inpainting* mencionadas (por ejemplo ver [169], [170], [171] y [172]). Algunas de las propuestas más recientes en el estado del arte utilizan estas técnicas y superan los resultados obtenidos por las propuestas anteriores. A su vez dichas técnicas permiten realizar el rellenado automático en regiones con un tamaño considerablemente grande en proporción al tamaño de la imagen (por ejemplo ver [173] y [174]).

Capítulo 4

Diseño

En este capítulo se presentan diferentes aspectos tomados en cuenta en el diseño de la plataforma.

4.1. Requerimientos

La plataforma a desarrollar tiene una serie de requerimientos tanto funcionales como no funcionales. Estos surgen del proyecto en si (definidos entre los estudiantes y los tutores), y también del AGU, que actúa tanto como cliente y como usuario de la plataforma. En los primeros meses de proyecto, hubo varias reuniones entre los desarrolladores, los tutores y un grupo de miembros del AGU. Estos últimos se definen como un grupo interdisciplinario, compuesto por historiadores, archivólogos y otros profesionales de la información y la documentación, además de personal con conocimientos técnicos de electrónica, software, y hardware que están más familiarizados con distintas tecnologías de restauración. Entre los requerimientos se incluyen distintas solicitudes que el cliente realiza y otros aspectos interesantes que surgen de las discusiones entre las partes, tanto funcionales como no funcionales que permitirían facilitar el flujo de trabajo de los restauradores.

Requerimientos funcionales

- Contar con películas previamente digitalizadas (la plataforma no se encargará de la digitalización).
- Cargar videos en diferentes formatos estándar.
- Procesar video con algoritmos de restauración.
- Detectar los cortes entre planos dentro de una película.
- Exportar videos en diferentes formatos estándar.

Requerimientos del AGU

- Restaurar con dos pantallas diferenciando la versión original de la versión restaurada, para mantener la referencia original.
- Poder comparar de forma simultánea un mismo video en distintos fotogramas.
- Los algoritmos deben funcionar para resolución HD.
- La posibilidad de incluir *metadata* específica de archivado en los archivos de video finales.

Requerimientos no funcionales

- Para desarrollar los algoritmos, intentar utilizar implementaciones ya existentes.
- Distribución con licencia de software libre.
- Interfaz de usuario amigable y usable.
- Modularidad: Cada algoritmo de restauración debe poder ser visto como un módulo. Por ello, incorporar nuevas herramientas de restauración a la plataforma no debería ser complejo.
- Escalabilidad: Los algoritmos de restauración deben poder trabajar con nuevos formatos de alta resolución (4K, 8K o RAW) y secuencias más largas que una película estándar.
- Mantenibilidad: Código prolijo y comentado que facilite tareas de mantenibilidad.
- Rendimiento y eficiencia. Se deben alcanzar tiempos de procesamiento que permitan realizar de forma fluida el proceso de restauración de películas en alta definición.
- Portabilidad: La plataforma debe poder ser compatible con diferentes sistemas operativos.

Se plantea basarse en alguna plataforma ya existente que resuelva varias de estas problemáticas. Por ejemplo, no trabajar sobre problemas ya resueltos como la carga y exportación de un video en distintos formatos, sino centrarse en los algoritmos de restauración y su rendimiento.

4.1.1. Interfaz de usuario

Uno de los requerimientos importantes del proyecto implica que la plataforma cuente con una interfaz amigable para el usuario, con herramientas de manipulación de secuencias, que permitan llevar a cabo el proceso de restauración de forma fluida.

En la Figura 4.1 se muestra un croquis con el diseño inicial de la interfaz, a partir del trabajo en conjunto con el AGU. Se incluye la posibilidad de monitorear video en dos visores distintos, una línea de tiempo navegable, y diferentes menúes con opciones para aplicar efectos, modificar sus parámetros y gestionar el proyecto.

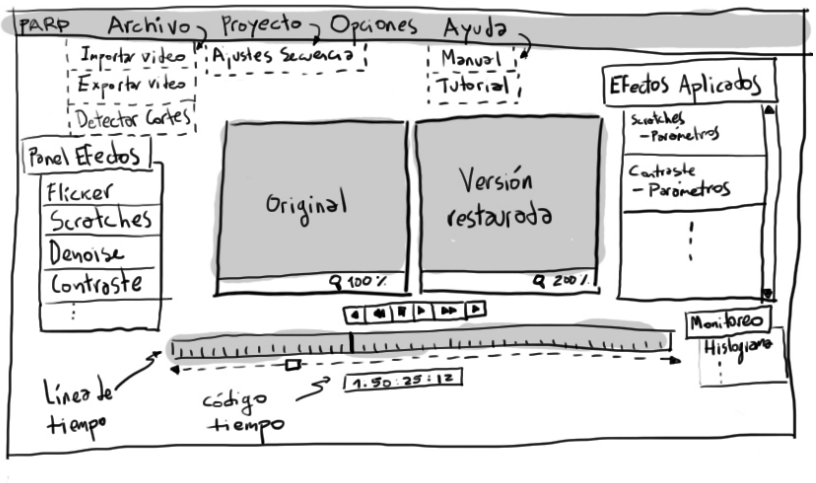


Figura 4.1: Croquis de diseño de la interfaz de usuario

4.2. Representación de alto nivel

En la Figura 4.2 se muestra una representación modular de alto nivel de la plataforma. Se parte de una película digitalizada suministrada y gestionada por algún archivo. Se espera que el formato de video sea alguno estándar. En primer lugar dicho video debe poder cargarse y visualizarse en la plataforma, respetando la información original. Luego se debe segmentar el video en diferentes fragmentos, en el caso más especifico estos fragmentos son los planos de una escena, pero podría segmentarse de forma arbitraria por el usuario, *e.g.* agrupando varios planos. A su vez el usuario debe poder aplicar filtros de restauración a los diferentes fragmentos, y poder ajustar los diferentes parámetros de los filtros. Finalmente se debe exportar el video en un formato estándar a elección.

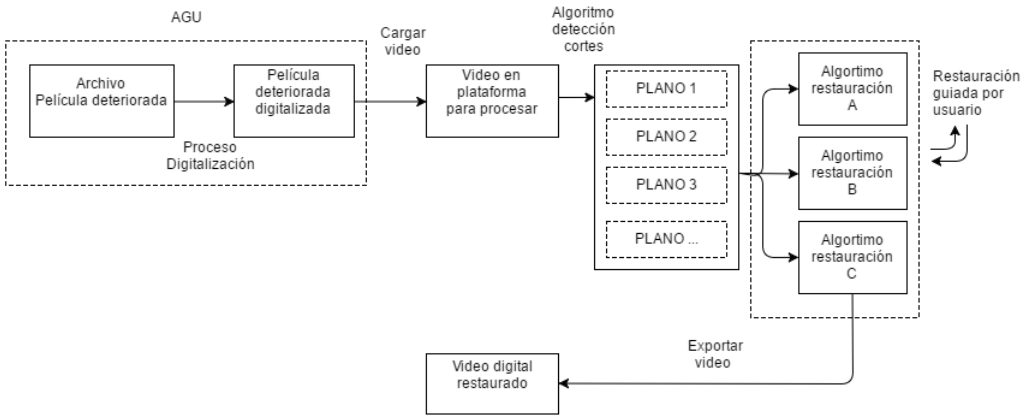


Figura 4.2: Representación de alto nivel de la aplicación

4.3. Flujo de trabajo de los algoritmos de restauración

En la Figura 4.3 se propone un flujo de trabajo para los algoritmos de restauración, ordenado de forma tal que las diferentes etapas se realicen de forma secuencial y estratégica.

Es importante mencionar que algunos algoritmos mejoran significativamente su desempeño con un correcto tratamiento previo de la secuencia [175]. Por ejemplo, un algoritmo de corrección de *flicker* generalmente "empareja" histogramas entre pares de fotogramas consecutivos. Para ello es imprescindible previamente detectar los cortes entre planos de la secuencia, para aplicar dicho algoritmo solo en ese intervalo de fotogramas. Además el efecto de *flicker* puede empeorar e impedir una correcta estimación de movimiento, necesaria para diversas tareas de restauración. Por esto, en primer lugar se aplica un algoritmo de detección de cortes que divide la película en planos y luego se realiza la corrección de *flicker* de luminancia.

Una vez que se elimina el *flicker*, se aplican técnicas de estimación de movimiento que sirven como entrada para los algoritmos siguientes en el flujo de trabajo (eliminación de ruido, manchas, *scratches*, borrosidad, etc). Por último se pueden utilizar diferentes técnicas de estabilización y corrección de color.

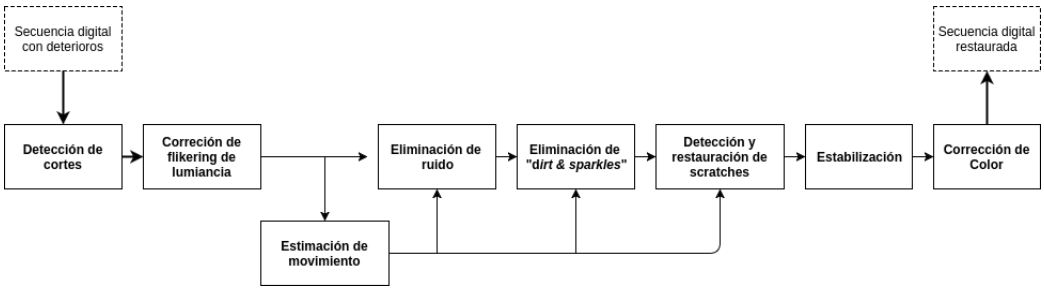


Figura 4.3: Vista esquemática de un sistema modular de restauración digital de material fílmico

Capítulo 5

Implementación

En este capítulo se presentan los aspectos relacionados al desarrollo y la implementación de la plataforma de restauración de películas.

5.1. Decisiones

En esta sección se presentan las razones de las elecciones de la plataforma y librerías de procesamiento de imágenes elegidas sobre la cual trabajar, y también de los problemas de restauración que se decidieron abordar.

5.1.1. Elección de plataforma

Restaurar digitalmente películas deterioradas implica la utilización de algoritmos de un elevado costo computacional. Por otro lado, las películas en fílmico que fueron previamente digitalizadas ya están editadas en su versión final, y no requieren grandes ajustes de edición ni modificación de los tiempos de la película.

Al momento de relevar plataformas para este proyecto, se encontró que la oferta de aplicaciones de composición por capas o aplicaciones de edición no lineal libres era prácticamente nula. Por ello se optó por trabajar en Natron, una aplicación de composición libre basada en nodos.

Esta aplicación permite llevar a cabo de forma eficiente la tarea de restauración digital de video, mediante la incorporación de *plugins* de postproducción bajo el estándar de OpenFX.

A continuación se presentan ventajas que ofrece Natron. La versión actual 2.2.7 ofrece las siguientes prestaciones:

- Soporte para decenas de formatos de archivo y codecs estándar en postproducción de video: EXR, DPX, TIFF, PSD, SVG, RAW, JPG, PNG, AppleProRes, H264. Esto es muy importante para poder importar cualquier tipo

de video presente en los archivos de material fílmico digitalizado, y además permite exportar distintas versiones para archivar las películas restauradas.

- Flujo de trabajo de color lineal, 32 bits en punto flotante.
- Soporte para varios *plugins* OpenFX libres y de código abierto:

  OpenFX-IO: Lectura de todo tipo de archivos que no sean imágenes estándar 8-bits (incluido en Natron).

  OpenFX-Misc: Un conjunto de varios nodos para procesamiento de imagen como Transformar, *Chroma Keyer*, Invertir, etc. (incluidos en Natron).

  OpenFX-Arena: Un conjunto extra de nodos (incluidos en Natron).

  OpenFX-OpenCV: Un conjunto de *plugins* basados en OpenCV (actualmente discontinuado).

  OpenFX-Yadif *deinterlacer*: Un *plugin* para deinterlacear video eficiente.

- Soporte para *plugins* OpenFX comerciales:

  RevisionFX [92]

  NeatVideo *denoiser* [93]

  The Foundry - *Furnace* [176]

  The Foundry - *Keylight* [177]

  GenArts Sapphire [178]

  Red Giant Software - *Universe* [179]

La interfaz de usuario de Natron es intuitiva y tiene fácil acceso a varias herramientas de manejo de video[1].

Natron trabaja con *plugins* escritos bajo el estándar OpenFX. Como se analizó anteriormente (ver Sección 2.7.3), trabajar con OpenFX ofrece grandes ventajas, entre las que se destacan la independencia del sistema operativo, la independencia del *host* (*plugins* escritos en OpenFX pueden ser incluidos tanto en Natron como en otras aplicaciones que utilicen el estándar), es suficientemente flexible como para soportar un amplio rango de efectos visuales, desde efectos de edición hasta composición, y es expandible a otras áreas en el futuro, como *plugins* de sonido.

Además, se decidió incorporar la librería OpenCV porque la misma incluye varias funciones específicas y eficientes que fueron requeridas al momento de implementar los algoritmos de restauración. Respecto a la versión, se optó por la estándar 2.4 en lugar de la actual 3.1. Esta última, al ser una versión más reciente, es más propensa a sufrir modificaciones y resulta menos estable.

[1] Esto se describe en el Manual de Usuario de PARP, disponible en el Anexo B.

5.1.2. Herramientas de restauración digital disponibles en Natron

Si bien Natron no es una aplicación de restauración estrictamente, cuenta con una serie de *plugins* útiles para restaurar determinados deterioros presentes en películas deterioradas. Se resumen a continuación dichas herramientas:

Sharpen y reducción de ruido [180] permite eliminar ruido y/o agregar nitidez (*sharpness*) a las imágenes, utilizando el algoritmo "*Adaptive image denoising by rigorous Bayesshrink thresholding*" [181] basado en *wavelets*, que provee resultados eficientes dentro del estado del arte [182].

Estabilización de imagen [183] permite rastrear ("trackear") uno o más puntos 2D y utilizarlos para estabilizar la imagen, en caso de que la imagen de entrada presente vibraciones de posición y rotación.

Image Inpainting [184] es un *plugin* en versión *beta* para realizar *image inpainting*. La implementación utiliza la librería abierta CImg. Se basa en el método propuesto por M. Daisy "*A Smarter Examplar-based Inpainting Algorithm using Local and Global Heuristics for more Geometric Coherence*"(2014).

Al combinar técnicas de *image inpainting* y síntesis de textura, este algoritmo es capaz de restaurar regiones grandes de la imagen. (ver Sección 3.6) Esto puede ser útil para remover manchas y roturas grandes en fotogramas particulares de una secuencia (ver Fig. 5.1). De todas formas, no se trata de un algoritmo automático para video, y puede presentar inconsistencias temporales en los resultados.

Figura 5.1: Proyecto de Natron donde se trabaja con la película "A bordo del Ancap" (1968, ICUR) A la izquierda en el visor de Natron, el fotograma original. A la derecha, *zoom* de la imagen con una mancha restaurada mediante el *plugin* de *image inpainting* de Natron. Este ejemplo se desarrolla en el Manual de Usuario.

5.1.3. Selección de los problemas a abordar

En este proyecto se desarrollaron los bloques de detección de cortes, corrección de *flicker* de luminancia, y un algoritmo de detección y restauración de *scratches* que no implica realizar estimación de movimiento. Los dos primeros se eligieron por estar al principio de la cadena de trabajo. Luego, los otros dos problemas más comunes son *dirt & sparkles* y *scratches*, y como las digitalizaciones del AGU presentaban un gran número de *scratches* se eligió trabajar sobre este último [2] (ver Fig. 5.2).

5.2. Integración entre plataformas

En esta sección se presenta la integración entre las distintas tecnologías que se utilizaron para el desarrollo de los *plugins*: Natron, Python, OFX (C++) y OpenCV. Lograr estas integraciones fue una tarea difícil en este proyecto, y entenderlas es esencial ya que de otra manera no se puede desarrollar un *plugin* en Natron.

[2] En los Capítulos 6, 8 y 7 se expone de forma detallada la implementación de los algoritmos. En la Sección de Trabajo Futuro 10.2, se presentan algunas posibles direcciones para incorporar nuevos algoritmos de restauración, de acuerdo con el flujo de trabajo propuesto.

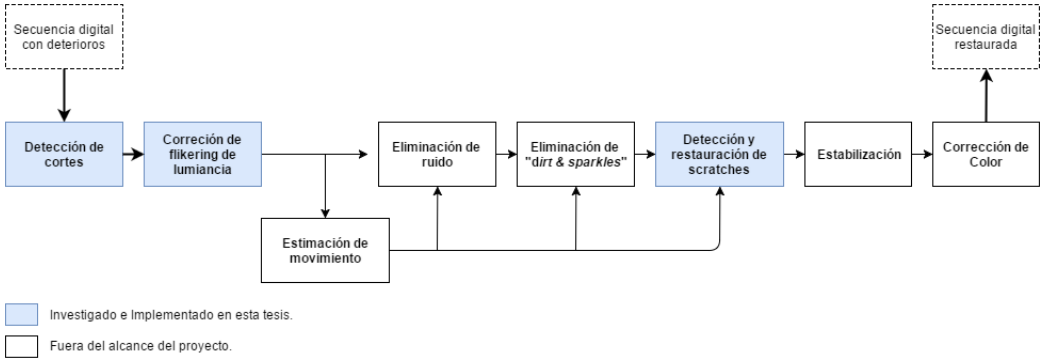


Figura 5.2: Vista esquemática de un sistema modular de restauración digital de material fílmico, donde se señalan los problemas abordados en este proyecto.

5.2.1. Diseño general de la aplicación

En la Figura 5.3 se muestra un diagrama de bloques con el diseño global de la solución propuesta para restaurar películas digitalizadas mediante la implementación de algoritmos en Natron.

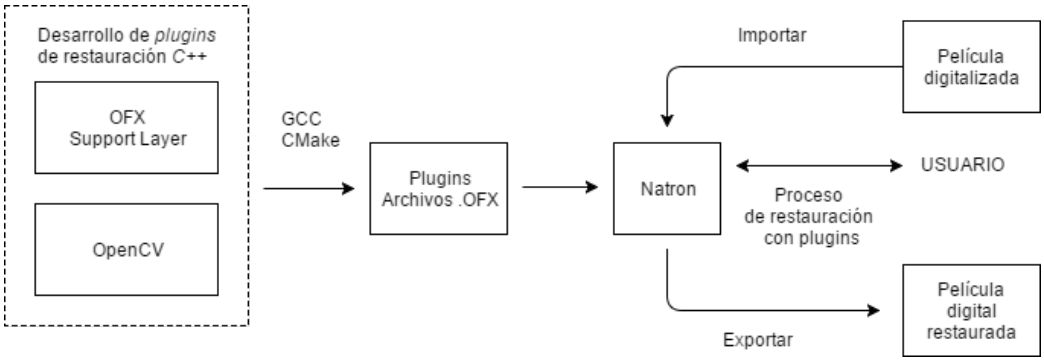


Figura 5.3: Diseño global de la implementación

Los diferentes bloques presentados y cómo se comunican e interactúan, se analizan a continuación.

5.2.2. Integración entre Natron y OFX

OFX es un protocolo que permite la comunicación entre diferentes *plugins* y *hosts*. Esta comunicación se logra a través de una API en C, que provee clases, funciones y lineamientos a seguir para manejar el procesamiento de video.

Para desarrollar un *plugin* que realiza tareas de procesamiento de video especificas, el estándar exige ciertas estructuras de datos y funciones que deben ser implementadas. A modo de ejemplo, OfxPlugin es una *struct* de C que se debe completar con sus propiedades para que el *plugin* se describa a sí mismo ante el *host*.

Se puede programar un *plugin* en C utilizando solamente el protocolo de OFX, pero esta tarea no es sencilla. Por ello, para facilitar la utilización de esta API y no trabajar con OFX directamente, existe un *wrapper* en C++ cuyo nombre es *Support Layer* [185]. En este proyecto se trabajó utilizando este *wrapper* y las funciones que brinda. Un primer ejemplo para introducirse en la utilización del mismo y crear un primer *plugin* puede encontrarse en la guía de programación de Natron.[186].

Todo *plugin* es instanciado por la aplicación de distinta manera dependiendo de su contexto/uso. Es decir, según el uso que el usuario le dará al *plugin* se define cómo será la interacción con el *host*. El conjunto completo de los distintos contextos se describen en la especificación de OpenFX [187]. En particular, para los *plugins* desarrollados en este proyecto se utilizó el contexto *general effect*.

En el estándar también se definen las funciones que el *host* puede llamar sobre los *plugins* para realizar acciones específicas. A su vez, el *plugin* puede llamar a ciertas funciones del *host* para obtener información que este requiera, por ejemplo para obtener alguna característica de la imagen de entrada. En la documentación de OFX se describen todas estas funciones [188].

Al compilar un *plugin* OFX, se genera un archivo de extensión .ofx. Para poder utilizar dicho *plugin* en un *host*, se debe propiciar al *host* la ruta donde se encuentra el archivo, para que este lo cargue cada vez que se ejecuta.

5.2.3. Integración entre Natron y Python

Natron permite utilizar *scripts* escritos en Python para lograr distintas funcionalidades. En la documentación oficial[189], se proveen distintas funciones que permiten entre otras cosas crear nodos, controlar los parámetros de los mismos, o determinar los *callbacks* que se ejecutan luego de procesar un nodo. Además, utilizando *start-up scripts*, que se ejecutan cuando se ejecuta Natron, se puede modificar la interfaz del programa, por ejemplo agregando opciones y menúes para el usuario.

5.2.4. Integración entre OpenCV y OFX

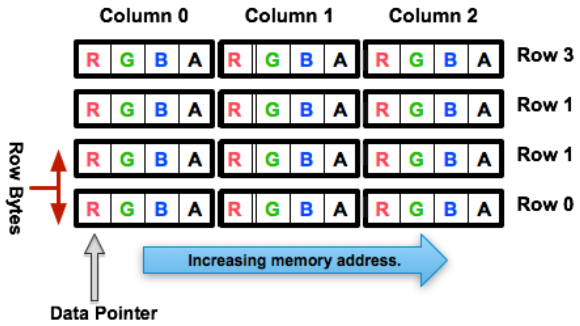
Para integrar OFX con la librería OpenCV, es necesario instalar la versión 2.4.13 de OpenCV[190]. Luego, en el código de OFX es necesario incluir los archivos *header* de OpenCV, y agregar la librería a los archivos CMake[191].

Naturalmente OpenCV y OFX utilizan diferentes estructuras de datos para almacenar y manejar las imágenes. Un aspecto muy importante de la integración, es que la conversión de un tipo de datos al otro sea eficiente, de lo contrario se enlentecerían los tiempos de ejecución de los algoritmos.

El estándar OFX utiliza las clase OFX:Clip y OFX:Image para almacenar las secuencias de imágenes y toda la información asociada al formato de video[3]. Las imágenes se leen como se muestra en la Figura 5.4. Para recorrer toda la imagen se utiliza la función OFX:GetPixelAdress, que provee un puntero hacia el primer píxel de una fila. Luego se incrementa dicho puntero para recorrer los diferentes canales de color de los píxeles, a lo largo de toda la fila[186].

Lectura y Escritura de una imagen OpenCV desde OFX

OFX utiliza imágenes con cuatro canales RGBA, con profundidad de color de 32 bits (8 bits por canal), representación en punto flotante, con valores entre 0 y 1, y espacio de color lineal[4].



(a)

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

(b)

Figura 5.4: Tipos de datos utilizados para almacenar imágenes utilizados por (a) clase OFX:Image en RGBA de OFX[194] (b) clase cv:Mat en BGR de OpenCV[195].

OpenCV provee dos estructuras de datos diferentes para gestionar las imágenes. Estas son las clases cv:IplImage[196] y cv:Mat[195]. Cv:IplImage utiliza el estilo de C y cv:Mat es una interfaz más cómoda para C++ y por esto fue elegida para implementar los algoritmos. Una ventaja importante de cv:Mat es que no requiere prealocar y liberar la memoria, estas tareas se realizan de forma automática[5].

[3]Todo lo relacionado a dichas clases se encuentran documentado en la guía de programación de OFX [192].

[4]Esto y más detalles se especifican en la documentación de Natron [193].

[5]Una comparación exhaustiva de estos tipos de datos se provee en un tutorial de OpenCV[197].

La estructura de datos para almacenar imágenes cv:Mat en memoria se puede ver en la Figura 5.4. Para la implementación de los algoritmos de restauración se utilizaron imágenes de tres canales BGR y tipo de datos de tipo *uchar*, con profundidad de color de 8 bits por canal, con valores entre 0 y 255. En el caso específico de restauración de películas, no se utiliza el cuarto canal *alpha*, y al no considerarlo se ahorra espacio en memoria.

OpenCV provee dos métodos para acceder a la información de los píxeles de una imagen cv:Mat. El método mas sencillo es utilizando la función *at* [198]. El otro método consiste en acceder a la imagen utilizando punteros a través de la función *.ptr* [199], que devuelve un puntero a la posición de memoria donde se almacena la información de los píxeles.

Para implementar los algoritmos de restauración se crearon dos funciones que realizan la lectura y escritura de un formato al otro. Primero una función de lectura, lee la imagen OFX:Image y la convierte al tipo cv:Mat. Se realiza el procesamiento de la imagen con los diferentes algoritmos en OpenCV y al final la imagen cv:Mat resultante, se copia dentro de una imagen OFX:Image de destino, que se despliega en la interfaz de Natron. Para convertir los tipos de datos de *float* a *uchar* se utilizó la función *convertTo* de OpenCV[199].

En dichas funciones, originalmente se había utilizado el método *.at* que provee OpenCV para acceder a los píxeles. Esto enlentecía la ejecución y fue corregido utilizando punteros en las iteraciones. Se probaron diferentes estrategias para iterar con los punteros y se eligió la más eficiente, que se presentan en los Algoritmos 1 para la lectura y 2 para el caso de la función de escritura. Esta implementación coincide con la sugerencia presentada en documentación de OpenCV[200], para iterar de forma eficiente con imágenes cv:Mat. Para evaluar el manejo de la memoria y los tiempos de ejecución implicados en estas funciones se utilizaron las herramientas Valgrind [201], Callgrind [202] y KCachegrind [203].

En el primer bucle, el Algoritmo 1 recorre todas las filas de la imagen OFX. En cada iteración se obtiene, para dicha fila, un puntero de acceso a memoria correspondiente al píxel ubicado en la primer columna de la imagen (ver Fig. 5.4a. El segundo bucle recorre los píxeles de todas las columnas de la imagen, incrementando el puntero. En dichas iteraciones se copia la información de píxel de la estructura OFX:Image a la imagen cv:Mat.

Algoritmo 1: Pseudocódigo de conversión entre OFX:Image y cv:Mat

Entrada: Imagen origen OFX:Image RGBA, *float*
Obtener nCols número de columnas de la imagen OFX:Image;
Obtener nFilas número de filas de la imagen OFX:Image;
Crear cv:Mat de tipo *float*;
para cada *fila i de OFX:Image* **hacer**
  Obtener puntero $*PixelOFX$ a la fila i con método *.getPixelAdress* de OFX;
  Obtener puntero $*PixelCV$ a la fila i $*PixOFX$ con método *.ptr* de OFX;
  para cada *cada columna j de la imagen OFX:Image* **hacer**
    $PixelCV_{ij}$ [0] = $PixelOFX_{ij}$[0] ;
    $PixelCV_{ij}$ [1] = $PixelOFX_{ij}$[1] ;
    $PixelCV_{ij}$ [2] = $PixelOFX_{ij}$[2] ;
    $PixelCV$ +=3 (píxeles BGR);
    $PixelOFX$+=4 (píxeles RGBA) ;
  fin
fin
Convertir cv:Mat de tipo *float* a *uchar* con el método *convertTo* de OpenCV;
Salida: Imagen cv:Mat, BGR y *uchar*

A partir de este punto es posible procesar las imágenes cv:Mat utilizando las funciones que provee OpenCV. Luego de realizar dicho procesamiento, se debe retransferir la imagen cv:Mat a la imagen OFX:Image de salida, que se monitorea en la interfaz de Natron. En el Algoritmo 2 se muestra el pseudocódigo correspondiente a dicha transferencia. Este procedimiento resulta bastante análogo al Algoritmo 1, aunque presenta algunas diferencias. Por ejemplo se debe rellenar el canal A (*alpha*) de la imagen OFX:Image con su valor máximo, para que la imagen no tenga transparencia.

Algoritmo 2: Pseudocódigo de conversión entre cv:Mat y OFX:Image

Entrada: Imagen origen cv:Mat BGR, *uchar*, Imagen OFX:Image vacía
Convertir cv:Mat de tipo *uchar* a *float* con el método *convertTo* de OpenCV;
Obtener nCols número de columnas de la imagen cv:Mat;
Obtener nFilas número de filas de la imagen cv:Mat;
para cada *fila i de cv:Mat* **hacer**
  Obtener puntero $*PixelOFX$ a la fila i con método *.getPixelAdress* de OFX;
  Obtener puntero $*PixelCV$ a la fila i $*PixOFX$ con método *.ptr* de OFX;
  para cada *cada columna de la imagen cv:Image* **hacer**
    $PixelOFX_{ij}$ [0] = $PixelCV_{ij}$[0] ;
    $PixelOFX_{ij}$ [1] = $PixelCV_{ij}$[1] ;
    $PixelOFX_{ij}$ [2] = $PixelCV_{ij}$[3] ;
    $PixelOFX_{ij}$ [3] = (float)1 ;
    $PixelCV$ +=3 (píxeles BGR);
    $PixelOFX$+=4 (píxeles RGBA) ;
  fin
fin
Salida: Imagen OFX:Image, RGBA y *float*

Con las implementaciones presentadas es posible programar diversos *plugins* utilizando funcionalidades provistas por OpenCV. En el repositorio de PARP se provee un ejemplo sencillo con instrucciones para que un usuario con conocimientos de C++ pueda programar nuevos módulos de procesamiento de imagen e incorporarlos como *plugins* a Natron u otros *hosts*.

Cabe mencionar que existe un paquete de *plugins* OFX implementados en OpenCV, pero no están actualizados y no funcionan con las nuevas versiones de Natron [204].

5.3. Licenciamiento

Anteriormente se presentaron las diferentes categorías de licencia de software disponibles (ver Sección 2.7.1). En esta sección se presenta la licencia bajo la que se distribuye PARP.

Uno de los objetivos del proyecto consiste en generar herramientas que sean libres y abiertas. La implementación de la plataforma de restauración se llevó a cabo contribuyendo con el software libre Natron, y en este sentido se debe duplicar

la licencia del software original para redistribuirlo [205].

Anteriormente a la versión 2.0, Natron se distribuía bajo la licencia Mozilla Public License versión 2.0 [206]. A partir de la versión 2.0, se distribuye bajo la licencia GNU *General Public License* versión 2 o mayor [101]. Todos los *plugins* distribuidos con el binario de Natron 2.0 o posterior tienen compatibilidad con GPLv2.

El contenido producido por Natron o cualquier software distribuido bajo GPL, no está cubierto por GPL, sino que los derechos pertenecen al usuario del programa.

OpenCV se distribuye bajo la licencia libre BSD en su versión revisada "*3-clause BSD License*" [207]. Dicha licencia es compatible con GPL v2 [208].

5.4. Repositorio de PARP

Se puede acceder y descargar el código fuente del proyecto mediante el siguiente repositorio en la plataforma GitHub: https://github.com/SebastianBugna/PARP.

Se provee una carpeta para cada algoritmo. Cada una de estas carpetas incluye el código necesario para compilar los *plugins* y los archivos makefile correspondientes.

Los *plugins* se compilan desde un terminal ejecutando comando *make* del compilador *GCC* [209]. Pueden compilarse los tres *plugins* a la vez desde la carpeta raíz del repositorio, o pueden compilarse por separado accediendo a la carpeta específica de cada *plugin*. Se pueden agregar diferentes opciones de configuración, dependiendo si se desea compilar la versión optimizada para utilizar los *plugins*, o trabajar en el código y poder hacer *debugging*, para lo cual se desactivan las optimizaciones del compilador GCC.[6]

Las carpetas openfx y SupporExt incluyen el código necesario para utilizar el estándar OFX y ejecutar los *plugins* en Natron. Tanto openfx [210] como SupportExt [211] son repositorios externos. Es posible actualizar el contenido de estas carpetas mediante la herramienta de submódulos de Git [212](ver Manual de Usuario). Sin embargo, se decidió no incluir esta opción en el repositorio PARP, para independizarse de las actualizaciones de los repositorios externos, y conservar una versión estable de la plataforma.

[6]Las instrucciones de compilación se detallan en el Manual de Usuario de PARP, disponible en el Anexo B.

5.5. Testing y validación

Se realizaron dos sesiones con miembros del AGU para validar distintas funcionalidades. Ambas fueron con un perfil de usuario distinto. Se denominará a dichos usuarios como Usuario A y Usuario B.

La primera sesión se realizó con el Usuario A, que es archivólogo e historiador. Poseía experiencia previa básica con aplicaciones de postproducción de video, y no conocía Natron, ni otros softwares similares que también utilicen nodos (como por ejemplo Nuke). La segunda sesión se realizó con el Usuario B, Ingeniero Electricista, quien poseía experiencia profesional en aplicaciones de postproducción, edición y corrección de color, y también en el área de restauración de imagen. En ambas sesiones se descubrieron *bugs* y se pudieron validar requerimientos del usuario, y ambas resultaron en cambios en la aplicación.

En la primera sesión, se presentó al Usuario A el Manual de Usuario y se probó qué tan intuitivo y usable era dicho manual, además de la interfaz de Natron. El objetivo era estudiar si por sí solo, un usuario podía completar distintas tareas de restauración. La sesión duró dos horas, y a continuación se presentan los distintos aspectos interesantes que surgieron:

- Respecto a la interfaz, la encontró intuitiva, destacando la utilidad de la visualización en dos columnas.
- Respecto al manual, el usuario no comprende del todo qué son los nodos, ni cómo se crean ni eliminan. Tampoco le queda claro cómo se exporta un video. Sugiere agregar al manual cómo desfasar temporalmente una de las dos vistas, lo cual fue realizado luego de la visita. Se llega a la conclusión que se debe modificar el orden de las tareas, y agregar un tiempo estimado de ejecución total.
- Respecto a los algoritmos, encuentra un *bug* con la ventana en el algoritmo de detección de *scratches* al probar distintos tamaños de la misma. Fue solucionado.
- Respecto a los requerimientos del AGU, hace énfasis sobre la importancia de la gestión de *metadata* de video en el área de restauración y archivado.

En la segunda sesión, se buscó el mismo objetivo con el Usuario B. Este usuario, más experimentado, decidió hacer a un lado el Manual de Usuario y comenzó a probar directamente los algoritmos de restauración. Por su cuenta recreó muchos casos de uso:

- Probó cargar videos con distintas resoluciones (SD, HD), que luego usaría con los *plugins*. Al mismo tiempo, probó distintas funcionalidades de la interfaz, manipulando la línea de tiempo y la visualización del video, donde no se encontraron *bugs* ni *crashes*.

- Respecto al *plugin* de detección de *scratches*, lo probó primero con una secuencia que contenía un *scratch*. Probó con distintas combinaciones de parámetros hasta encontrar el *scratch* que buscaba, así como rellenarlo con los métodos de *inpainting* disponibles. No resultó conforme con el resultado, alegando a su vez que el video que probaba no era Full HD y existían desperfectos debidos a la la digitalización y la compresión.
- Probó resoluciones mayores a 4K, que de por sí Natron las procesa lento. Esto provocó algún *crash* por falta de memoria.
- Luego utilizó el *plugin* de *deflicker* con distintos parámetros. No obtiene buenos resultados cuando lo prueba con imágenes de distintos planos (el *plugin* funciona bien si trabaja sobre un mismo plano). Utiliza también un video en HD sin *flicker*, y el *plugin* no modifica notoriamente la imagen. Descubre alguna fuga de memoria que genera un *crash* en Natron, que tuvo que ser corregido.

En conclusión, ambas sesiones, si bien diferentes, sirvieron para corregir distintos aspectos, tanto de implementación como de documentación. Se descubrieron *bugs* y *crashes* que fueron corregidos, se hicieron ajustes al Manual de Usuario y se le agregaron funcionalidades disponibles que facilitan el trabajo del usuario.

Capítulo 6

Herramienta para Separar una Secuencia en Planos

En este capítulo se presentan dos métodos implementados para separar una secuencia en sus distintos planos en Natron [1]. Se implementaron dos algoritmos: uno implementa un método de detección de cortes semi-automático, y otro utiliza una lista de cortes dada.

6.1. Método mediante lista de cortes

En Natron, separar manualmente una secuencia en sus distintos planos es una tarea tediosa. Un usuario debe recorrer la secuencia fotograma a fotograma, identificar los cortes, y generar por cada uno la estructura interna necesaria de Natron, utilizando nodos[2].

Utilizando la API de Python (ver Sección 5.2.3) es posible crear *scripts* para modificar la interfaz de Natron y así ofrecer nuevas funcionalidades. Se creó de esta forma una nueva opción disponible en el menú de herramientas de Natron, mediante el cual es posible ejecutar el *script* que divide la película según la lista de cortes. En primer lugar se pide al usuario que seleccione un archivo donde se tienen los cortes marcados. Luego, el *script* recorre esta lista, creando para cada corte la estructura interna necesaria, en lugar de tener que realizarla el usuario. El formato que se utilizó para las listas de entrada es el de *Edit Decision List* (EDL) [50]. Es un formato estándar para describir las escenas, que contiene, además de los fotogramas donde se dividen las escenas, duración, descripción, y comienzo y fin expresados como códigos de tiempos.

[1] El problema y la importancia de la detección de cortes fueron presentados en la Sección 3.1.

[2] Por más información sobre cómo dividir una secuencia en Natron, ver el Manual de Usuario en el Anexo B.

6.2. Método de detección de cortes semi-automático

El objetivo de la detección de cortes es ayudar al usuario a detectar y segmentar correctamente una secuencia en sus distintos planos.

Se decidió utilizar el método de suma de diferencias absolutas (SDA) que dadas dos imágenes, calcula su distancia como la suma total de la diferencia píxel a píxel. Como se puede ver en el trabajo "Detección de cortes en películas'[213]' de Juan Andrés Friss de Kereki, este método es fácil de implementar, detecta correctamente los cortes duros (que son la mayoría), y arroja buenos resultados.

La implementación realizada se encuentra dividida en dos partes. Se desarrolló un algoritmo en C++ y OFX que calcula el puntaje SDA para cada imagen respecto a su imagen siguiente. En el Algoritmo 3 se presenta un pseudocódigo donde se calcula el valor de SDA para cada imagen.

Algoritmo 3: Pseudocódigo de detección de cortes por SDA

Entrada: Secuencia de imágenes I_t
para cada *imagen* I_t *de la secuencia* **hacer**
    Obtener I_{t+1}, siguiente imagen de la secuencia;
    SDA = 0;
    para cada *píxel* p **hacer**
        para cada *canal* c **hacer**
            SDA += abs($I_{t+1}(p_c)$ - $I_t(p_c)$);
        fin
    fin
    Asignar puntaje a I_t con SDA calculada;
fin
Salida: Cada imagen con su puntaje SDA asociado

También se implementó un *script* de Python. Luego de calcular el puntaje SDA para cada fotograma, el usuario selecciona un valor de umbral que le parece adecuado para generar los cortes. Entonces, este *script* recorre toda la secuencia, generando la estructura interna de Natron necesaria cada vez que se encuentra con un valor mayor al indicado por el usuario. Este método es semi-automático ya que requiere de la interacción del usuario para decidir el umbral de SDA a partir del cual se define la presencia de un corte.

6.2.1. Parámetros

El método manual tiene un único parámetro de entrada, que es la lista de cortes en formato de EDL a partir del cual se generan los cortes.

El método semiautomático tiene un parámetro de entrada en el paso intermedio. La ejecución del algoritmo que calcula el SDA para cada fotograma no requiere ningún parámetro. Luego, el usuario ajusta el umbral o *threshold* a partir del cual se generarán los cortes, lo cual es el parámetro de entrada para el *script* Python. Este umbral puede ser cualquier número entre 0 y 1; generalmente será mayor a 0.1, aunque su elección varía mucho según la secuencia.

6.3. Evaluación

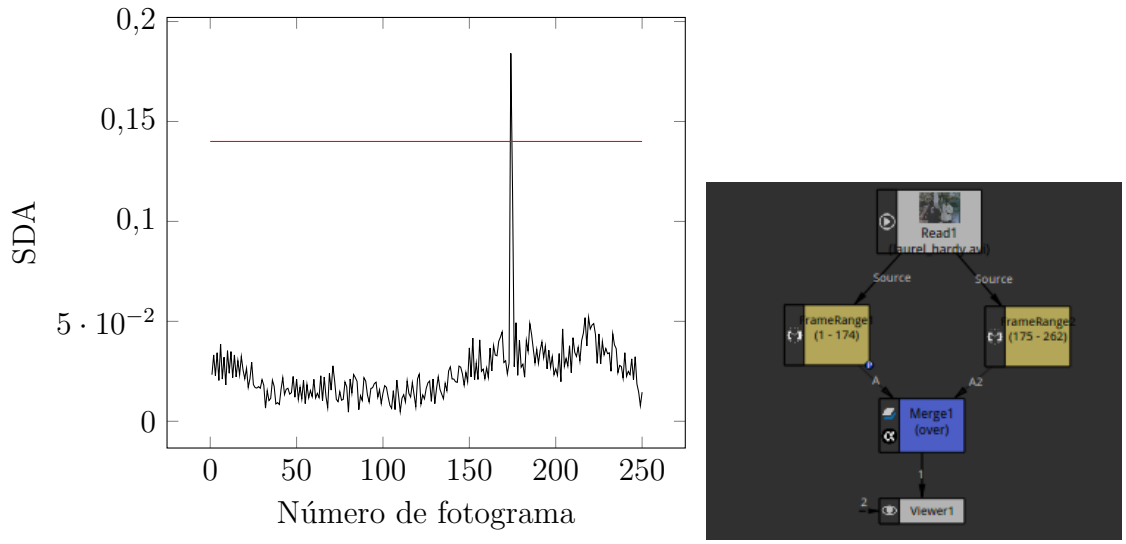
En primer lugar se trabajó sobre el video de prueba "*Laurel and Hardy*" de 250 fotogramas [214]. La secuencia presenta un único corte, en el fotograma 175; véase la Figura 6.1. Este corte no es duro, ya que por falta de precisión en la digitalización, se superponen imágenes de los dos planos.

(a) (b) (c)

Figura 6.1: Corte en "*Laurel and Hardy*"

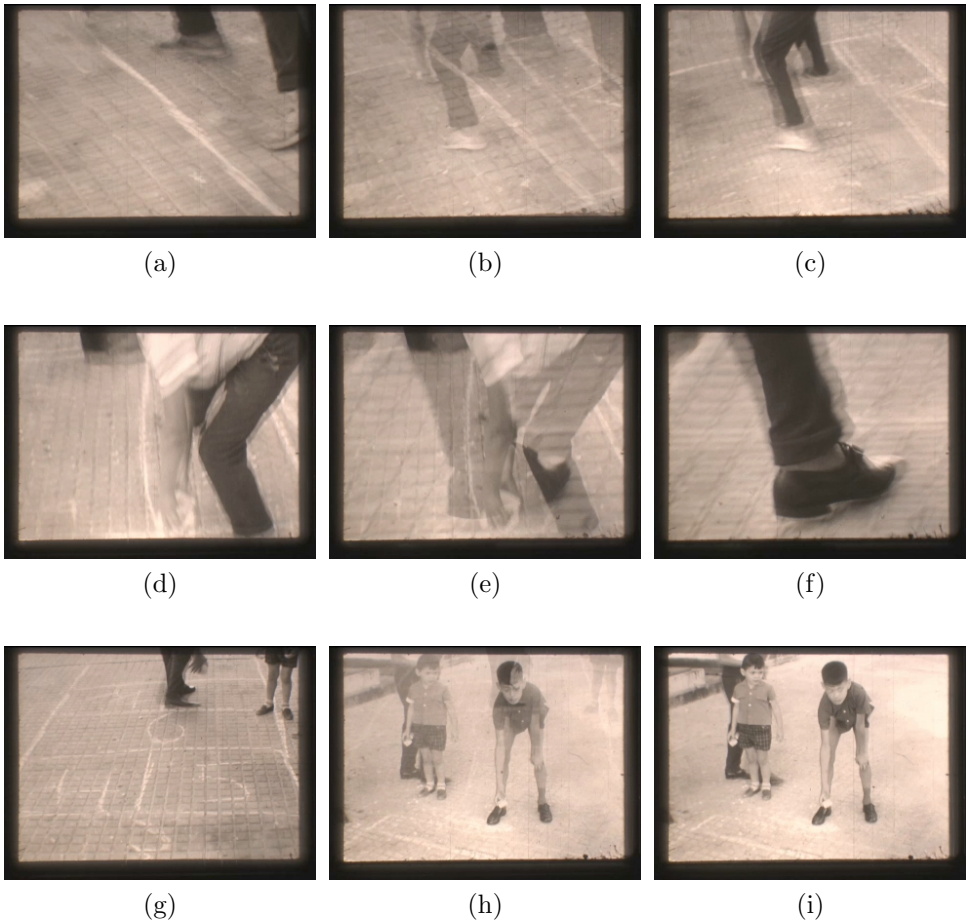
Se ejecutó el algoritmo sobre esta secuencia, obteniendo los valores de SDA que se muestran en la Figura 6.2a. En particular, en el fotograma 174 se obtuvo un valor de SDA de 0.184, y en el 175 de 0.115; los puntajes para los demás fotogramas fueron menores a 0.1. Por lo tanto, cualquier umbral mayor a 0.115 y menor a 0.184 serviría. En particular en este ejemplo se eligió un umbral de 0.14. Al hacer esto y generar los nodos (ver Manual de Usuario), se crea la configuración de nodos que se puede ver en la Figura 6.2b.

Se realizó otra prueba sobre una secuencia que presentó falsos positivos. La secuencia utilizada fue "Juegos y Rondas", de más de 700 fotogramas de largo. Presenta tres cortes, en los fotogramas 144, 417 y 623; véase la Figura 6.3.



(a) (b)

Figura 6.2: (a) Valores de SDA para cada fotograma de "*Laurel and Hardy*". En rojo, una posible elección de umbral por parte de un usuario que generaría el corte correcto. (b) Nodos generados al utilizar el algoritmo de detección de cortes y un umbral de 0.14.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 6.3: Los tres cortes existentes en la secuencia de "Juegos y Rondas" utilizada. Ninguno es un corte duro; todos son producto de una digitalización poco precisa.

Al ejecutar el algoritmo que calcula el puntaje SDA, los tres cortes presentan valores de 0.207, 0.288 y 0.455. Sin embargo, existen otros fotogramas con valores cercanos al 0.3, como se muestra en la Figura 6.4. Estos casos son de movimientos rápidos (un hombre caminando, un niño cayendo) por lo que la diferencia entre fotogramas genera un valor alto de SDA. La gráfica con los puntajes SDA obtenidos para todos los fotogramas se puede ver en la Figura 6.5, donde se selecciona un umbral de 0.287.

(a) (b)

(c) (d)

Figura 6.4: Falsos positivos en "Juegos y Rondas". Movimientos rápidos y zonas muy claras u oscuras que queden descubiertas entre fotogramas generan un valor alto de SDA.

Por último, se generan los nodos como se muestra en la Figura 6.6. El último paso sería eliminar los nodos creados erróneamente por los falsos positivos detectados, y crear uno para el corte existente en el fotograma 144, cuyo valor de SDA cayó debajo del umbral.

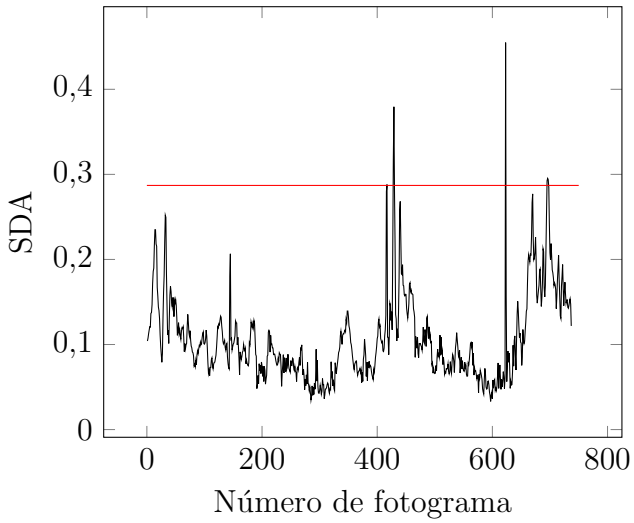


Figura 6.5: Valores de SDA en una secuencia de "Juegos y Rondas". En rojo, un posible umbral de 0.287.

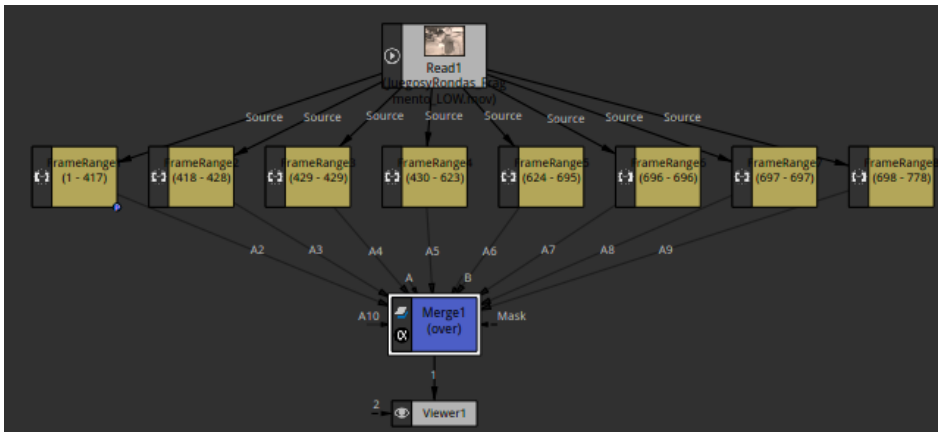


Figura 6.6: Nodos generados para la secuencia de 'Juegos y Rondas" al utilizar el algoritmo de detección de cortes y un umbral de 0.287.

Capítulo 7

Herramienta de Corrección de *Flicker*

En este capítulo se presenta el algoritmo para corregir el *flicker* de luminancia. El objetivo es restaurar una secuencia de imágenes que presenta degradación por *flicker*, para lograr una iluminación y contraste más natural y consistente. [1]

Se eligió, como base, la implementación del método "*Midway Image Equalization*" propuesto en el artículo ''*Implementation of the Midway Image Equalization*" escrito por Thierry Guillemot y Julie Delon [127], y su extensión a video con el método global STE (ver Sección 3.2). Este método fue elegido porque al ser global es más general y robusto, y aplicable a más casos que los métodos locales, además de generar menos artefactos ante la presencia de manchas y *scratches* [126], deterioros comunes y presentes en las películas deterioradas.

Una propiedad interesante que debería presentar un algoritmo de *deflicker* es que "sobre una secuencia libre de *flicker*, el proceso de estabilización debería tener la habilidad de dejar la secuencia intacta"[126].

7.1. Descripción del algoritmo

La idea principal del "*Midway Image Equalization*" es aplicar el mismo "histograma medio" a dos imágenes consecutivas, esto es, transformar ambas imágenes para que tengan el mismo histograma, pero manteniendo lo más posible la información y contenido de las imágenes. Si imaginamos dos imágenes, una más oscura y otra más clara, el efecto de aplicar el algoritmo aclara la primera, y oscurece la segunda. Este algoritmo se puede aplicar para corregir problemas de *flicker*(ver [215] y [216]).

El método se propone para imágenes monocromáticas (un solo canal de información), caso que se utilizó para la implementación de este proyecto. Sin embargo,

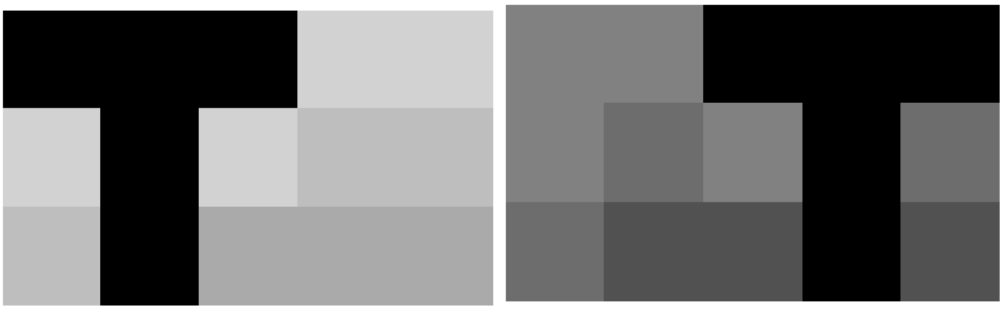
[1]Dicho problema fue presentado cuando se introducen los problemas asociados al celuloide en la Sección 2.2, mientras que diferentes algoritmos que existen para solucionarlo fueron presentados en la Sección de 3.2 de Estado del Arte.

se puede extender a imágenes en colores (tres canales), tratando cada canal por separado y luego juntándolos [2].

Al tratarse de una película, dos imágenes consecutivas tienen las mismas dimensiones y trabajan con los mismos tipos de datos (por ejemplo, valores enteros entre 0 y 255). En estos casos, el cálculo del histograma común y su posterior aplicación se puede simplificar de gran manera con referencia a la versión original del algoritmo [3]. Lo que el algoritmo hace en este caso es lo siguiente. Tomemos cada imagen como una matriz de dimensiones $M \times N$ y valores entre 0 y 255. Entonces:

1. Para cada imagen, se crea un vector de tamaño $M \times N$ con todos los valores de grises ordenados de menor a mayor, y con una referencia a la posición original del píxel.
2. Se crea el "histograma común", simplemente promediando los dos vectores generados en el primer paso.
3. Por último, se asignan los valores del vector generado en el paso dos a cada una de las imágenes según el vector de índices.

Para entender este proceso mejor, se presenta el siguiente ejemplo detallando los pasos de la ejecución. En la Figura 7.1 se tienen dos fotogramas, donde se representa una T oscura que se mueve sobre un fondo que presenta *flicker*: se oscurece de un fotograma al siguiente.



(a) (b)

Figura 7.1: Dos fotogramas consecutivos que presentan el problema de *flicker*. En (a) el fondo está iluminado pero en (b) el fondo está oscuro.

Las matrices que representan a estas imágenes son las siguientes:

[2]Esta es una solución simple y fácil de aplicar, con resultados aceptables aunque podría generar artefactos de color [127]. Una mejor solución implicaría aplicar técnicas de "*optimal transportation*" (ver [217] y [218]).

[3] Si al lector le interesa probar el algoritmo, en el artículo se puede encontrar una implementación de solo ocho líneas que se puede ejecutar tanto en Matlab como Octave.

$$imagen\_a = \begin{bmatrix} 0 & 0 & 0 & 210 & 210 \\ 210 & 0 & 210 & 190 & 190 \\ 190 & 0 & 170 & 170 & 170 \end{bmatrix} \quad imagen\_b = \begin{bmatrix} 130 & 130 & 0 & 0 & 0 \\ 130 & 110 & 130 & 0 & 110 \\ 110 & 80 & 80 & 0 & 80 \end{bmatrix}$$

En el primer paso de la ejecución, para cada imagen o matriz se ordenan sus valores en un vector de MxN, en este caso de largo 15, almacenando a su vez los índices con la posición original de cada uno de estos valores.

$$sort\_imagen\_a = \langle 0, 0, 0, 0, 0, 170, 170, 170, 190, 190, 190, 210, 210, 210, 210 \rangle$$

$$sort\_idx\_a = \langle 1, 4, 5, 6, 7, 9, 12, 15, 3, 11, 14, 2, 8, 10, 13 \rangle$$

$$sort\_imagen\_b = \langle 0, 0, 0, 0, 0, 80, 80, 80, 110, 110, 110, 130, 130, 130, 130 \rangle$$

$$sort\_idx\_b = \langle 7, 10, 11, 12, 13, 6, 9, 15, 3, 5, 14, 1, 2, 4, 8 \rangle$$

En el segundo paso, se crea el "histograma común", promediando los dos vectores ordenados de valores de gris generados en el paso anterior:

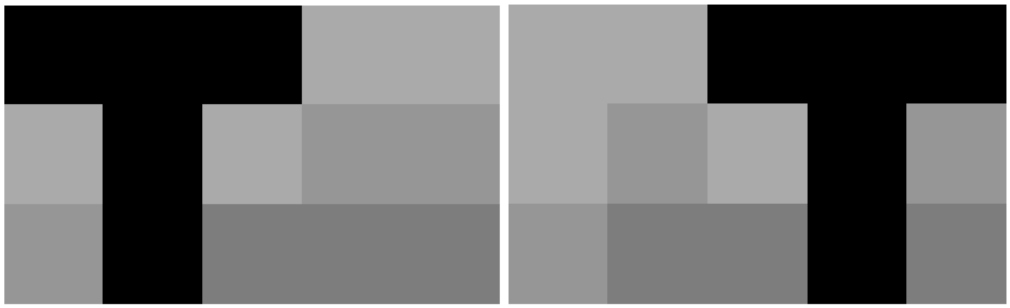
$$histograma\_comun = \langle 0, 0, 0, 0, 0, 125, 125, 125, 150, 150, 150, 170, 170, 170, 170 \rangle$$

Por último, se asignan los valores del "histograma común" en cada una de las imágenes, en la posición que indican los índices. Por ejemplo, para generar la primera imagen, el primer 0 va en la posición 1, el segundo 0 en la posición 4, y asi sucesivamente.

$$restaurada\_a = \begin{bmatrix} 0 & 0 & 0 & 170 & 170 \\ 170 & 0 & 170 & 150 & 150 \\ 150 & 0 & 125 & 125 & 125 \end{bmatrix} \quad restaurada\_b = \begin{bmatrix} 170 & 170 & 0 & 0 & 0 \\ 170 & 150 & 170 & 0 & 150 \\ 150 & 125 & 125 & 0 & 125 \end{bmatrix}$$

Las imágenes restauradas se muestran en la Figura 7.2.

Extensión a secuencias de imágenes Para trabajar con secuencias de imágenes, se tomó como base la extensión a video STE[219], que presenta cómo utilizar el algoritmo "*Midway Histogram Equalization*" en secuencias de imágenes para restaurar el problema de *flicker*.



(a) (b)

Figura 7.2: Dos fotogramas consecutivos que presentaban el problema de *flicker*, restaurados luego de aplicar el algoritmo *"Midway Image Equalization"*. En (a) el fondo se oscurece y en (b) el fondo se aclara.

Existen dos diferencias con la propuesta original del algoritmo. La primera es que para trabajar con p fotogramas, en la versión original simplemente se promedia el vector ordenado de cada imagen con un peso de *1/p* para cada fotograma. En cambio, en el artículo "*GPU Based Implementation of Film Flicker Reduction Algorithms*"[219] de Martín Piñeyro y colegas, se utiliza una distribución normal para asignar los pesos. Según el mismo, el mejor valor para la varianza es 2 por la cantidad de fotogramas T_n. Este es el valor que se utiliza en la implementación realizada, donde se genera un vector de pesos según la distribución normal de varianza $\sigma^2 = 2 \times T_n$. Como se ilustra en la Figura 7.3, el valor central de este vector es el que se asigna al fotograma que se está procesando. Por ser una distribución normal, la suma de todos los valores del vector es igual a 1.

La segunda diferencia es que en OpenFX sólo se modifica la imagen actual que se está procesando. Para generar su histograma, se utilizan también las imágenes anteriores y siguientes, pero estas no se modifican. Por esto, no necesariamente las imágenes de una secuencia tendrán finalmente el mismo histograma.

Dithering Al aplicar el algoritmo, si las imágenes presentaban histogramas muy distintos, pueden aparecer artefactos debido a errores de cuantización. Esto no ocurre generalmente si se aplica sobre fotogramas pertenecientes a un mismo plano. Se puede resolver este problema agregando ruido Gaussiano a las imágenes previo a aplicar el algoritmo [127], a esta técnica se le denomina *dithering*.

7.2. Implementación realizada

Se provee en el Algoritmo 4 un pseudocódigo de la implementación realizada. En el mismo, se pueden observar los pasos descritos en la sección anterior. Se especifican luego algunas aclaraciones sobre el código, así como casos especiales y

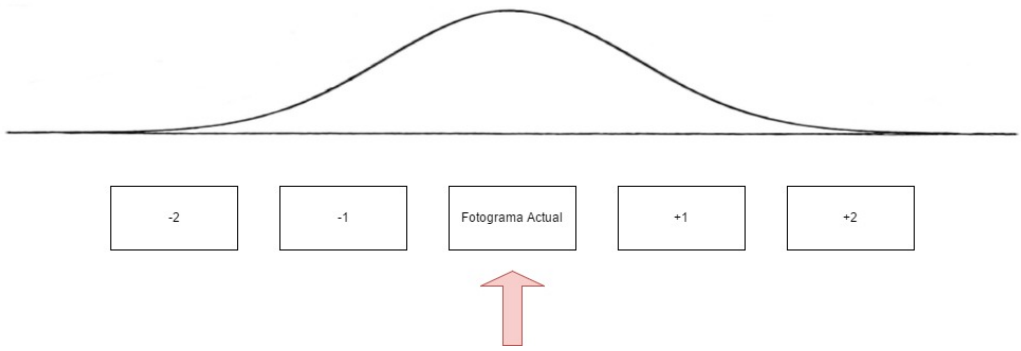


Figura 7.3: Ejemplo ilustrativo del promedio ponderado con una ventana de radio 2.

casos de borde.

Algoritmo 4: Pseudocódigo de algoritmo de corrección de *flicker*

Entrada: Secuencia de imágenes I_i
para cada *imagen* I_i *de la secuencia* **hacer**
  Asignar en ventana el parámetro de entrada T_n;
  Calcular pesos utilizando la ventana;
  Inicializar histograma;
  para cada *imagen J anterior y posterior dentro de la ventana* **hacer**
    Crear el vector ordenado de J;
    Acumular en histograma común, el vector ordenado multiplicado por su peso correspondiente;
  fin
  Asignar a la imagen I el histograma común calculado;
fin
Salida: Cada imagen restaurada

El primer bucle recorre todas las imágenes de la secuencia de entrada. Aunque desde el punto de vista del algoritmo en sí no es importante, la secuencia de entrada debería pertenecer a la misma toma, o al menos a la misma escena. Si no, se podrían generar resultados deficientes al trabajar con fotogramas de escenas distintas.

Luego, se toma el único parámetro de entrada que tiene el algoritmo, que es el largo de la ventana temporal o cantidad de fotogramas sobre los cuales se trabajará, T_n. Este número es impar, por lo cual el radio se calcula como $(T_n\text{-}1)/2$.

Luego, utilizando el largo de la ventana T_n, se crea el vector con los pesos a asignar a cada imagen, centrado en la imagen actual que se está procesando. Para

una imagen i, su peso se calcula como: $w_i = exp(-\frac{(i-T_n/2)^2}{2\sigma^2})$ donde el valor de σ^2 que se utiliza es $2T_n$. A su vez, se crea el vector histograma, donde se irán almacenando los valores que finalmente se asignarán en la imagen actual.

Luego, otro bucle recorre las imágenes de la secuencia que caen dentro de la ventana de fotogramas. Para cada una, se ordenan sus valores de gris, y se realiza una suma de vectores, multiplicando sus valores por el peso que se le debe asignar a esta imagen.

Luego se consideran los casos de borde, osea cuando se están procesando fotogramas del principio o final de la secuencia. En estos casos, existen valores del vector que no se utilizarán, ya que su imagen correspondiente no existe. Entonces, se normaliza teniendo esto en cuenta para que nuevamente la suma de los valores del vector de 1.

Finalmente, se calculan los índices de la imagen actual, y según ellos se asignan los valores del histograma que se creó. Para realizar las tareas de ordenamiento, se utilizaron las funciones *sort* [220] y *sortIdx* [221] de la librería OpenCV. Para calcular el valor de x (columna) y de y (fila) en la imagen de OpenFX a partir de un índice, se hace lo siguiente, teniendo en cuenta que OpenFX lee las columnas de izquierda a derecha, y las filas de abajo hacia arriba. El valor de x es el resto de la división entera del índice entre el ancho de la imagen. El valor de y es el resultado de la división entera del índice entre el ancho de la imagen.

7.3. Parámetros

Este algoritmo tiene un sólo parámetro de entrada, la ventana de fotogramas a tomar en cuenta cuando se realiza el histograma a ser aplicado en la imagen actual. Este parámetro es un entero impar. Para entender su aplicación en el algoritmo, se presenta un ejemplo concreto. Suponiendo que vale siete, entonces el radio de la ventana será de tres: se tomará el fotograma actual, los tres siguientes, y los tres anteriores. Por ende, cuanto mayor sea este parámetro, mayor tiempo de ejecución, ya que se deberán procesar más fotogramas. Sin embargo, se muestra en la siguiente sección que se pueden obtener mejores resultados al aumentar esta ventana, teniendo como única contrapartida el costo computacional. Se debe tener en cuenta que extender demasiado este parámetro generaría que toda la secuencia tenga una iluminación pareja y esto puede afectar la fotografía original e intencional de la película.

7.4. Evaluación

En esta sección se presentan resultados cualitativos al aplicar el algoritmo implementado sobre una secuencia.

En una primera prueba, se aplicó *flicker* sintético a una escena de "Juegos y Rondas" provista por el AGU. Esta se presenta en la Figura 7.4.

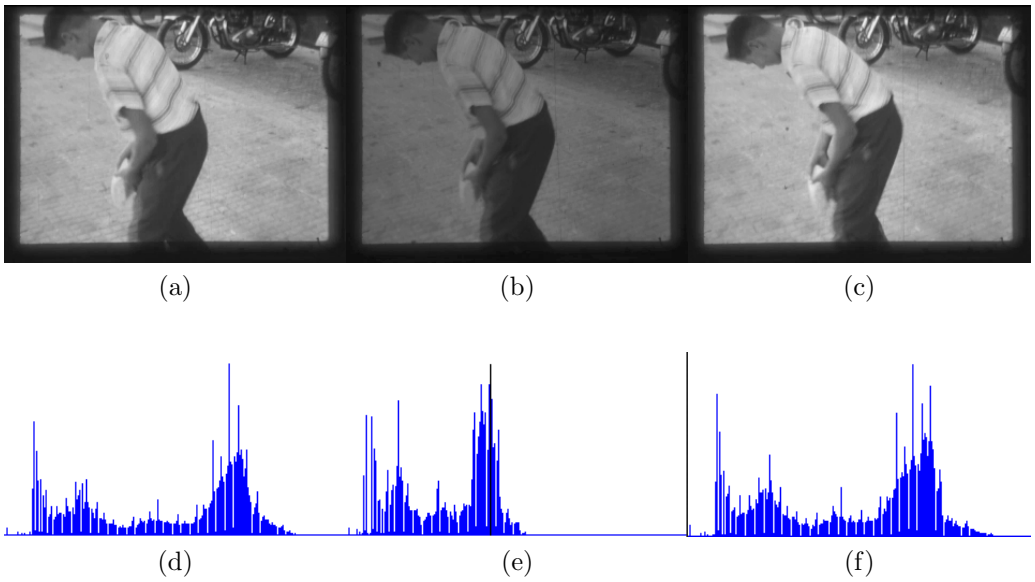
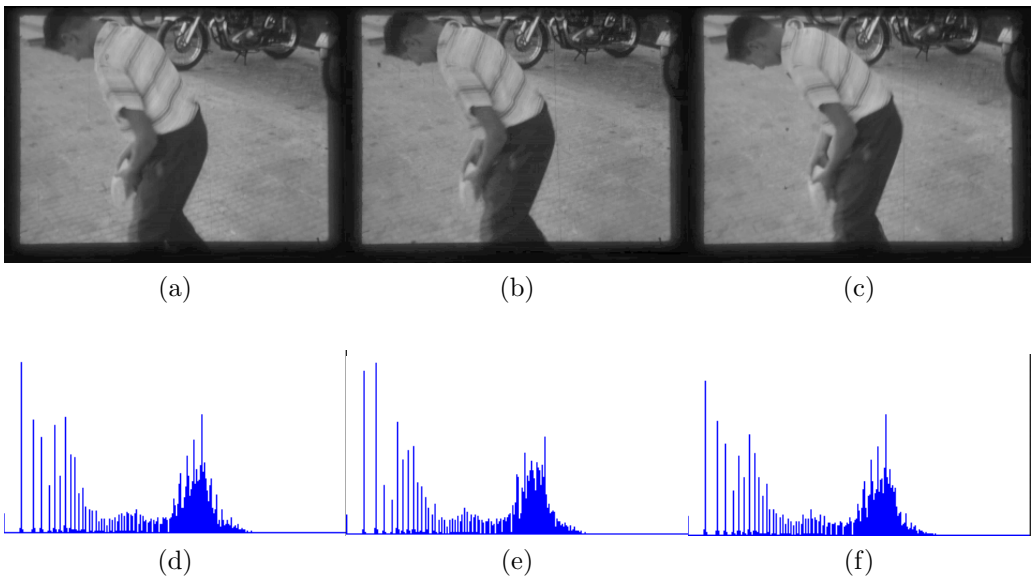


Figura 7.4: Tres fotogramas de una escena de "Juegos y Rondas" con *flicker* sintético aplicado, y sus respectivos histogramas. Se observa que los valores de los histogramas de las imágenes claras se encuentran sobre la derecha, o valores con mayor luminosidad, mientras que en la imagen oscura el histograma se concentra sobre valores más bajos.

Se aplicó el algoritmo implementado, obteniendo los resultados de la Figura 7.5. Las tres imágenes comparten ahora tonos de grises similares, y sus histogramas son muy parecidos.



(a) (b) (c)

(d) (e) (f)

Figura 7.5: Imágenes restauradas utilizando el algoritmo. Presentan histogramas similares, y la concentración de valores se encuentra sobre valores más centrales que en las imágenes anteriores.

Otra prueba realizada es la siguiente, donde se estudió una escena de un video provisto por el AGU, "Buscando un Diabético" (Tálice). Se trabajó con formato Full HD. Se estudió y graficó la luminancia de la secuencia (utilizando el espacio de color HSL [222]), y se notó que presenta una fluctuación en la luminancia global cada aproximadamente diez fotogramas. Entonces, se aplicó el algoritmo para verificar que suavizara la diferencia entre estos valores. El valor que se eligió para el parámetro de entrada (la ventana de fotogramas) fue 11, o sea un radio de cinco fotogramas. En la Figura 7.6 se pueden ver fotogramas de la escena original y los correspondientes fotogramas restaurados. En la Figura 7.7 se pueden ver los valores de luminancia para diez fotogramas consecutivos de la escena, tanto para los originales como los restaurados.

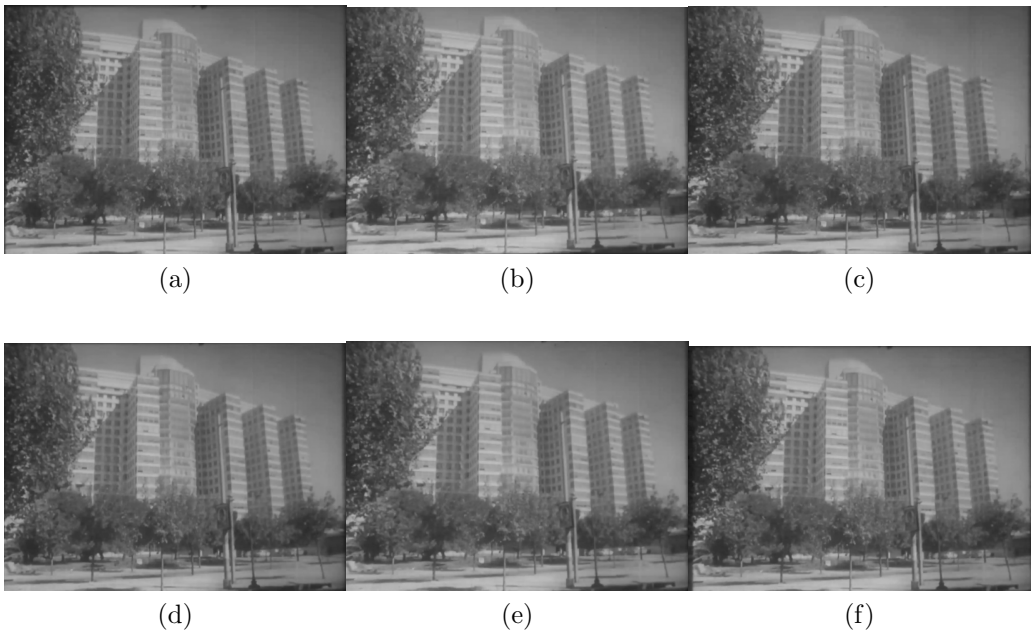


Figura 7.6: Arriba, tres fotogramas de una escena de "Buscando un Diabético" que presenta *flicker*. La imagen central (b) es más clara que (a) y (c). La degradación es más notoria cuando se reproduce el video, que en imágenes fijas. Abajo, los mismos fotogramas pero restaurados.

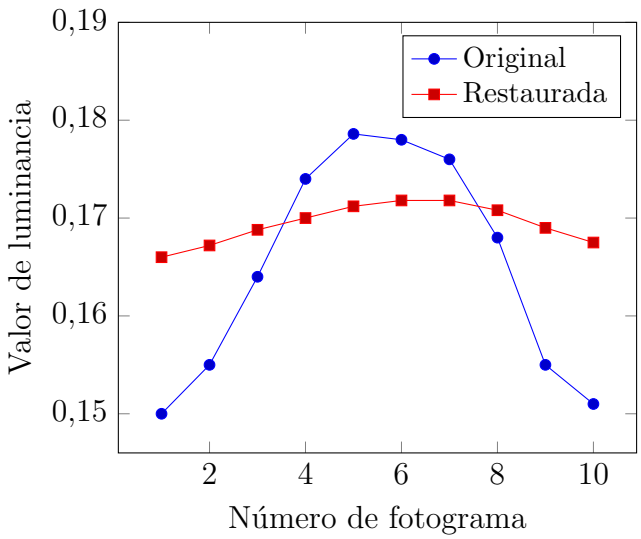
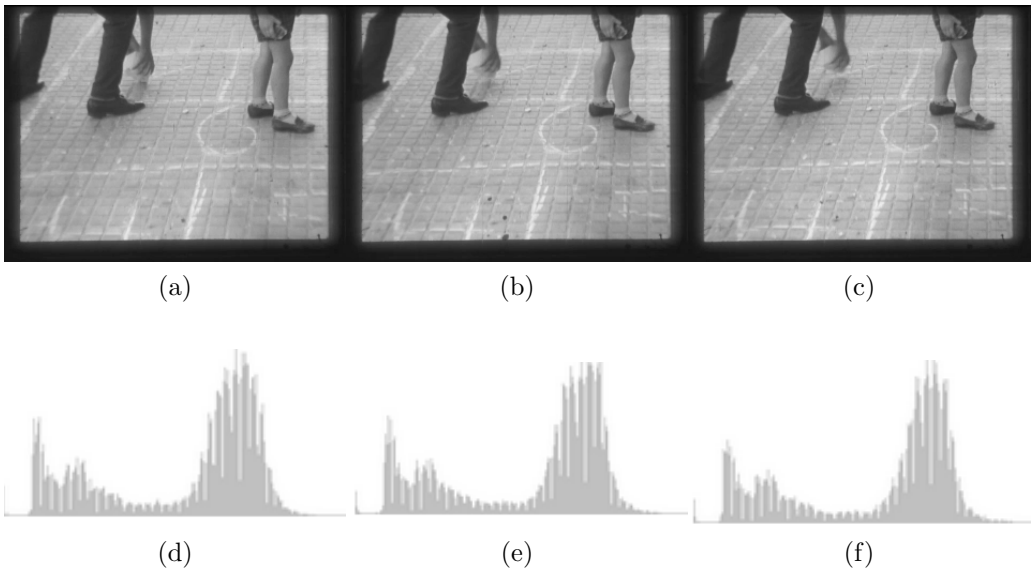


Figura 7.7: Valores promedio de la luminancia global en una secuencia de diez fotogramas, antes y después de aplicar el algoritmo de *deflicker*.

Por último, se estudió el efecto de aplicar el algoritmo sobre una secuencia sin *flicker*, y se comprobó que el algoritmo no genera artefactos ni pérdida de información de la imagen. Se incluyen los histogramas de tres fotogramas consecutivos

de una escena de "Juegos y Rondas" en la Figura 7.8 y se hace lo mismo para las imágenes ya procesadas en la Figura 7.9. El parámetro utilizado fue de 13 fotogramas.

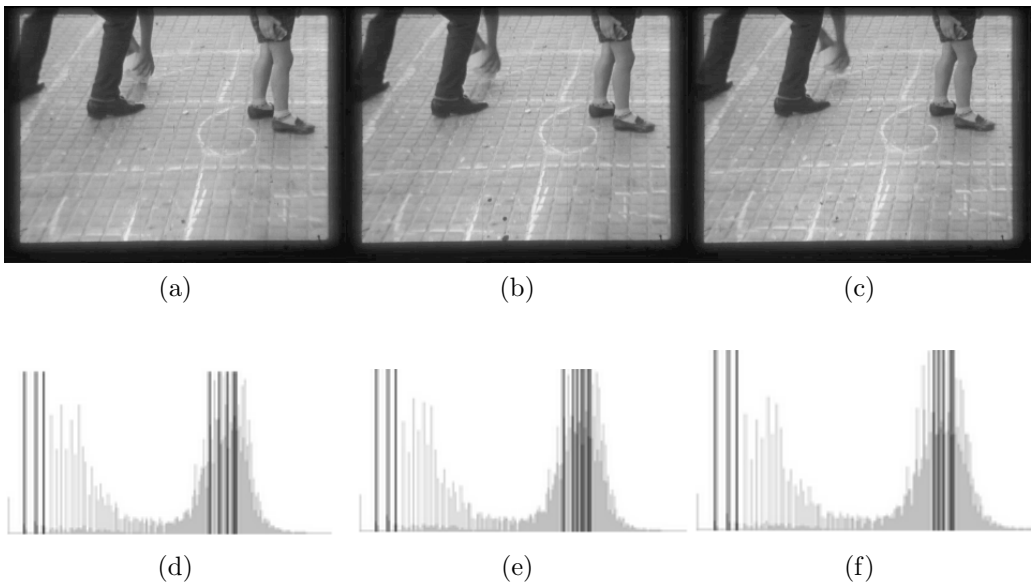


(a) (b) (c)

(d) (e) (f)

Figura 7.8: Tres fotogramas consecutivos de una escena de "Juegos y Rondas" y sus histogramas correspondientes. La escena no presenta *flicker*.

Al aplicarles el algoritmo se han modificado sus histogramas y ahora presentan valores similares entre ellos y algunos picos que anteriormente no tenían. Sin embargo, el contenido de las imágenes no ha variado prácticamente a nivel cualitativo.



(a) (b) (c)

(d) (e) (f)

Figura 7.9: Imágenes restauradas utilizando el algoritmo. Las imágenes originales no tenían *flicker*, por lo que las restauradas no varían mucho, aunque si sus histogramas.

Capítulo 8

Herramienta de Detección y Restauración de *Scratches*

En este capítulo se presenta la implementación de un algoritmo de detección y restauración de *scratches* semi-automático. El objetivo es implementar un algoritmo que logre reparar de forma precisa este desperfecto común a gran parte de las películas deterioradas.[1]

El método desarrollado de detección de *scratches* se basa en las ideas propuestas en el artículo "*Robust Automatic Line Scratch Detection in Films*" de Andrés Almansa y colegas [147], en 2014.

Se eligió este algoritmo porque proporciona resultados robustos al ruido y las texturas, con precisión a escala del píxel (ver Sección 3.4), y es más preciso y eficiente que otras técnicas propuestas anteriormente [150].

Para restaurar los *scratches* se deben rellenar de forma automática las detecciones obtenidas, y el resultado debe ser indetectable para el espectador. Para ello se decidió utilizar técnicas de *image inpainting*[2].

La elección de estos métodos está basada fuertemente en la pre-existencia de implementaciones eficientes de *image inpainting*, tanto en la librería libre de *computer vision* OpenCV, como en Natron.

En muchos casos es posible restaurar apropiadamente los *scratches*. Sin embargo al utilizar técnicas de *image inpainting* en video, los resultados presentan inconsistencias temporales (ver Sección 3.6). La extensión a *video inpainting* es un tema importante que se propone abordar como trabajo a futuro.

[1] Dicho problema fue presentado cuando se introducen los problemas asociados al celuloide en la Sección 2.2, y diferentes algoritmos que existen para solucionarlo fueron presentados en la Sección 3.4 de Estado del Arte.

[2] Las técnicas de *image inpainting* fueron presentadas en la Sección 3.6.

8.1. Resumen del algoritmo de detección espacial adaptativa de *scratches*

El algoritmo de detección espacial de *scratches* elegido trabaja en tres etapas principales:

1. Se realiza una detección *per-pixel* de potenciales *scratches*.
2. Se aplica la transformada de Hough para acelerar el proceso.
3. Se utiliza una metodología *a contrario* para agrupar los píxeles detectados en segmentos (*scratches*) visualmente significativos. Para esto se utiliza un modelo de fondo para que el método sea robusto al ruido y a la propia textura de la imagen.

1. **Criterio de detección binaria *per-pixel***

   En primer lugar se define un modelo para el perfil horizontal en los niveles de gris de un *scratch*. [3].

   Presentando un ejemplo ilustrativo, dada una línea horizontal con los niveles de gris de la imagen, si imaginamos que la línea es oscura a excepción de cinco píxeles contiguos que son claros, entonces es probable que estos pertenezcan a un *scratch*. Esto se formaliza y extiende a distintos casos, como se presenta a continuación.

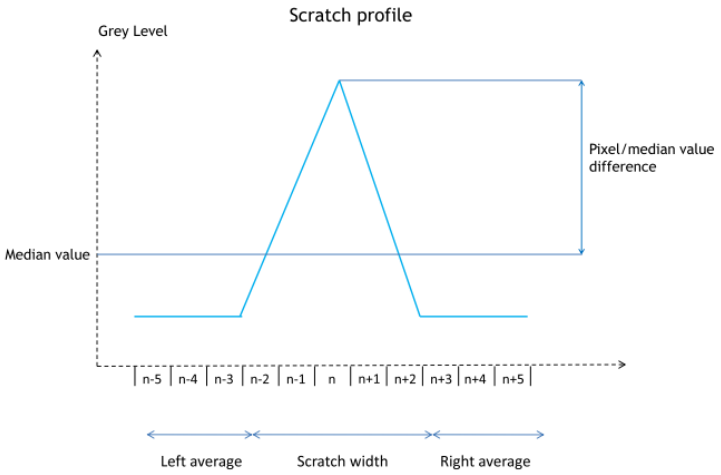
   Para la imagen original I, se establece un umbral que determina si el nivel de gris de un píxel considerado no es consistente con el nivel de gris de sus vecinos horizontales:

$$c_1(x,y) : |I_g(x,y) - I_m(x,y)| \geq s_{med} = 20 \tag{8.1}$$

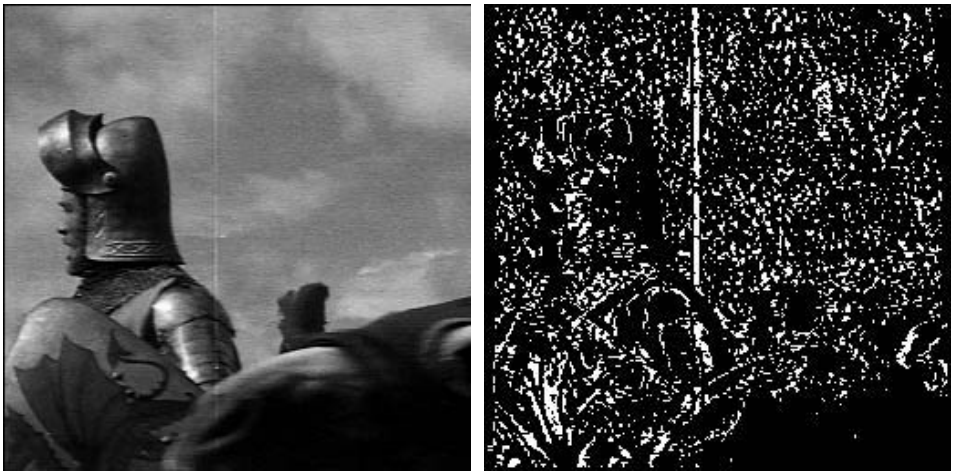
$$c_2(x,y) : |I_d(x,y) - I_i(x,y)| \leq s_{avg} = 3 \tag{8.2}$$

   $I_g(x,y)$ resulta de aplicar a I un filtro de desenfoque *gaussiano* de *kernel* 3x3 y desviación estándar de un píxel, esto reduce el ruido común en películas deterioradas. $I_m(x,y)$ es una imagen generada calculando la mediana horizontal de cinco píxeles vecinos en $I_g(x,y)$, sin considerar el píxel (x,y). De esta forma la Ecuación (8.1) impone que el valor del píxel (x,y) debe alejarse un valor s_{med} del valor de su mediana horizontal, para pertenecer a un *scratch*.

[3] Para ello se utilizan las ideas propuestas originalmente en el libro "*Motion Picture Restoration. Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video*" [140] de Anil Kokaram, modificando algunos elecciones, tal como se presenta a esta sección.



(a)



(b) (c)

Figura 8.1: (a) Modelo del perfil horizontal del *scratch* para el método de detección binaria.[147]. (b) Fotograma de la secuencia "*Knight*" con resolución 257×257 (c) Detección binaria de *scratches* obtenida con la implementación del método.

La Ecuación (8.2) impone que los promedios del nivel de gris a los lados del *scratch* sean similares. $I_d(x, y)$ y $I_i(x, y)$ son los promedios a derecha y a izquierda respectivamente.

En el artículo citado se fijaron empíricamente los valores de $s_{med} = 3$ y $s_{avg} = 20$ en las Ecuaciones (8.1) y (8.2). Finalmente la detección binaria se realiza mediante la siguiente expresión:

$$I_B(x, y) = \begin{cases} 1 \;\; si \;\; c_1(x, y) \; y \; c_2(x, y) \\ 0 \text{ en otro caso} \end{cases}$$

Se puede observar el resultado en las Figuras 8.1a y 8.1b.

2. **Agrupamiento de los píxeles detectados en segmentos significativos**

   El resultado anterior genera falsas detecciones debido al ruido y a la propia textura de la imagen. Por ello se utiliza una etapa posterior que permite agrupar píxeles, de forma robusta, en segmentos visualmente significativos, utilizando una metodología *a contrario* propuesta en el libro "*From Gestalt theory to image analysis*"[223], de Desolneaux *et al.*

   La metodología *a contrario* se basa en el principio de Helmholtz, que establece que nuestra percepción no permite distinguir estructuras (denominadas "*Gestalts*") dentro de una imagen aleatoria, a menos que se produzca una desviación considerable de dicha aleatoriedad.

   Para aplicar dicho principio se requiere establecer un modelo de fondo apropiado. En este caso se tiene de fondo la imagen binaria $I_B(x, y)$. Dado un segmento formado por l píxeles, a cada píxel x_i se le asocia una variable aleatoria X_i, dicha variable toma valor 1 si el píxel está alineado con el segmento, o 0 en otro caso. Sea $S_l = \sum X_i$, asumiendo que las V.A. X_i son independientes e idénticamente distribuidas (*iid*) y que tienen distribución Bernoulli de parámetro $p \in [0, 1]$, la probabilidad de encontrar al menos k_0 píxeles alineados en el segmento sigue la distribución binomial:

$$X_i \sim Ber(p) \Rightarrow Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} \binom{l}{k} p^k (1-p)^{l-k} =: Bin(p; l, k_0) \quad (8.3)$$

   Un segmento es significativo si $Bin(p; l, k_o)$ es suficientemente pequeño, lo que implica que la probabilidad de que aparezca en el modelo de fondo es pequeña. Esto a su vez depende del número de segmentos considerados N_{tests}. El número de falsas alarmas (NFA) se define como:

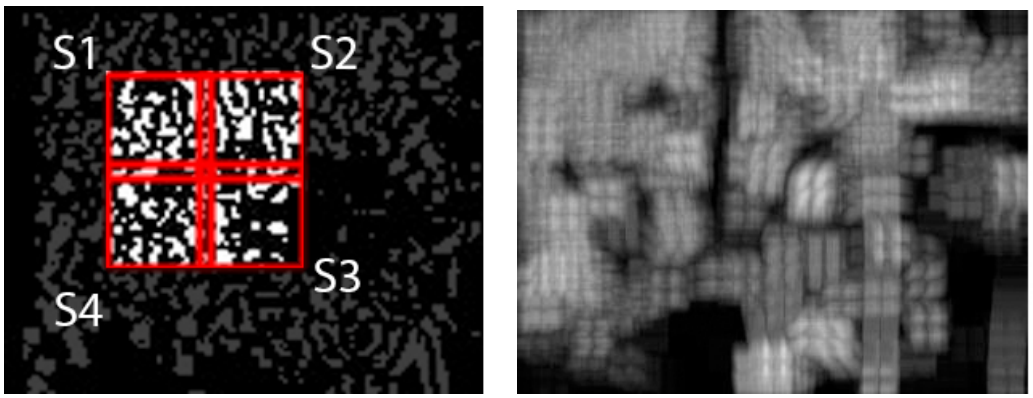
$$NFA(l, k_0) = N_{tests} B(p; k_o, l)$$

   Un segmento se considera significativo cuando su NFA está por debajo de un cierto umbral ε. La elección de dicho parámetro se presenta a continuación, luego de establecer un modelo de fondo apropiado.

3. **Agrupamiento adaptativo localizado**

   El modelo de fondo utilizado también debe considerar que la probabilidad de detección varía localmente con la densidad de píxeles en la imagen binaria I_B, ya que las detecciones *per-pixel* obtenidas generan falsos positivos en las regiones de mucha textura de la imagen, así como también fallan ante la presencia de ruido.

Se propone, para obtener la densidad de píxeles de la imagen I_B, calcular para cada píxel (x, y) la máxima densidad entre cuatro cuadrados $L \times L$ en torno al píxel considerado (ver Fig. 8.2). Los cuadrados tienen lados de largo L, proporcional al ancho de la imagen N, y se fija $L = \frac{N}{30}$. Se demuestra experimentalmente que con dicho valor se obtienen mejores resultados para varios videos de prueba [147].



(a) (b)

Figura 8.2: (a) *Zoom* en la imagen I_B de dimensión $M \times N$ para un ejemplo. En cada píxel se toma la máxima densidad entre cuadros de ancho $L = \frac{\mathrm{M}}{30}$. (b) Mapa densidad de píxeles obtenido.

Adoptando este modelo deja de ser válido el resultado de la Ecuación (8.3). En este caso se tiene la distribución binomial de Poisson:

$$Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} \sum_{\sum x_i = k} \prod_{i=1}^{l} p_i^{x_i} (1 - p_i)^{x_i} \tag{8.4}$$

La expresión anterior resulta muy costosa computacionalmente y se utiliza como aproximación conservadora la desigualdad de Hoeffding [223], que provee una cota superior a la probabilidad $Pr(S_l \geq k_0)$:

$$Pr(S_l \geq k_0) \leq H(l, k_0) := exp[-l(log\frac{r}{p_m}) + (1 - r)log(\frac{1 - r}{1 - p_m})] \tag{8.5}$$

Donde p_m es la densidad promedio entre todos los píxeles del segmento, $r = \frac{k_0}{l}$ y $p_m \times l < k_0 < l$. Se detecta un segmento significativo si se cumple:

$$NFA(l, k_0) = N_{tests} H(l, k_0) \leq \varepsilon = 1 \tag{8.6}$$

Se fija $\varepsilon = 1$. Esta elección es razonable dado que ε es una cota al número esperado de falsas alarmas bajo el modelo de fondo. De todas formas, es posible ajustar ε para alterar la cantidad de detecciones (ver Sección 8.5).

Se fija $N_{tests} = (M \times N) \times N \times \Phi$, donde M y N son el alto y ancho de la imagen respectivamente, y Φ la cantidad de ángulos considerados, donde se considera $\Phi = 40$, de discretizar en 0.5 grados los ángulos de los segmentos evaluados y restringir la inclinación máxima del *scratch* a un ángulo $\phi_{max} = 10$ grados.

Se agrega la restricción de que un *scratch* tiene un largo mínimo proporcional a la altura de la imagen M, y se fija este parámetro como $\frac{M}{10}$. Se descartan segmentos, aunque sean significativos, cuyo largo exceda dicho parámetro.

4. **Principio de maximalidad**

La detección anterior genera un gran número de resultados redundantes, dado que un segmento muy significativo debe contener en sí mismo otros segmentos significativos con mayor NFA. Por ello se utiliza la noción de "maximalidad" [223]. Un segmento es "significativo maximal" si no incluye, ni está incluido, en otro segmento más significativo (de menor NFA). Solo se admiten segmentos que cumplen dicha propiedad (ver Fig. 8.3(b)).

(a) (b) (c)

Figura 8.3: (a) Fotograma original "*Laurel and Hardy*"[214] (b) Detección segmentos significativos con el principio de maximalidad. Se observan en los resultados algunas falsas detecciones. (c) Detecciones de (b) luego de aplicar el principio de exclusión. Resultados obtenidos durante la implementación del método.

5. **Principio de exclusión**

Un *scratch* puede tener ancho de varios píxeles, y esto genera sobre-detecciones que se superponen (Fig. 8.3(b)). Se establece, como criterio para obtener mayor precisión en las detecciones, que un píxel puede pertenecer a lo sumo a un único *scratch*.

Si un píxel está contenido en más de un segmento, se conserva el más significativo. Dicho píxel se elimina del resto de los segmentos y se recalculan los

valores de NFA del resto de los segmentos, sin considerar el píxel eliminado. Si los nuevos candidatos no son significativos entonces se descartan. Un ejemplo del resultado se muestra en la Figura 8.3.

El proceso se lleva cabo guardando las coordenadas y los valores de NFA de cada segmentos en tablas, y ordenando dichas tablas según los valores de NFA. Aquellos segmentos con menor valor de NFA son los más significativos.

A su vez se puede agregar el criterio de excluir aquellos píxeles detectados que disten menos de un cierto umbral τ_X entre si. Si dos píxeles detectados distan estrictamente menos que τ_X se considera que pertenecen al mismo *scratch*.

8.2. Implementación del algoritmo de detección espacial de *scratches*

En la Figura 8.4 se muestra un diagrama de bloques con los distintos módulos que se implementaron.

El código de la herramienta de detección y restauración de *scratches* se implementó utilizando la librería OpenCV. Se integró dicho código en el estándar OFX para que pueda ejecutarse sobre Natron, donde el usuario accede a las herramientas interactivas y parámetros de ajuste de la detección.

Los Algoritmos 6, 7, 8 y 9 proveen pseudocódigos de los diferentes bloques realizados, presentados en la Figura 8.4.

La implementación del algoritmo de detección de *scratches* (que llama a todos los otros algoritmos) se presenta en el Algoritmo 5, donde se incluye la metodología *a contrario* para realizar las detecciones.

Para los diferentes parámetros se utiliza la notación definida en la sección anterior y se utilizaron todos los valores por defecto sugeridos. En particular se tiene como entradas el ancho de la mediana horizontal (o ancho del *scratch*) de $w = 5$ píxeles, $s_{med} = 3$, $s_{avg} = 20$, un largo mínimo para los *scratches* $l_{min} = \frac{M}{10}$ e inclinación máxima $\phi_{max} = 10$ grados. Para el principio de exclusión se toma $\tau_x = 3$. El umbral para la transformada de Hough se fija en Δ_{thresh}=90 y el umbral para el número esperado de falsas alarmas en $\varepsilon = 1$ [4].

[4]En la Sección 8.5 se discuten de forma detallada todos los parámetros del algoritmo.

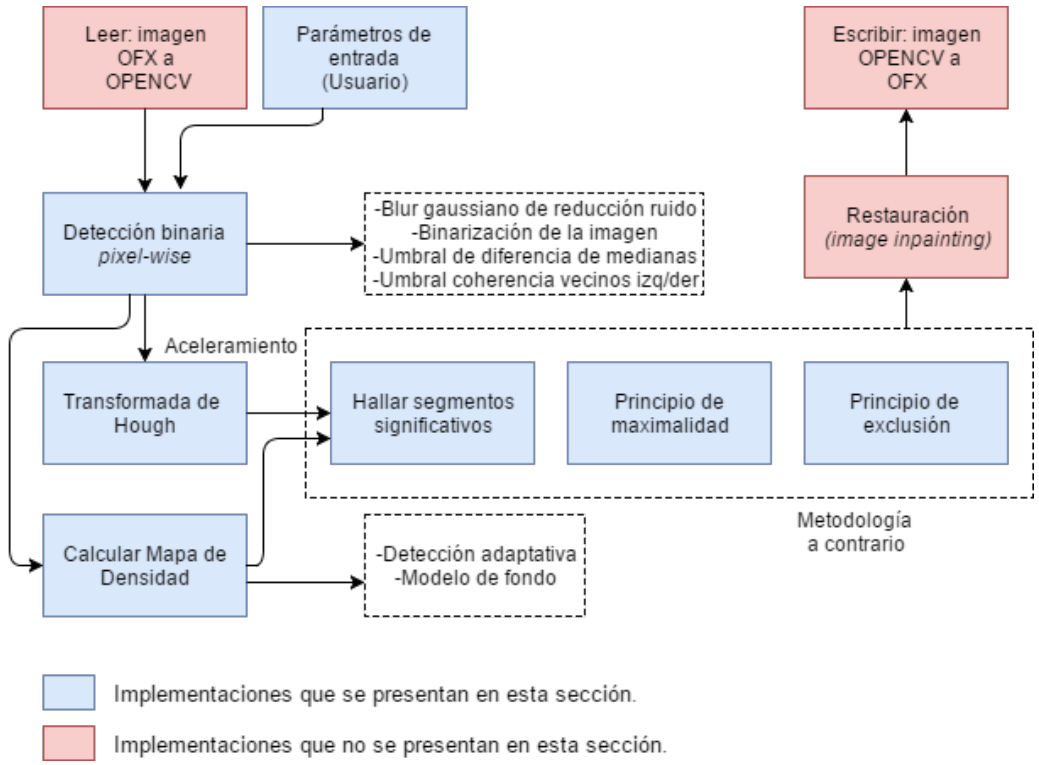


Figura 8.4: Diagrama de bloques con las distintas tareas que se implementaron. Los bloques en rojo se presentan en las Secciones 5.2.4 y 8.4 respectivamente.

Algoritmo 5: Pseudocódigo del algoritmo de detección de *scratches*.

Entrada: Secuencia de imágenes I_i,
parámetros: w, s_{avg}, s_{med}, Δ_{thresh}, ϕ_{max}, l_{min}, ε, d_{min}
para cada *imagen* I_i *de la secuencia* **hacer**
  Leer parámetros ;
  Hallar imagen I_B utilizando w, s_{avg} y s_{med}. (Algoritmo 6) ;
  Hallar imagen P_M, mapa de densidad de I_B. (Algoritmo 7);
  Calcular N_{tests}, utilizando ϕ_{max} ;
  Crear lista *LíneasHough*, con la transformada de Hough de I_B, utilizando Δ_{thresh};
  Crear e inicializar lista *SegmentosMaximales*;
  para cada *línea en LíneasHough* **hacer**
    Hallar todos los segmentos significativos utilizando P_M, ε y N_{tests}. (Se debe verificar que $NFA \leq \varepsilon$ y $largo \geq l_{min}$) ;
    Aplicar principio de maximalidad al conjunto de segmentos detectados y actualizar lista *SegmentosMaximales*. (Algoritmo 8);
  fin
  Aplicar principio de exclusión a la lista *SegmentosMaximales*. (Algoritmo 9);
fin
Salida: Lista de detecciones *SegmentosMaximales*

El Algoritmo 6 corresponde a la implementación del criterio de detección binaria *per-pixel*.

Algoritmo 6: Pseudocódigo de detección binaria *per-pixel*

Entrada: Imagen I, parámetros w, s_{med}, s_{avg}.
Crear imagen I_g, desenfoque gaussiano de I. (*kernel* 3×3);
Crear imagen I_m, con la mediana horizontal de w píxeles de I;
Crear imagen I_i, promedio a izquierda, de ancho $r = \lfloor \frac{w}{2} \rfloor + 1$ píxeles de I;
Crear imagen I_d, promedio a derecha, de ancho $r = \lfloor \frac{w}{2} \rfloor + 1$ píxeles de I;
Crear imagen I_B binaria, inicializar todos los píxeles en valor 0 ;
para cada *píxel (x,y) de* I **hacer**
  si $(|I_g(x,y) - I_m(x,y)| \geq s_{med})$ ***y*** $|I_d(x,y) - I_i(x,y)| \leq s_{avg}$ **entonces**
    $I_B(x,y) = 1$;
  fin
fin
Salida: Detección binaria I_B

A partir de la imagen I_B se calcula el mapa de densidad P_M que permite hacer la detección adaptativa a las zonas con mayor cantidad de textura de la imagen. Para ello se implementó el Algoritmo 7.

Algoritmo 7: Pseudocódigo de cálculo de mapa de densidad P_M

Entrada: Imagen I_B, generada con el Algoritmo 6, parámetro $L = \frac{N}{30}$.
Definir *DensidadMáxima*= 0;
Crear imagen P_M de ceros, de tipo *float*;
para cada *pixel (x,y) de* I_B **hacer**
  Considerar S_1, S_2, S_3 y S_4, 4 cuadrados $L \times L$, que se encuentran con respecto a (x, y) abajo y a izquierda, abajo y a derecha, arriba y a derecha, arriba y a izquierda, respectivamente. (ver figura 8.2);
  Calculo d_1,d_2,d_3 y d_4, densidad de píxeles con valor 1 en I_B, dentro de los cuadrados S_i ;
  Establecer *DensidadMáxima* como el máximo entre d_1,d_2,d_3,d_4;
  $P_M(x,y)$ =*DensidadMáxima* ;
fin
Salida: Mapa de densidad P_M

Para realizar la transformada de Hough se utilizó la función *HoughLines* de OpenCV[224]. De acuerdo con la documentación, se utilizó el método estándar. En este caso, la salida de la función es una matriz donde se representa cada línea por un vector (ρ, θ), donde ρ es la distancia entre el punto $(0, 0)$ y la línea, y θ es el angulo entre el eje x y la normal a la línea. Como parámetros de entradas de dicha función se utilizaron resoluciones para la distancia y los ángulos considerados por la transformada de Hough de $\rho_{res} = 1$ y $\theta_{res} = 0{,}5$ grados, como se presentó anteriormente.

Para analizar los segmentos significativos incluidos en las líneas arrojadas por la transformada de Hough se utilizó la función *LineIterator* de OpenCV[225]. Dicha función toma como entrada dos puntos p_1 y p_2 de una imagen, y devuelve el valor y las coordenadas de todos los píxeles contenidos en el segmento $[p_1, p_2]$. Esta implementación es eficiente y es adecuada para iterar en un gran número de segmentos de recta, como lo requiere la metodología *a contrario*.

Una vez que se dispone de una lista con los segmentos significativos contenidos en una línea, es necesario aplicar el principio de maximalidad, para eliminar detecciones redundantes y solo conservar los segmentos más significativos (de menor NFA). El Algoritmo 8 presenta el pseudocódigo del principio de maximalidad.

Algoritmo 8: Pseudocódigo del principio de maximalidad.

Entrada: Lista de segmentos significativos alineados.
La lista incluye candidatos con todos los intervalos J significativos en la línea, que empiezan con un punto no alineado y son precedidos por uno alineado, y terminan con un punto alineado seguido por uno no alineado ;
Considerar por turnos todos los pares (J, K) en la lista, donde J y K satisfacen $J \subset K$. Si K es estrictamente mas significativo que J, borro J de la lista;
Iterar hasta que no quedan pares (J, K);
Guardar segmentos en la lista *SegmentosMaximales*;
Salida: Lista *SegmentosMaximales* con nuevos segmentos detectados

El principio de exclusión se llevó a cabo utilizando el criterio de que un píxel pertenece a lo sumo a un único scratch, como se muestra en el Algoritmo 9. También se agregó el criterio de que los píxeles de dos detecciones diferentes deben distar como mínimo $\tau_x = 3$ píxeles.

Algoritmo 9: Pseudocódigo del principio de exclusión.

Entrada: Lista *SegmentosMaximales*, salida del Algoritmo 8. τ_x distancia mínima entre dos scratches.
Ordenar *SegmentosMaximales* según los segmentos más significativos;
para cada *segmento en SegmentosMaximales* **hacer**
  si *interseca con otro segmento de SegmentosMaximales* **entonces**
    Quitar de la lista el segmento menos significativo;
    Borrar el punto de intersección del segmento menos significativo;
    Determinar si los dos segmentos nuevos son significativos;
    si *alguno es significativo* **entonces**
      Agregar el nuevo segmento en la lista *SegmentosMaximales*;
    en otro caso
      Descartar el segmento no signifativo;
    fin
  fin
fin
Salida: Lista *SegmentosMaximales* modificada.

Se implementó también una ventana interactiva, que puede ajustar el usuario en caso de que los *scratches* que desea detectar estén localizados en una región específica de la imagen.

Para eliminar falsas detecciones, se propone en la literatura utilizar técnicas de filtrado temporal (ver Sección 3.4). Si bien estas propuestas eliminan algunas falsas detecciones, funcionan en situaciones específicas, y no son útiles para la generalidad de los casos. Por ejemplo, hacer una estimación global del movimiento de la escena para descartar los *scratches* que siguen dicho movimiento, es útil únicamente para planos en los que existe movimiento de cámara. Si la cámara está fija, como sucede generalmente, este método no funcionaría. Otros abordajes asumen que el *scratch* perdura por varios fotogramas en la secuencia, y esto tampoco es aplicable a una gran cantidad de planos. Debido a que estos algoritmos atacan casos específicos, no fueron agregados a la etapa de detección y se propone explorar esta posibilidad a futuro (ver Sección 10.2).

8.3. Optimización del algoritmo de detección de *scratches*

Los tiempos de ejecución implicados por el algoritmo de detección espacial de *scratches* son elevados y no permiten a un usuario detectar con fluidez los *scratches* incluso para videos de baja resolución. Por eso se realizaron tareas de optimización y depuración del código implementado, que permitieron acelerar en gran medida los tiempos de ejecución obtenidos inicialmente.

En el artículo de referencia[149] se muestra un promedio del tiempo de ejecución que toma procesar un fotograma para distintos videos de prueba, al aplicar el algoritmo de detección de *scratches*. Dichos tiempos de ejecución corresponden a una implementación en *Matlab* con funciones *cmex* para acelerar ciertas partes del algoritmo. Fueron ejecutados en un procesador Intel Core i5 CPU (2.67 GHz). Esto es útil como referencia del desempeño esperado del algoritmo.

En particular se utilizó para nuestra optimización el video de prueba "*Knight*" [214]. El video tiene dimensiones 257×257 y una duración de 64 fotogramas.

El tiempo de ejecución relevado por el artículo de referencia[149] para dicho video es de 0.49 segundos por fotograma. En este caso se utilizó para las pruebas un procesador Intel Core i7 CPU (2.2 GHz) y se obtuvo al ejecutar el código sobre Natron, un tiempo de ejecución promedio de 0.41 segundos por fotograma. Esto implica una tasa de reproducción lenta, de aproximadamente 2.4 fotogramas por segundo, resultado similar al del artículo de referencia.

Para realizar la optimización fue necesario independizar el bloque de código en OpenCV del bloque en OFX. Esto permitió controlar el flujo de ejecución del programa fuera de Natron, lo que facilita la depuración y permite evaluar los tiempos de ejecución insumidos por las diferentes tareas del algoritmo.

Para evaluar el manejo de la memoria y los tiempos de ejecución del algoritmo se utilizaron las herramientas Valgrind [201], Callgrind [202] y KCachegrind [203]. En la Tabla 8.1 se muestran los tiempos de ejecución relevados inicialmente en KCachegrind por el algoritmo implementado.

Tabla 8.1: Tiempos de ejecución iniciales relevados con *KCachegrind*. Se ejecuta el algoritmo de detección de *scratches* para el video de prueba "*Knight*"

Función	**Ciclos de Reloj**
(Total) Detección *scratches*	53.389.606.162
Mapa de densidad P_M	45.628.957.905
Detección binaria I_B	3.980.201.035
Principio de exclusión	2.079.370.716
Trasnformada de Hough	988.368.382
Principio de maximalidad	39.031.073

En el artículo de referencia[149] se propone que para mejorar el tiempo de ejecución se acelere la metodología *a contrario* de detección mediante las ideas propuestas en el artículo "*LSD: A fast line segment detector with a false detection control*" [226].

Sin embargo, como se muestra en la Tabla 8.1, en este caso el 85 % del tiempo

de ejecución total se usa para calcular el mapa de densidad P_M. Este cuello de botella del algoritmo no se acelera con el método LSD mencionado, ya que el modelo de fondo asumido no cambia, por lo que aún se requiere la imagen P_M (ver Sección 8.1) [5].

Este costo computacional elevado está asociado a que el Algoritmo 7 accede a los píxeles de un gran número de cuadrados $L \times L$ a lo largo de toda la imagen. Se estimó la complejidad de dicho algoritmo mediante la cantidad de operaciones implicadas al recorrer los píxeles en una imagen de tamaño $M \times N$. Para ello se considera que en los bordes de la imagen, el algoritmo también trabaja sobre cuatro cuadrados S_i de dimensión $L \times L$. Esto no es así, ya que cuando los cuadrados exceden los límites de la imagen, estos se recortan y se reduce el número de iteraciones. Con esta aproximación conservadora, se tiene un número de operaciones sobre la imagen de:

$$\#operaciones = M \times N \times 4 \times L^2 \tag{8.7}$$

La implementación presentada fue escrita de forma intuitiva, pero no aprovecha una enorme redundancia de información presente en los cálculos de las densidades de cada cuadrado S_i. En las operaciones no se aprovecha, por ejemplo, que las densidades de los cuadrados de un píxel difieren marginalmente de las densidades de los píxeles contiguos.

La estrategia para aprovechar dicha redundancia consiste en generar una nueva imagen auxiliar P_v, donde a cada píxel se le asigna la suma de L píxeles vecinos verticales en la imagen original I_B. De esta forma, para hallar la densidad dentro de un cuadrado S_i basta con recorrer L veces la imagen nueva P_v, en vez de recorrer los píxeles de todo el cuadrado en la imagen I_B, lo cual implicaba L^2 iteraciones. Esta estrategia no incluye la banda de ancho L en los bordes de la imagen. El pseudocódigo se muestra en el Algoritmo 10.

[5] El video "*Knight*" presenta pocos *scratches* por fotograma, como sucede en un gran número de secuencias. Con un número mucho mayor de *scratches* podría ser más costosa computacionalmente la metodología *a contrario*.

Algoritmo 10: Pseudocódigo del cálculo de mapa de densidad P_M, optimizado

Entrada: Imagen I_B $(M \times N)$, generada con el Algoritmo 6, parámetro $L = \frac{N}{30}$.

para cada *imagen binaria* I_B *de la secuencia* **hacer**

Definir *DensidadMáxima*= 0;

Crear imagen P_M, tipo *float*;

Crear imagen P_v, suma vertical de L píxeles de la imagen I_B, dividido por L ;

para cada *píxel (x,y) de* I_B *en la banda de ancho L en los bordes de* I_B **hacer**

Mismo procedimiento que en Algoritmo 7 ;

Establecer *DensidadMáxima* como el máximo entre d_1,d_2,d_3,d_4;

$P_M(x, y)$ =*DensidadMáxima* ;

fin

para cada *píxel (x,y) de* I_B *fuera de banda de ancho L* **hacer**

Hallar d_1, suma de L píxeles a izquierda de (x,y-L) en P_v;

Hallar d_2, suma de L píxeles a derecha de (x,y-L) en P_v;

Hallar d_3, suma de L píxeles a derecha de (x,y) en P_v;

Hallar d_4, suma de L píxeles a izquierda de (x,y) en P_v;

Establecer *DensidadMáxima* como el máximo entre d_1,d_2,d_3,d_4;

$P_M(x, y)$ =*DensidadMáxima* ;

fin

fin

Salida: Mapa de densidad P_M

El nuevo número de operaciones se puede estimar como:

$$\#operaciones_{opt} = (M - L) \times (N - L) \times 4 \times L + 2 \times L \times (N + M) \times L^2 \quad (8.8)$$

$L = \frac{N}{30}$ (definido anteriormente), y a grandes rasgos se puede aproximar que en los formatos de video $M \approx N$. Esto nos permite comparar de forma aproximada entre $\#operaciones$ y $\#operaciones_{opt}$. El número de operaciones antes de optimizar, presentado en la Ecuación (8.7), se puede expresar entonces como:

$$\Rightarrow \#operaciones \approx \frac{4}{30^2} \times M^4 \approx 4{,}44{,}10^{-3} \times M^4 \quad (8.9)$$

Del mismo modo el número de operaciones después de optimizar, presentado en la Ecuación (8.8), se puede expresar como:

$$\#operaciones_{opt} \approx (\frac{29}{30})^2 \times \frac{4}{30} \times M^3 + \frac{4}{30^3} \times M^4 \Rightarrow$$
$$\Rightarrow \#operaciones_{opt} \approx 0,125 \times M^3 + 4{,}9{,}10^{-6} \times M^4 \quad (8.10)$$

En la Figura 8.5 se grafican ambas funciones para valores de M entre 0 y 2048 (lo que corresponde a la resolución de un video *2K*).

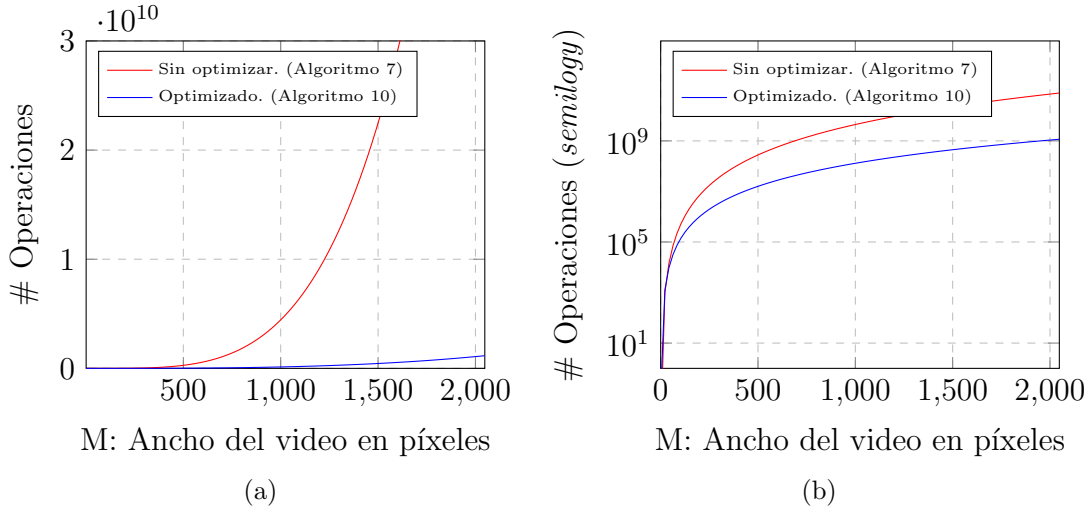


Figura 8.5: Número de operaciones estimados para el cálculo de la imagen P_M antes y después de optimizar. Respectivamente las Ecuaciones (8.9) y (8.10). *#operaciones* en escalas (a) lineal (b) semilogarítmica.

Como puede verse tanto en el resultado analítico como en la gráfica, la optimización logra bajar prácticamente un grado el orden de complejidad del algoritmo (de $\mathcal{O}(M^4)$ a $\mathcal{O}(M^3)$).

Mejoras en el desempeño

Para acelerar los tiempos de ejecución obtenidos, se utiliza el *flag* de optimización que proporciona GCC "*-O3*"[227]. Los resultados de esta optimización se muestran en la Tabla 8.2. Se implementó la optimización presentada en el Algoritmo 10 y los nuevos tiempos de ejecución obtenidos mejoran sensiblemente el desempeño, reduciendo el tiempo de ejecución total al 11 % del tiempo obtenido originalmente.

Función	**Original**	**-O3**	**-O3+Algoritmo 10**
(Total) Detección de *scratches*	55.373.385.557	15.308.300.334	5.967.197.680
Mapa de densidad P_M	45.628.957.905	11.916.290.506	2.552.487.916

Tabla 8.2: Tiempos de ejecución totales en ciclos de reloj, antes y después de realizar las optimizaciones.

También se relevaron los tiempos de ejecución por fotograma (en promedio) para distintos videos de prueba, ejecutando el algoritmo de detección espacial sobre Natron. Los resultados se muestran en la Tabla 8.2. La secuencia "*Sitdown*" tiene más cantidad de *scratches* y mayor resolución que "*Knigth*"[214]. También se utilizó un fragmento de la película "Juegos y Rondas" (1968, ICUR), que presenta varios *scratches* y tiene resolución Full HD.

Tiempo de ejecución (s)	"*Knight*"	"*Sitdown*"	"Juegos y Rondas"
Resolución	257×257	514×514	1920×1080
Original	0.41	55.32	64.53
Optimizado	0.028	0.58	0.51

Tabla 8.3: Tiempos de ejecución en segundos, luego de realizar las optimizaciones, para distintos videos de prueba.

Comparación de los tiempos de ejecución experimentales con la complejidad estimada

En el caso de las pruebas realizadas con el video "*Knight*", se tienen dimensiones $M = N = 257$ y duración de 64 fotogramas. Los tiempos de ejecución relevados por *KCachegrind* en la Tabla 8.2, se pueden expresar en ciclos de reloj por fotograma, dividiendo entre 64 los valores. Esto se muestra en la Tabla 8.4.

Ciclos de reloj	**Sin Optimizar**	**Optimizado**
Mapa de densidad P_M	$7{,}1 \times 10^8$	$3{,}9 \times 10^7$

Tabla 8.4: Tiempos de ejecución por fotograma relevados por *KCachegrind*, en ciclos de reloj, luego de optimizar el cálculo de la imagen P_M.

En la Figura 8.6 se muestra la estimación teórica de la complejidad para valores de M cercanos a 257. De las Ecuaciones (8.9) y (8.10) se obtienen los valores las estimaciones para M=257:

- $\#operaciones_{257x257} \approx 2 \times 10^7$
- $\#operaciones_{257x257.opt} \approx 2 \times 10^6$

Podemos decir que se verifica una reducción de un orden de magnitud en los tiempos de ejecución del algoritmo que calcula la imagen P_M, tanto para la estimación teórica como para los datos relevados experimentalmente.

8.4. Restauración de *scratches* con técnicas de *image inpainting*.

Luego de la instancia de detección se deben restaurar los *scratches* detectados. Para ello se propone utilizar tres técnicas de *image inpainting*, para las cuales existen implementaciones libres y eficientes, que se presentan en esta sección. [6]

En el *plugin* de detección y restauración de *scratches* de PARP, se incluye la opción de elegir entre dos algoritmos de *image inpainting* implementados en la función cvInpaint de OpenCV [228][229]. Dicha función utiliza los métodos propuestos por M. Bertalmío y colegas "*Navier-Stokes, Fluid Dynamics, and Image*

[6] Los tres métodos elegidos fueron presentados en la Sección 3.6.

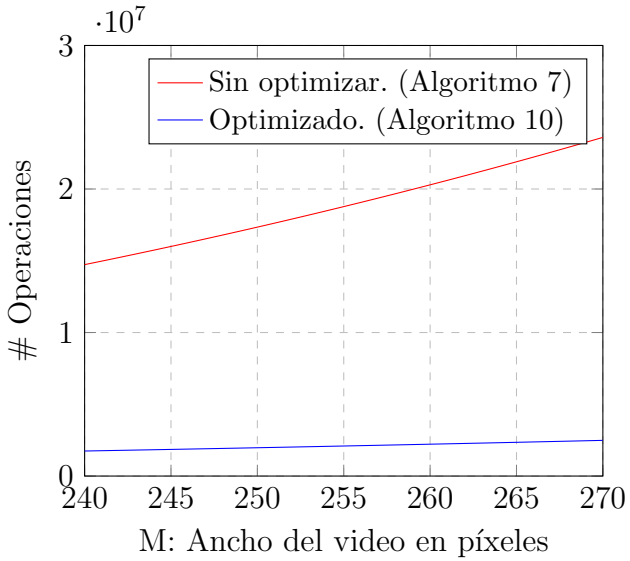


Figura 8.6: Número de operaciones antes y después de optimizar, para resolución de prueba 257x257. Sin Optimizar ec.8.9, Omptimizado ec.8.10.

and Video Inpainting"(2001) [158] y la propuesta de A. Telea "*An Image Inpainting Technique Based on the Fast Marching Method*" (2003) [163].

Recientemente Natron incluyó un *plugin* de *image inpainting* (ver Sección 5.1.2) cuya implementación se encuentra en versión *beta*. Utiliza el método propuesto por M. Daisy "*A Smarter Examplar-based Inpainting Algorithm using Local and Global Heuristics for more Geometric Coherence*" (2014). Este método presenta algunas ventajas respecto a los métodos que utiliza OpenCV.

La desventaja que presentan todas estas técnicas, como se discutió anteriormente, es que los algoritmos de *image inpainting* no toman en cuenta la información temporal del video para hacer el rellenado, y esto puede llegar a ocasionar artefactos en las secuencias.

8.5. Parámetros

Detección espacial de *scratches*

Se definen los siguientes parámetros de entrada, y sus valores de acuerdo con lo presentado en la sección anterior:

- I_i: fotogramas $M \times N$ de una secuencia de imágenes RGB.
- $w = 5$: ancho del *scratch*. Corresponde al ancho de la mediana horizontal y el ancho de los promedios laterales a ambos lados del *scratch* como $r = \lfloor \frac{w}{2} \rfloor + 1$=3 (ver Fig. 8.7). Para gran parte de los videos es adecuado $w = 5$ [147],

sin embargo para resoluciones cercanas a Full HD en general los *scratches* tienen anchos mayores, y se pueden dar casos en que los *scratches* sean más finos (ver Fig. 8.9) y se haga necesario ajustar este parámetro manualmente. Este valor puede ajustarlo el usuario en Natron entre $w = 2$ y $w = 10$.

- $s_{med} = 3$: umbral para la diferencia entre el nivel de gris del píxel y su mediana horizontal. Existen casos en que este valor para un *scratch* puede estar por debajo del umbral para ciertos fotogramas, y esto genera inestabilidades temporales en los resultados. Este valor puede ajustarlo el usuario en Natron para valores de s_{med} entre 1 y 6, si se desea corregir ciertas detecciones (ver Fig. 8.10).

- $s_{avg} = 20$: umbral para la diferencia entre los promedios laterales del *scratch*. Se fijó en 20 y es adecuado empíricamente para todas las pruebas realizadas.

- Δ_{thresh}=90: umbral permisivo para las detecciones de la transformada de Hough. Este valor puede ajustarlo el usuario en Natron para limitar la cantidad de detecciones, y recuperar falsos negativos. Es posible ajustarlo entre valores de 10 a 500.

- Resolución de la transformada de Hough: para la distancia ρ, se fijó resolución de $\rho_{res} = 1$ píxel y para la resolución de los ángulos se fijó $\phi_{res} = 0{,}5$ grados.

- ϕ_{max}=10: inclinación máxima (en grados) considerada para los *scratches* con respecto a la dirección vertical. Este parámetro puede ajustarlo el usuario en Natron para eliminar falsas detecciones con ángulos mayores al de los *scratches*. Se pueden utilizar valores de ϕ_{max} entre 1 y 20 grados.

- $L = \frac{N}{30}$: tamaño de los cuadrados para calcular el mapa de densidad P_M. Se demuestra experimentalmente que la performance del algoritmo no mejora al modificar este parámetro [147].

- $l_{min} = \frac{M}{10}$: largo mínimo considerado para los *scratches*. Se fija automáticamente como la décima parte de la altura de la imagen, pero puede ser ajustado por el usuario en Natron para obtener *scratches* con largos mínimos entre 2 y 500 píxeles.

- $\varepsilon = 1$: impone cota al número esperado de falsas alarmas bajo el modelo de fondo. Se demostró que el parámetro es estable en torno a $\varepsilon = 1$ [147]. Se fija en 1 pero puede ser ajustado por el usuario en Natron, para obtener mayor o menor número de detecciones, dependiendo si el interés es mejorar la precisión o el *recall* del algoritmo. Se modifica utilizando potencias de 10, con valores entre 10^{-20} y 10^{20}. Esto se debe a la naturaleza logarítmica del número de falsas alarmas definido.

- $N_{tests} = N^2 M\theta$: cantidad de segmentos analizados. Se define para calcular NFA. θ es la cantidad de ángulos considerados. Se consideró una discretización de los ángulos de $\phi_{res} = 0{,}5$. Por ejemplo, si la inclinación máxima

es ±10 grados, $\theta = \frac{20}{0,5} = 40$. Este parámetro se fija a partir de los otros parámetros.

- $\tau_x = 3$: distancia mínima entre las detecciones, utilizada por el principio de exclusión. El usuario puede ajustarlo en Natron para eliminar detecciones muy cercanas entre sí, para valores de τ_x entre 0 y 20.
- Grosor=1: permite al usuario ensanchar o afinar el grosor de los segmentos detectados. Es útil en casos en que el perfil horizontal del *scratch* es irregular (ver Fig. 8.12) y para extender la región donde se realiza la interpolación. Se pueden ajustar el grosor entre 1 y 5 píxeles.
- Ventana ajustable: [Opcional] por defecto se encuentra desactivado. Al activarlo el usuario puede restringir el área en el que se realizan las detecciones, directamente en el visor de Natron. Esto es útil para *scratches* localizados, o cuando existen regiones de la imagen propensas a generar falsos positivos (ver Fig. 8.12).

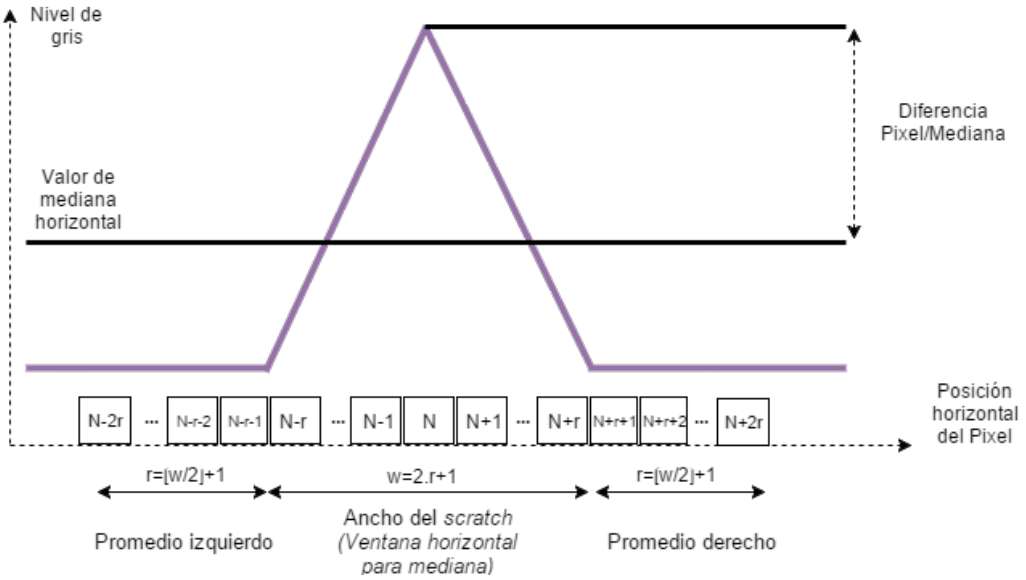


Figura 8.7: Perfil del *scratch*, dependiendo del parámetro r, radio del *scratch*.

Los parámetros a los que accede el usuario se fijan por defecto como se muestra en la Tabla 8.5, y corresponden a la detección automática de *scratches* [147]. En la misma tabla se muestran los rangos de variación habilitados para cada parámetro.

Tabla 8.5: Parámetros de entrada de usuario y sus valores por defecto para el algoritmo de detección automática de *scratches*. También existe el parámetro *booleano* de ventana ajustable, que por defecto está desactivado.

Parámetro	w	s_{med}	Δ_{thresh}	ϕ_{max}	l_{min}	ε	τ_x	Grosor
Valor por defecto	5	3	90	10	$\frac{M}{10}$	1	3	1
Rango de variación	[2..10]	[2..6]	[10..500]	[1..20]	[2..500]	$[10^{-20}..10^{20}]$	[0..20]	[1..10]

También existe el parámetro "Video de salida". A través de un menú de opciones el usuario puede seleccionar si desea visualizar (ver Fig. 8.8b):

- Las detecciones obtenidas superpuestas sobre la película original.
- Las detecciones obtenidas superpuestas sobre fondo negro (esta opción se utiliza para poder trabajar con el *plugin* de *inpainting* de Natron).
- la versión restaurada con *inpainting*.
- La película original.

Image inpainting

A continuación se resumen los parámetros de los métodos de restauración mediante *image inpainting*.

Dentro del *plugin* de PARP, se habilitaron todos los parámetros disponibles en la implementación de *image inpainting* existente en OpenCV [229]. Estos se resumen a continuación:

- Método: el usuario puede elegir entre dos métodos para realizar *image inpainting*: *Navier-Stokes* o *Fast Marching Squares*. Por defecto se elige usar *Navier-Stokes* porque provee resultados más precisos, sin embargo puede generar mayores incoherencias temporales en las secuencias.
- Radio de *inpainting*=4: determina los píxeles en una vecindad circular a cada punto que será restaurado. Al aumentar el radio se consideran más píxeles vecinos para realizar la interpolación. Es apropiado ajustarlo en relación a la resolución de video. Sin embargo esto depende también de la propia información de cada video, y en todas las pruebas fue apropiado utilizar el valor de 4 píxeles. El usuario puede ajustar este parámetro para valores entre 1 y 20 píxeles.

Por otro lado, el *plugin* de *image inpainting* disponible en Natron incluye los siguientes parámetros:

- Tamaño del *patch*: tamaño de los bloques cuadrados de la imagen que utiliza el algoritmo para realizar el rellenado mediante síntesis de textura. Por defecto se utilizan *patches* de siete píxeles.
- Tamaño de la ventana de búsqueda: ajusta la región donde se consideran estos *patches*. Si la ventana es mayor, mayor es la información que luego se utiliza para rellenar la región deseada. Al aumentarlo se obtienen en general mejores resultados para imágenes fijas pero más inestables temporalmente, y por lo tanto se generan artefactos en el video. Por defecto se realizan las búsquedas en una ventana cuadrada con lados de 16 píxeles.

- Tamaño de mezcla: permite ajustar la forma en que se combinan y superponen los diferentes *patches* dentro del rellenado. Por defecto este parámetro vale 1.20. Al aumentarlo el rellenado se suaviza y la imagen puede resultar desenfocada.

Al tratarse de una versión *beta* del algoritmo, se asume que en las siguientes versiones de Natron este *plugin* pueda sufrir mejoras y modificaciones en sus parámetros[7].

8.6. Evaluación

En primer lugar se relevan cuantificadores de desempeño del algoritmo de detección de *scratches*, aplicado a varios videos de prueba. Se utilizan los videos disponibles en el sitio *web* del artículo de referencia[214], donde a su vez proveen anotaciones manuales sobre la posición de los *scratches* en cada fotograma. Se utilizan los siguientes cuantificadores de desempeño:

- $Recall = 100\,\% \times \frac{\text{píxeles anotados detectados}}{\text{total píxeles anotados}}$.
  Representa el porcentaje de *scratches* detectados.
- $\text{Precisión} = 100\,\% \times \frac{\text{píxeles anotados detectados}}{\text{total de píxeles detectados}}$.
  Representa el porcentaje de acierto de los *scratches* detectados.
- $\text{F1-}Score = 100\,\% \times 2 \times \frac{recall \times \text{precisión}}{recall + \text{precisión}}$.
  Se combinan ambos criterios a través de su media armónica.
- Tiempo de ejecución: se releva el tiempo de ejecución por fotograma, en promedio, y se expresa en segundos. Se elige esta medida porque es independiente de las tasas de reproducción propias de cada video.

Los resultados se muestran en la Tabla 8.6. Se obtuvieron puntuaciones inferiores al artículo de referencia, por lo que los resultados son mejorables. Sin embargo, los tiempos de ejecución mejoran sensiblemente, siendo estos más de 20 veces más rápidos que los obtenidos en el artículo de referencia, en virtud de las optimizaciones realizadas.

Puntuaciones (%)	"*Knight*"	"*Sitdown*"	"*Star*"	"*Laurel and Hardy*"	"*Afgrunden1*"	"*Afgrunden2*"	"*Gate*"
Recall	71.32	69.32	73.07	54.20	75.66	86.42	70.76
Precisión	65.40	67.03	48.50	36.30	60.52	24.61	2.09
F1-Score	68.23	68.15	58.30	43.48	67.25	38.31	4.06
Tiempo de ejecución (s)	0.028	0.58	0.98	0.46	0.18	0.58	0.68

Tabla 8.6: Cuantificadores de desempeño para el algoritmo de detección de *scratches*, para varios videos de prueba, utilizando anotaciones manuales de los *scratches*[214].

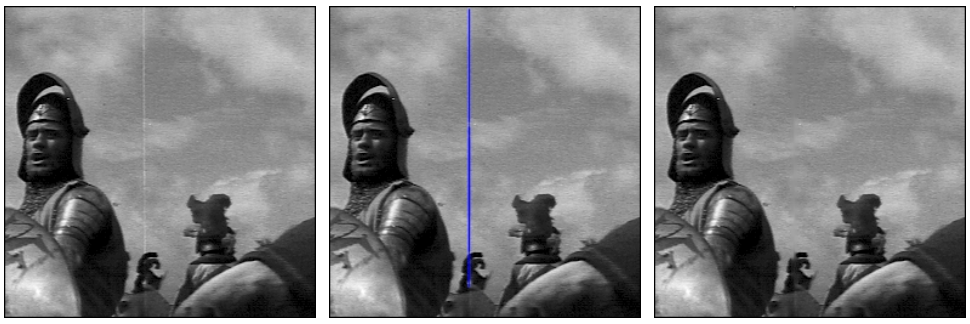
[7] En la siguiente sección se presentan algunos primeros resultados obtenidos al utilizar el *plugin* de *inpainting* de Natron.

Como se muestra en la Tabla 8.6, el *recall* no supera el 80 % en la mayoría de los casos. Lo mismo sucede con los resultados en [147]. Esto significa que no se detecta un porcentaje considerable de los *scratches* utilizando los parámetros por defecto del algoritmo. Por eso se propone habilitar al usuario parámetros para que pueda corregir manualmente las detecciones. [8]

A continuación se presentan y discuten algunos resultados y ajustes de parámetros usuales, tanto para el algoritmo de detección de *scratches* como para los algoritmos de restauración mediante *image inpainting*.

En la Figura 8.8 se muestra un primer resultado cualitativo del algoritmo de detección y restauración de *scratches* para un fotograma particular de la secuencia de prueba "*Knight*", con resolución 257×257. Se utilizaron todos los parámetros por defecto, presentados en la Tabla 8.5. Se detectó de forma precisa el *scratch* presente en la imagen, sin embargo una pequeña porción del mismo en la parte inferior del fotograma no fue detectado, debido a la presencia de textura en la imagen y al hecho de que el nivel de gris del *scratch* es similar al nivel del gris del fondo en dicha región.

El algoritmo de *image inpainting* incluido en el *plugin* de PARP, obtiene un resultado aceptable con sus valores por defecto, dado que la interpolación de los píxeles es indetecable para este fotograma en particular. Sin embargo, como ya fue comentado, durante la reproducción del video se observan incoherencias temporales que hacen que el resultado no sea indetectable, a pesar de que se logró eliminar el *scratch*.

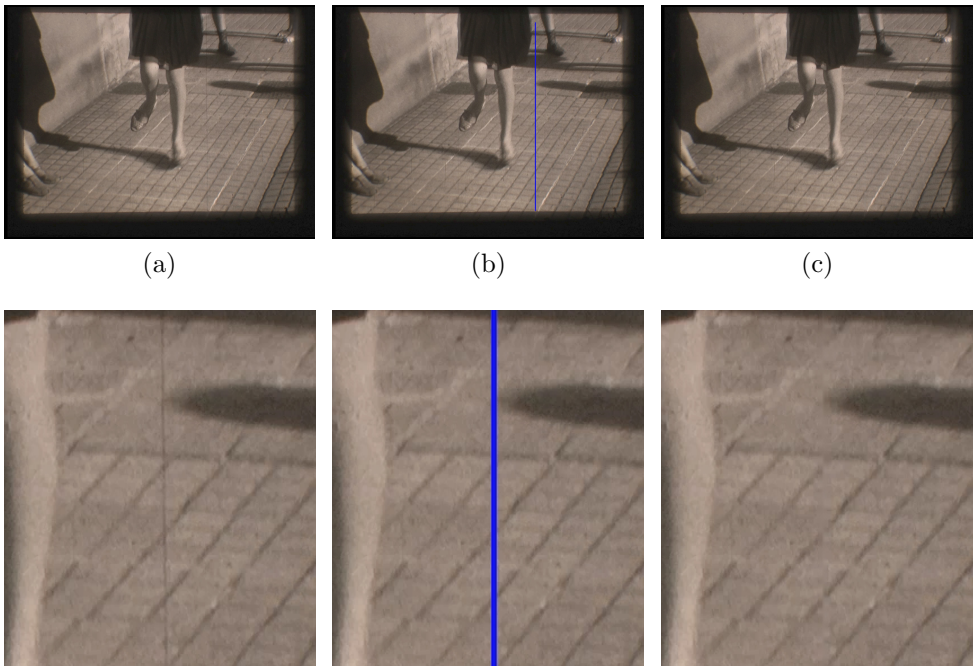


(a) (b) (c)

Figura 8.8: Fotograma de la secuencia de prueba "*Knight*" [214] (a) Original, presenta un único *scratch* (b) Detección espacial de *scratches* en azul con los parámetros por defecto (c) Restauración del *scratch* interpolando el nivel de gris de los píxeles con técnica de *image inpainting* por defecto.

[8] Estos parámetros se presentaron en la Sección 8.5.

En la Figura 8.9 se muestra una prueba en un fotograma en resolución Full HD (1920x1080) de la película "Juegos y Rondas" (1968, ICUR). Se obtiene en promedio, un tiempo de ejecución de 0.028 segundos por fotograma, lo cual es aceptable y permite realizar ajustes de parámetros de forma fluida, aunque no se alcance la reproducción en tiempo real, correspondiente a una tasa de 25 fotogramas por segundo. En este caso se utilizaron todos los parámetros por defecto. Cabe destacar que los *scratches* presentes en esta película son finos y en algunos casos dejan de ser perceptibles visualmente. Por esto se dejó fijo el parámetro $w = 5$ que determina el ancho del *scratch*, sin embargo en otras secuencias HD puede ser necesario aumentar este parámetro. [9]



(a) (b) (c)

(d) (e) (f)

Figura 8.9: Fotograma de la película "Juegos y Rondas" (1968, ICUR) (a) Original (b) Detección espacial de *scratches* (c) Restauración de los *scratches* con técnica de *image inpainting* con valores por defecto. (d),(e),(f) *Zoom* en el *scratch*

En la Figura 8.10 se presenta una nueva prueba donde disminuir el umbral s_{med} permite mejorar las detecciones. Se utilizaron todos los parámetros por defecto del algoritmo en la Figura 8.10 (b) incluido $s_{med} = 3$, y en la Figura 8.10(c) se ajustó $s_{med} = 2$. En este video de prueba, las detecciones del *scratch* presentan inestabilidades temporales debido a que la mayor parte del *scratch* se encuentra sobre una región de fondo, que además de poseer textura propia, tiene un nivel de gris similar al del *scratch*, como sucedía en la Figura 8.8. Reducir el umbral

[9] Se sugiere al lector observar las imágenes en máxima resolución para poder analizar cualitativamente los resultados.

de la diferencia de nivel de gris del píxel con su mediana horizontal s_{med}, es el único ajuste que permite detectar este tipo de *scratch*, ya que con este ajuste la etapa de detección binaria es más permisiva y se consigue que la línea vertical sea más significativa en conjunto. Este ajuste de parámetros no es adecuado para la totalidad de los fotogramas de la secuencia, y para resolver esto puede utilizarse la creación de fotogramas clave en Natron, que permite ajustar diferentes valores para los distintos fotogramas (ver Manual de Usuario).

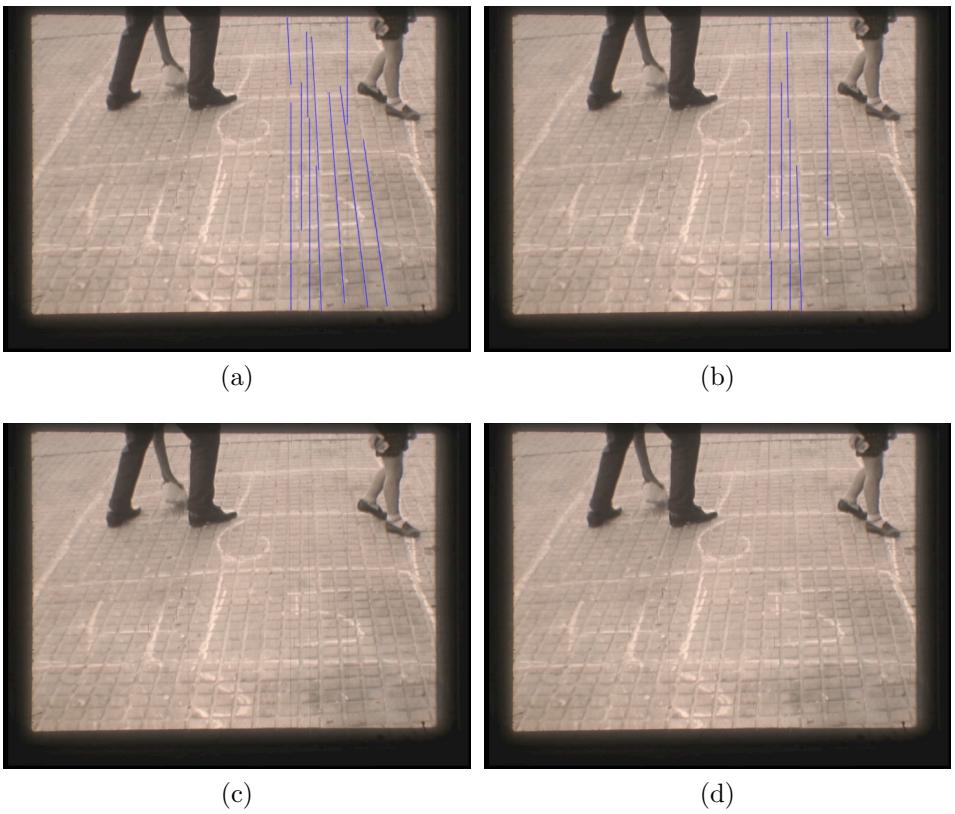
(a) (b) (c)

Figura 8.10: Fotograma de la secuencia "*Laurel and Hardy*" [214] (a) Original (b) Detección con los parámetros por defecto, en particular $s_{med} = 3$ (c) Ídem que (a), pero se ajusta $s_{med} = 2$ y se observa una mejora en la detección

El algoritmo de detección de *scratches* detecta, a grandes rasgos, estructuras finas y verticales presentes en la imagen. No es capaz de distinguir si dichas estructuras son elementos presentes en la escena o si son verdaderos *scratches*. En los siguientes ejemplos se muestra cómo estas situaciones pueden ocasionar falsas detecciones y se proponen ajustes de parámetros sencillos que permiten corregir el resultado automático del algoritmo.

En la Figura 8.11 se muestra un ejemplo donde la textura propia de la imagen genera falsos positivos. En este caso el problema se puede solucionar fácilmente restringiendo el parámetro ϕ_{max} que establece el ángulo máximo de los *scratches* detectados.

Otro ejemplo donde se generan falsos positivos se muestra en la Figura 8.12(b). Nuevamente, la textura propia de una región de la imagen es detectada como varios *scratches* por el algoritmo. En este caso, como estos falsos positivos se encuentran alejados del *scratch* que se desea localizar, es posible utilizar la ventana ajustable del *plugin* para restringir las detecciones a la zona específica del *scratch*. De esta forma se eliminan todos los falsos positivos.



(a) (b)

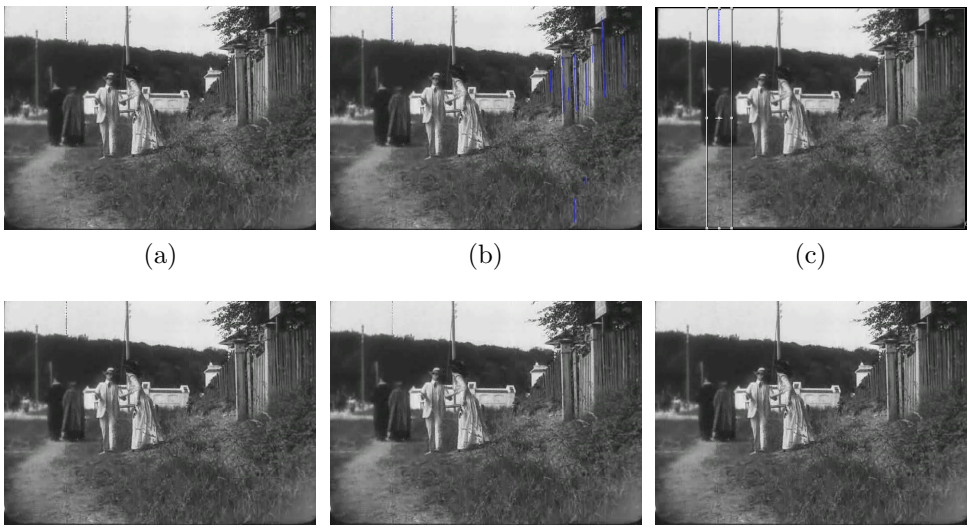
(c) (d)

Figura 8.11: Fotograma de la película "Juegos y Rondas", (1968, ICUR) (a) Detecciones con los parámetros por defecto, incluyendo $\phi_{max} = 10$. Se generan falsas detecciones en la textura propia del suelo presente en la escena (b) Detección restringiendo el parámetro a $\phi_{max} = 4$ grados (c) Fotograma original con *scratches* (d) Restauración con *inpainting* de PARP, utilizando las detecciones en (b). Se logran restaurar gran parte de los *scratches* presentes en la escena.

Resultados experimentales obtenidos con el *plugin* de *image inpainting* de Natron

Para utilizar el algoritmo de *image inpainting* disponible en Natron, se debe ajustar el parámetro Video de salida en el *plugin* de detección de *scratches*, para que las detecciones se muestren sobre un fondo negro. De esta forma el algoritmo de Inpaint de Natron utiliza la detección como máscara para realizar el *inpainting* (ver Fig. 8.13). [10]

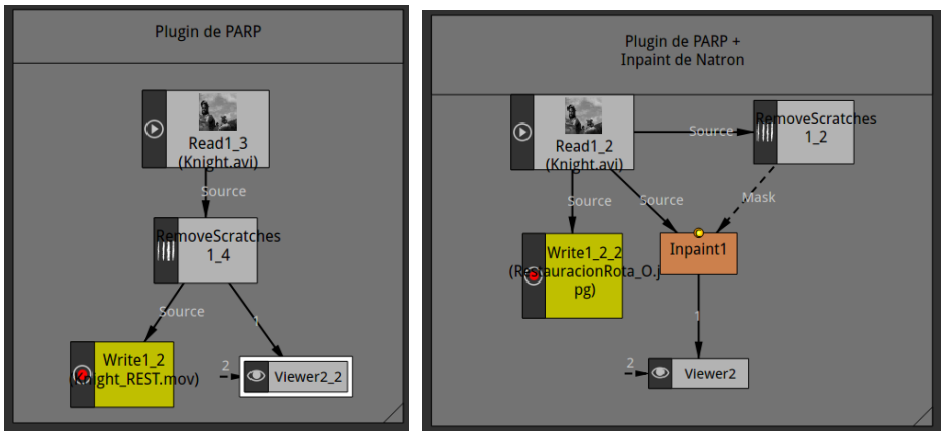
[10]Se incluyen instrucciones más detalladas sobre como utilizar todas las herramientas de *inpainting* en el Manual de Usuario de PARP, disponible en el Anexo B.



(a) (b) (c)

(d) (e) (f)

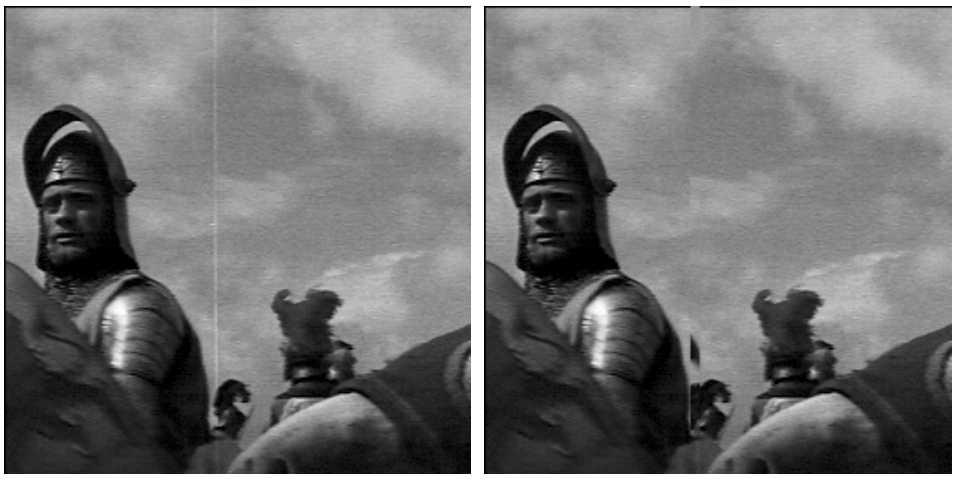
Figura 8.12: Fotograma de la secuencia de prueba "*Afgrunden Gate*" [214] (a) y (d): Fotograma original (b) Detecciones obtenidas con los parámetros por defecto. Se observan falsos positivos en la derecha del fotograma. (c) Se activa la ventana ajustable del *plugin* y se restringe a la región del *scratch*. (e) Restauración del *scratch* detectado en (c), con el parámetro Grosor=1 en su valor por defecto. El resultado no es indetectable. (f) Restauración indetectable del *scratch* señalado en (c), ajustando el parámetro Grosor=2.



(a) (b)

Figura 8.13: Diagramas de nodos en la interfaz de Natron para aplicar distintas técnicas de *inpainting* disponibles (a) Configuración utilizando *image inpainting* del *plugin* de restauración de *scratches* de PARP. (b) Configuración utilizando el *plugin* de *image inpainting* de Natron para restaurar las detecciones.

Los resultados obtenidos al utilizar esta técnica generan inconsistencias temporales y en general ocasionan nuevos artefactos, y se considera a partir de las pruebas realizadas y varios ajustes de parámetros, que esta técnica no es apropiada para restaurar *scratches* (ver Figura 8.14).



(a) (b)

Figura 8.14: (a) Fotograma original de la secuencia "*Knight*" (b) Restauración mediante el *plugin* de *image inpainting* de Natron. El resultado presenta inconsistencias temporales y artefactos.

Sin embargo, cabe mencionar que la técnica que utiliza el *plugin* de Natron es apropiada para restaurar otro tipo de deterioros (ver Sección 5.1.2).

Capítulo 9

Análisis Experimental y Discusión

En este capítulo se presentan algunos resultados generales del proyecto, así como también se discuten las posibilidades y limitaciones de la plataforma de restauración de películas desarrollada. Se analiza cualitativamente un ejemplo de restauración de una película corta, se evalúa el desempeño general obtenido y se presentan los tiempos de ejecución para varios ejemplos.

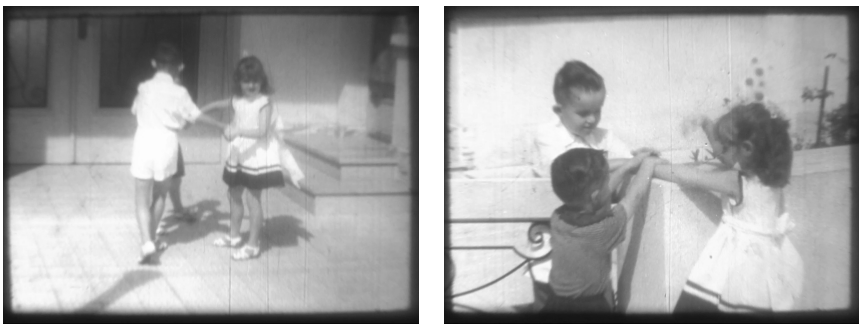
Los resultados específicos obtenidos con cada algoritmo implementado ya fueron presentados (ver Secciones 6.3, 8.6 y 7.4).

9.1. Desempeño en la restauración de una secuencia

Con el fin de ilustrar el desempeño de la aplicación, se analizan cualitativamente los resultados obtenidos al restaurar un fragmento de la película "Oxiurosis", realizada por el ICUR, y proporcionada por el AGU. Dicha película digital tiene resolución Full HD y compresión H264, de calidad media.

Se eligió un fragmento que presenta simultáneamente los deterioros que la plataforma es capaz de restaurar: *flicker* de luminancia y *scratches*. La secuencia cuenta con 647 fotogramas (correspondientes a 26 segundos de película, con una tasa de reproducción de 25 fotogramas por segundo). Existen dos cortes duros, el primero de ellos es un corte entre planos y el otro corresponde a un corte sobre el mismo plano, pero registrado en otro momento (*jump-cut*).

Además de los deterioros mencionados la imagen presenta borrosidad, vibración, manchas, suciedades, regiones sobre-expuestas con pérdidas de información, el marco del fotograma está desenfocado y es irregular, además de los artefactos ocasionados por la digitalización y la compresión del video (ver Fig. 9.1).



(a) (b)

Figura 9.1: Fotogramas de la película "Oxiurosis" realizada por el ICUR y proporcionada por el AGU. Se muestran algunos de los deterioros que presenta la película.

En primer lugar se carga la película en Natron y se aplica la herramienta de detección de cortes. El algoritmo analiza la secuencia completa en aproximadamente un minuto. Observando los valores que toma el parámetro SDA para algunos fotogramas, es sencillo establecer un umbral para fragmentar la película en los dos planos. En este caso se fijó manualmente el umbral en un valor de 0.6, y se segmentó la película correctamente, como se muestra en la Figura 9.2. Se tomó la decisión de no incluir el *jump-cut* en la separación de planos. A partir de este punto, es posible aplicar efectos a cada plano por separado [1].

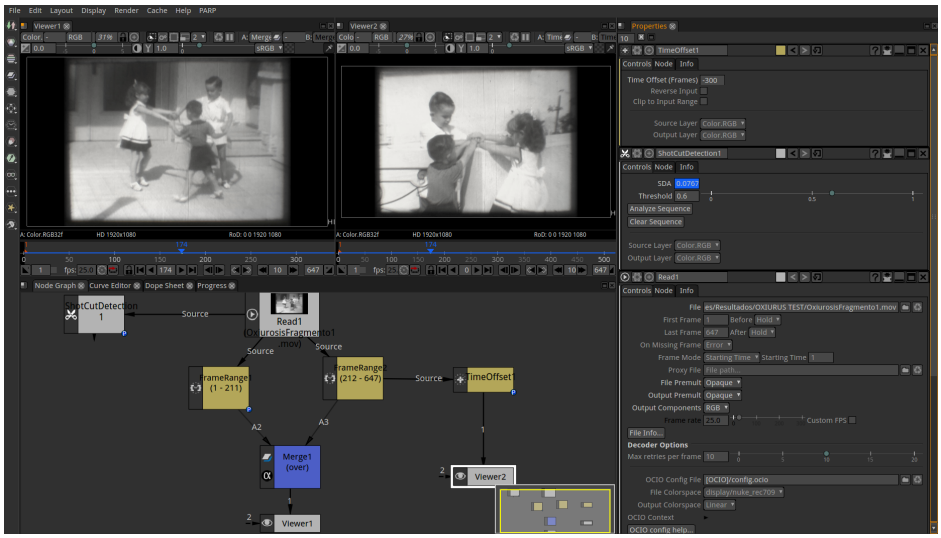


Figura 9.2: Proyecto de Natron donde se carga la película "Oxiurosis" y se la separa en dos planos mediante la herramienta de detección de cortes. En el visor de la interfaz de Natron se monitorean ambos planos simultáneamente.

[1] Por más detalles sobre la interfaz de Natron y cómo utilizar las distintas funcionalidades de la plataforma consultar el Manual de Usuario en el Anexo B.

A continuación se aplica la herramienta de corrección de *flicker* de luminancia, de acuerdo al flujo de trabajo de los algoritmos propuesto en este proyecto (ver Sección 4.3). Se muestra en la Figura 9.3 que la película presenta fluctuaciones violentas de la luminancia global durante un *jump-cut*. Además la secuencia posee *flicker* de luminancia que varía irregularmente en el tiempo, afectando con más fuerza a algunos grupos de fotogramas que a otros, de forma aleatoria. En particular, uno de los problemas más importantes que ocasiona este defecto es que algunos fotogramas de la película resultan sobre-expuestos. En este caso se ajustó la ventana temporal a nueve fotogramas. En la Figura 9.3 se muestra como el algoritmo es capaz de corregir en gran medida los desperfectos ocasionados por el *flicker*, aún cuando dicho desperfecto sobre-expone la imagen [2]. Los tiempos de ejecución al aplicar el efecto se redujeron a una tasa de siete fotogramas por segundo, lo que permite al usuario continuar trabajando de forma fluida en la secuencia.

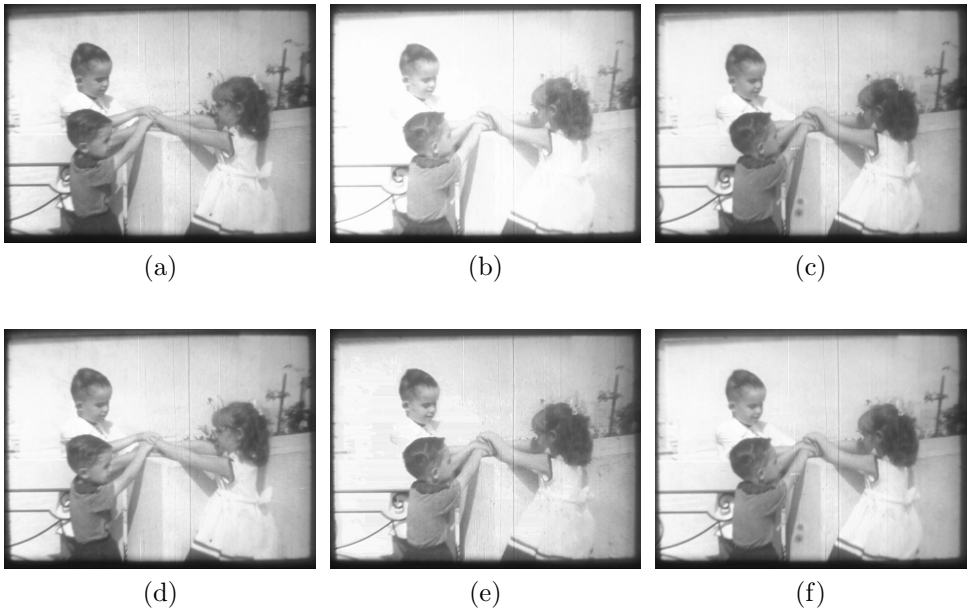
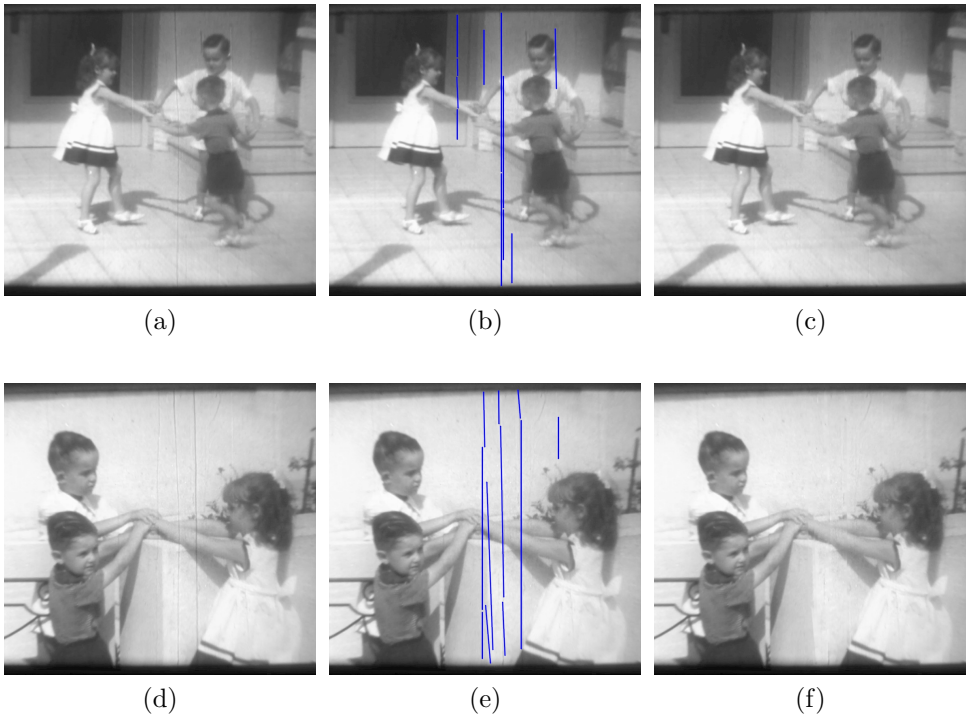


Figura 9.3: Fotogramas consecutivos de la película "Oxiurosis" (ICUR). Existe un corte duro *(jump-cut)* en la película entre las imágenes (a) y (b). Arriba, fotogramas originales que presentan *flicker* de luminancia con sobre-exposición de la imagen. Abajo, los mismos fotogramas restaurados utilizando el algoritmo de *deflicker* con ventana temporal de nueve fotogramas.

Luego se aplica a cada plano la herramienta de detección y restauración de *scratches*. Como puede verse en la Figura 9.4, la película presenta varios *scratches*

[2] En particular en el fotograma de la Figura 9.1, la sobre-exposición de la imagen durante el *jump-cut* proviene del diafragma de la cámara que hizo el registro. No se trata propiamente un deterioro del formato. En este sentido el restaurador puede tomar la decisión de no corregirlo. Sin embargo, el ejemplo es útil para ilustrar el desempeño del algoritmo.

diferentes entre sí, algunos son claros, otros oscuros y tienen distinto ancho y extensión a lo largo del cuadro. Sin embargo un ajuste sencillo de los parámetros del algoritmo permite detectar la mayoría de los *scratches* presentes en la escena. En particular se ajustaron los parámetros $s_{med} = 2$, $\phi_{max} = 4$ y se aumentó el ancho del *scratch* a siete píxeles. Para lograr restaurar los *scratches* detectados mediante *inpainting*, se aumentó el grosor de las detecciones a cuatro píxeles.



(a) (b) (c)

(d) (e) (f)

Figura 9.4: Fotogramas de la película "Oxiurosis". (a) y (d) Fotogramas originales. (b) y (e) Detecciónes obtenidas con el algoritmo de detección de *scratches*, con el ajuste de parámetros: $s_{med} = 2$, $\phi_{max} = 4$ y $w = 7$.

Figura 9.5: Proyecto de Natron para restaurar un fragmento de 26 segundos de la película "Oxiurosis", utilizando los tres algoritmos implementados. Los diferentes nodos separan de la película en planos y corrigen el *flicker* y los *scratches*. A su vez se crean los nodos de visualización y exportación. En el visor de la interfaz de Natron, a la izquierda se monitorea el fotograma original de la película y a la derecha el fotograma restaurado.

9.2. Tiempos de ejecución

Natron posee una herramienta para relevar información estadística sobre los tiempos de ejecución insumidos por cada efecto al procesar una película; dividiendo este valor por la cantidad de fotogramas se obtiene un promedio del tiempo de ejecución por fotograma insumido por los algoritmos. Cabe mencionar que el algoritmo de detección de cortes se ejecuta solo una vez antes de comenzar a restaurar la película, y por ello el tiempo de ejecución no es tan crítico como en los otros dos casos.

En la Tabla 9.1 se presentan algunos tiempos de ejecución obtenidos para distintas secuencias de prueba, y se observa su evolución a medida que se aumenta la resolución de video. Todos los algoritmos se utilizaron con sus parámetros en sus valores por defecto.

Secuencia	Knight	Sitdown	Juegos y Rondas	Oxiurosis	Nueva digitalización del AGU
Resolución (píxeles)	257×257	514×514	Full HD	Full HD	4K
Tiempo de ejecución por fotograma (s)					
Detección de cortes	$3{,}13 \times 10^{-3}$	0.04	0.09	0.09	0.22
Corrección de scratches	0.028	0.58	0.51	0.66	0.82
Deflicker	0.024	0.071	0.12	0.14	0.53

Tabla 9.1: Tiempos de ejecución por segundo, en promedio, para distintos video de prueba.

A grandes rasgos se puede observar que los tiempos de ejecución en resolución Full HD son aceptables, pero están lejos de la tasa de reproducción en tiempo real, de 25 fotogramas por segundo.

9.3. Limitaciones

En esta sección se analizan las limitaciones que presenta la plataforma implementada, tanto a nivel global como en lo específico de cada algoritmo y su *plugin* correspondiente.

En primer lugar la cantidad de problemas que se pueden atacar mediante PARP es limitada, más aún en contraste con la enorme variedad de problemas específicos y diferentes fuentes de degradación de una película, tanto en su formato analógico como digital, y en las transferencias de un formato al otro.

Si bien en este proyecto se implementaron únicamente dos técnicas de restauración de películas, Natron ofrece algunos *plugins* capaces de realizar otras tareas de restauración (eliminación de ruido, *image inpainting*, estabilización de imagen, etc. Ver Sección 5.1.1), y en este sentido se pueden reducir las limitantes que surjan, explorando nuevos caminos y utilizando los nodos pre-existentes de forma creativa.

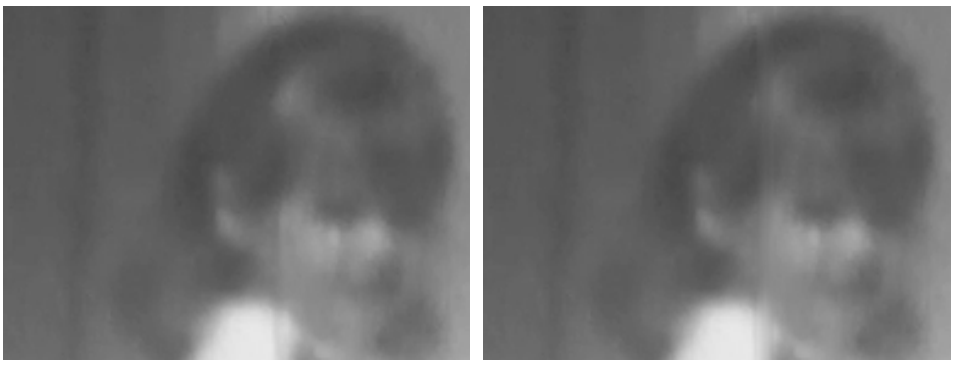
Aún faltan desarrollar varias herramientas para que sea posible restaurar de forma profesional una película completa. Existen restricciones tanto para el flujo de trabajo como para los tiempos de ejecución implicados al trabajar con archivos de película de larga duración, con resolución Full HD o mayor, y baja compresión de datos.

Natron es un software de composición y postproducción de video, y no de edición no lineal (NLE) (ver Sección 2.7.2), por ende carece de herramientas para las tareas específicas de edición de video. En este sentido al trabajar con secuencias extensas, por ejemplo de cinco minutos de duración, se generan una enorme cantidad de nodos y los proyectos se vuelven difíciles de gestionar. En estos casos existe la posibilidad de agrupar los diferentes planos y trabajarlos desde distintos proyectos de Natron, para conservar la organización y jerarquía del proyecto[3]. A su vez, la restauración de películas en muchas ocasiones no requiere realizar tareas de edición, dado que ya se encuentra editada y es posible aplicar varios algoritmos de restauración sin necesidad de cortar la película en planos. Sin embargo, se propone la posibilidad a futuro de integrar la plataforma PARP a aplicaciones de edición no lineal, a través del estándar OFX (ver Sección 10.2).

Los algoritmos de restauración pueden generar algunos artefactos digitales en la imagen. En el capítulo de Conlusiones y Trabajo Futuro se presentan soluciones a varios de estos problemas. A su vez, los algoritmos de detección de cortes y de *scratches* implican una supervisión considerable del usuario para obtener los mejores resultados.

La detección de *scratches* es inestable temporalmente, y en ocasiones se pueden generar un gran número de falsos positivos. Se atribuye al usuario una libertad

[3]Ver el Manual de Usuario, en el Anexo B



(a) (b)

Figura 9.6: Fotogramas de la película "Oxiurosis". (a) *Zoom* sobre un fotograma original. (b) Se rellenan las detecciones obtenidas con el algoritmo de detección de *scratches*. Si bien se detecta el *scratch* correctamente, se generan nuevos artefactos en la imagen al realizar la interpolación. Se aumentó de forma excesiva el parámetro Grosor hasta un valor de seis píxeles, para ilustrar este defecto.

de decisión grande, dado que el algoritmo posee varios parámetros, y esto puede resultar desventajoso para la toma de decisiones.

A su vez, la interpolación de píxeles mediante técnicas de *image inpaining* presenta inconsistencias temporales (ver Sección 8.6), y en ocasiones incluso para fotogramas fijos los resultados son detectables e inadecuados (ver Figura 9.6). Es un tema importante que se propone abordar como trabajo a futuro.

El algoritmo de *deflicker* de luminancia en ocasiones puede generar artefactos debido a errores de cuantización (ver la Sección 7.4).

Los tiempos de ejecución pueden resultar lentos en resolución Full HD si se aplican varios efectos encadenados, como se muestra en la Tabla 9.1.

Actualmente PARP se instala únicamente sobre el sistema operativo Linux. Es parte del trabajo a futuro extenderlo a otros sistemas operativos.

Capítulo 10

Conclusiones y Trabajo Futuro

10.1. Conclusiones

La realización de PARP permite afrontar problemas específicos de restauración digital en películas deterioradas digitalizadas, en una plataforma basada en software libre, con herramientas semi-automáticas que facilitan el proceso de trabajo de los restauradores. De esta forma se logra cumplir el objetivo inicial y principal de este proyecto.

Los usuarios pueden utilizar las herramientas de restauración de forma interactiva a través de la interfaz amigable de Natron. Se lograron desarrollar herramientas de forma modular, escalable y sustentable, para que a futuro puedan agregarse nuevos módulos, mejorar los ya existentes, así como modificar la interfaz de trabajo.

La investigación sobre el contexto y las diferentes tecnologías de restauración disponibles permite conocer que las opciones existentes en el mercado tienen costos elevados y en muchos casos el volumen de los acervos audiovisuales no amerita la inversión implicada, a pesar de su valor histórico. Por otro lado no existen en el mercado herramientas profesionales de software libre para restaurar video.

El área de restauración de películas deterioradas es realmente vasta. Las problemáticas que surgen son específicas de los diferentes formatos de captura, reproducción y archivado, y de sus condiciones de conservación. Las digitalizaciones también pueden ser fuente de diversos tipos de artefactos. Los algoritmos de restauración digital que afrontan estos desafíos se encuentran en activo desarrollo, y no existen soluciones completas, automáticas y eficientes para muchos de los problemas, como sería deseable.

Tempranamente el alcance de este proyecto implicaba la elección e implementación de cuatro algoritmos de restauración, en base a herramientas libres, preexistentes y disponibles en el mercado. Este objetivo fue modificado al conocer las complejidades específicas de las tecnologías de video a integrar, y la escasez de im-

plementaciones libres disponibles de los algoritmos. Finalmente se implementaron tres herramientas, que son claves en la cadena de trabajo de restauración digital de material fílmico: detección de cortes entre planos de una película, corrección de *flicker* de luminancia, detección y restauración de *scratches*. Estas herramientas alcanzan resultados dentro del estado del arte, pero a su vez son mejorables y algunos aspectos se encuentran en desarrollo, como la técnica de *inpainting* utilizada para restaurar los *scratches*.

Se evaluó durante el desarrollo del proyecto que no existen herramientas completamente automáticas para restaurar los distintos deterioros comunes en películas deterioradas, aunque este es un objetivo presente en el estado del arte del área. De todas formas es muy importante conservar la libertad de decisión del usuario en todo momento durante el proceso de restauración, además del hecho de que los algoritmos actuales no alcanzan su mejor desempeño si no es con la supervisión de un usuario.

La elección de la plataforma de post-producción de video profesional libre Natron proporciona ventajas importantes al proyecto. Por un lado permite importar, manipular y exportar la gran mayoría de formatos estándares de video en la industria audiovisual, utilizando una interfaz completa y ergonómica, que dispone de varios filtros y efectos de video eficientes. Además, desde 2012 la aplicación se encuentra en desarrollo y mejora continua, y posee una comunidad activa de usuarios y colaboradores.

Por otra parte, se eligió programar las herramientas de restauración bajo el estándar de desarrollo de *plugins* de efectos visuales OFX. Esto permite que los *plugins* desarrollados en esta plataforma sean compatibles con otras aplicaciones de post-producción profesionales, además de Natron, y se puedan distribuir bajo el mismo estándar que utilizan otros *plugins* de post-producción digital de video en el mercado.

A su vez se utilizó la librería de *computer vision* OpenCV. Esta elección permitió tener a disposición una gran cantidad de herramientas muy eficientes que fueron necesarias para llevar a cabo el proyecto. Sin embargo, requiere una instalación adicional, ya que no forma parte de las librerías incluidas en Natron, como si sucede con el resto de las herramientas. La integración entre Natron, OFX y OpenCV se implementó consiguiendo la mayor eficiencia posible.

Es importante destacar que, para poder desarrollar *plugins* es vital entender cómo integrar estas tecnologías entre ellas (por ejemplo Natron y OFX, u OFX y OpenCV), así como conocer sus limitaciones. No fue una tarea sencilla, pero al documentarlas y proveer un ejemplo de programación de un *plugin* en el repositorio, se allana el camino para que otros desarrolladores puedan crear sus propios *plugins* para continuar con este trabajo.

El trabajo en conjunto con el equipo interdisciplinario del AGU como usuario, nutrió en gran medida el proyecto y sirvió para orientar las investigaciones, tomar elecciones en cuanto a las diferentes tecnologías y plantear las perspectivas a futuro.

Es importante destacar aquí que actualmente el AGU cuenta con un equipo de trabajo abocado a la tarea específica de poner en funcionamiento un nuevo sistema de digitalización de alta calidad para el acervo audiovisual nacional, basado en el proyecto de hardware libre Kinograph. Se entiende que la plataforma PARP, por su carácter de software libre, es complementaria y podría ser de gran utilidad para enfrentar los problemas específicos, así como las posibilidades de restauración, que surjan con las nuevas digitalizaciones.

De las investigaciones realizadas para abordar este proyecto, se observa que la digitalización es en sí misma objeto de debate para los especialistas en conservación de material audiovisual. Algunos expertos plantean que el único método aceptable y confiable para preservar el material es la transferencia fílmico a fílmico, debido principalmente a la inestabilidad de los formatos digitales actuales.

Uruguay se encuentra atrasado tecnológicamente con respecto al panorama internacional, así como otros países de América Latina, tanto en la conservación y digitalización, como en la restauración digital del archivo y patrimonio audiovisual. En este sentido es necesario continuar formando y fortaleciendo redes interinstitucionales en este ámbito para la socialización de conocimiento y conceptos, y además para obtener el acceso a tecnologías de preservación del archivo audiovisual.

10.2. Trabajo Futuro

Mientras que en los resultados de esta tesis se ha demostrado el potencial y la eficiencia con que pueden ser restaurados algunos desperfectos en películas deterioradas, aún existe oportunidad para mejorar dichos resultados y extender el alcance del proyecto. Esta sección presenta algunas de estas direcciones.

Respecto a las mejoras en la plataforma que pueden realizarse se encuentran:

- Agregar opción de instalación en sistemas operativos Windows y Mac OS. Esto no debería implicar mayores complicaciones, ya que los algoritmos de OFX se compilan utilizando CMake que es multiplataforma.
- Migrar los *plugins* a un software de edición de video profesional y con licencia libre. Actualmente no existe una herramienta con tales prestaciones, y utilizarla podría potenciar en gran medida las herramientas de edición que puede llegar a requerir un proyecto de restauración. Esto es factible gracias a la compatibilidad del estándar OFX con diferentes *hosts* (ver Sección 2.7.3).
- Agregar procesamiento paralelizado para los algoritmos.

- Implementar algoritmos que aprovechen las posibilidades de cálculo de las unidades de procesamiento gráfico (GPU).
- Permitir visualizar, crear y editar *metadata* de los archivos trabajados. La *metadata* es una parte muy importante en el proceso de restauración de archivo audiovisual, y en la plataforma actual aún no existe posibilidad de manejarla. Cabe mencionar que no existen estándares de *metadata* de video, y en general es específica de cada formatos y de cada software. En general se presentan incompatibilidades de *metadata* al utilizar diferentes herramientas. Es deseable encontrar un camino compatible con el software libre para desarrollar un sistema de *metadata*, capaz de incorporar a los videos exportados información relevante para el proceso de restauración [1].

Para los algoritmos implementados se proponen las siguientes mejoras:

- Respecto a la herramienta de detección semi-automática de cortes se podría agregar un detector de picos de las SDA para que el parámetro *threshold* se genere automáticamente, en vez de exigir la supervisión del usuario como sucede actualmente (ver la Sección 6.2). También sería apropiado agregar una instancia opcional para corregir algunos de los cortes detectados, por ejemplo debería ser fácil eliminar falsos positivos y modificar el fotograma en el que caen los cortes detectados. Debería proponerse alguna solución para el caso de las transiciones entre planos, actualmente solo se consideran los cortes duros. Por último, se propone también combinar diferentes técnicas de detección de cortes para potenciar el desempeño, por ejemplo utilizar también la puntuación mediante diferencia de histogramas.
- Respecto a la herramienta de *deflicker* existen diferentes mejoras que pueden implementarse. Se mejorarían los tiempos de ejecución obtenidos, si en vez de calcular los histogramas necesarios cada vez que se va a procesar un fotograma, se precalcularan los histogramas de todos los fotogramas de la secuencia en un paso previo de ejecución. También podría agregarse una corrección al problema del *dithering* [127], aunque dicho problema no ha sido perceptible en las pruebas con películas deterioradas, como se mostró en la Sección 7.4. Cabe mencionar que recientemente se ha publicado "*Midway Video Equalization*" [231], una implementación de la extensión del algoritmo "*Midway Image Equalization*" para video. Dicha implementación podría ser de interés para mejorar el algoritmo de *deflicker* implementado.
- Respecto a la herramienta de detección y restauración de *scratches* se podría mejorar su desempeño si se realizara filtrado temporal para eliminar falsas detecciones. Sin embargo los diferentes abordajes propuestos en la literatura

[1] OFX propone como un cambio mayor para su próxima versión, la incorporación de un nuevo sistema de *metadata*. Esto es muy favorable porque en este caso la *metadata* sería independiente de los diferentes *hosts*[230]

sirven en determinadas circunstancias que deben ser discriminadas, y por eso agregar esta prestación al algoritmo no resulta una tarea trivial (ver Sección 3.4). Debería mejorarse el algoritmo de *inpainting* para restaurar las detecciones. Actualmente se utiliza una implementación de OpenCV para *image inpainting* que puede generar inconsistencias temporales en los resultados (ver Sección 8.6). Esto se solucionaría utilizando un algoritmo de *video inpainting* que utilice información temporal del video. En la literatura existen propuestas con resultados eficientes[158].

Un aspecto importante a mejorar es la incorporación de nuevos algoritmos de restauración que ataquen nuevos desperfectos presentes en las películas deterioradas. Al respecto se propone incorporar los siguientes algoritmos:

- Estimación de movimiento mediante técnicas de *optical flow* y *block matching* eficientes, que puedan ser utilizadas por diferentes algoritmos.
- Restauración de ruido de tipo *dirt & sparkles*.
- Eliminación de borrosidad o *deblurring*.
- Estabilización de vibraciones en la imagen.

Por último, la incorporación a corto plazo de un nuevo sistema de digitalización de alta calidad por parte del AGU, implica nuevos desafíos, algunos de los cuales podrán ser abordados por esta plataforma.

Referencias

Los enlaces a sitios *web* fueron accedidos por última vez el 22 de Mayo del 2017.

[1] Film vs digital – what is hollywood shooting on? https://stephenfollows.com/film-vs-digital/.

[2] Serge Bromberg and Eric Lange. Le voyage extraordinaire, 2011. http://www.imdb.com/title/tt2134092/.

[3] Agu. archivo general de la universidad de la república. http://www.universidad.edu.uy/renderPage/index/pageId/447.

[4] Acetato de celulosa y síndrome del vinagre. https://www.imagepermanenceinstitute.org/resources/newsletter-archive/v12/vinegar-syndrome.

[5] Estructura acetato, guía de preservación. http://www.slq.qld.gov.au/resources/preserving-collections/preservation_guides/motion-picture-film.

[6] Formatos estándares de películas analógicas. https://www.scart.be/?q=en/content/short-guide-identify-nitrate-films-and-vinegar-syndrome-degradation-audio-visual-collection

[7] Formatos estándares de películas analógicas. http://afana.org/preservation.htm#Film%20stock%20basics.

[8] Formatos estándares de películas analógicas. http://web.archive.org/web/20131201151033/filmforever.org/edgecodes.html.

[9] Formatos estándares de películas analógicas. http://www.paulivester.com/films/filmstock/.

[10] Formatos estándares de películas analógicas. http://littlefilm.org/.

[11] In the splendor of 70mm. http://www.in70mm.com/newsletter/2002/67/splendour_of_70mm/uk/index.htm.

[12] Brava project. http://brava.ina.fr/.

[13] National Film and Sound Archive. Physical damage. https://www.nfsa.gov.au/preservation/guide/handbook/damage.

[14] Proyecto idis. http://proyectoidis.org/sindrome-del-vinagre/.

[15] AGU. *Buenas prácticas en archivos históricos de servicios universitarios - Algunas lecciones derivadas de la experiencia del Área de Investigación Histórica del AGU*. AGU, 2012.

[16] International Federation of Film Archives. Fiaf: Resources of the technical commission (tc). http://www.fiafnet.org/pages/E-Resources/Technical-Commission-Resources.html.

[17] Consejo internacional de archivos. Isad-g. norma internacional general de descripción archivística. http://www.ica.org/sites/default/files/isad%20g%20SP.pdf.

[18] Julieta Keldjian, Ana Laura Cirio, and Isabel Wschebor. Primer informe: Consultoría icau sobre patrimonio audiovisual en uruguay, 2011. http://www.icau.mec.gub.uy/innovaportal/file/4713/1/primer_informe_consultor_a_icau.pdf.

[19] ASOPROD Coordinación general ICAU, OLM. Compromiso audiovisual uruguay 2015-2020, 2014. http://www.icau.mec.gub.uy/innovaportal/file/58362/1/compromiso-audiovisual-digital.pdf.

[20] Ley n.º 18.220. creación del sistema nacional de archivos, 2007. http://www.agn.gub.uy/ley18220.html.

[21] Juan Andrés Belo. Publicación cine universitario "tercer film" - todo lo que usted siempre quiso saber sobre archivos en uruguay (pero temía preguntar), 2016. http://www.revistafilm.com/wp-content/uploads/2016/06/Tercerfilm-N5-PDF-web.pdf.

[22] Publicación cine universitario "tercer film" sobre patrimonio audiovisual, 2016. http://www.revistafilm.com/tag/patrimonio-audiovisual/.

[23] Macarena F. Puig. Publicación cine universitario "tercer film" - archivos nacionales. (5), 2016. http://www.revistafilm.com/wp-content/uploads/2016/06/Tercerfilm-N5-PDF-web.pdf.

[24] PC Magazine Encyclopedia. Betacam. http://www.pcmag.com/encyclopedia/term/38568/betacam.

[25] Margaret Rouse. Video home system (vhs). http://whatis.techtarget.com/definition/VHS-Video-Home-System.

[26] PC Magazine Encyclopedia. U-matic. http://www.pcmag.com/encyclopedia/term/61997/u-matic.

[27] Margaret Rouse. Digital versatile disc (dvd). http://searchstorage.techtarget.com/definition/DVD.

[28] Sodre, archivo nacional de la imagen y la palabra. http://www.sodre.gub.uy/archivodelaimagenylapalabra.

[29] Universidad católica del uruguay. archivo audiovisual “prof. dina pintos”. http://ucu.edu.uy/es/archivoaudiovisual.

[30] Quantum Technologies. Tecnología linear tape-open (lto). http://www.quantum.com/sp/technologies/lto/index.aspx.

[31] Federal Agencies Digitization Guidelines Initiative. Digitizing motion picture film exploration of the issues and sample sow. http://www.digitizationguidelines.gov/guidelines/FilmScan_PWS-SOW_20160418.pdf.

[32] Kinescope. https://www.provideocoalition.com/kinescope-recording-televisions-antique-recording-medium/.

[33] Center for History and New Media at George Mason University. General guidelines for scanning. https://chnm.gmu.edu/digitalhistory/links/cached/chapter3/link3.45.CDPscanningguidelines.html.

[34] Federal agencies digitazion guidelines initiative. Guidelines: Motion picture film scanning projects. http://www.digitizationguidelines.gov/guidelines/Motion_pic_film_scan.html.

[35] Paul Read and Mark-Paul Meyer. *Restoration of Motion Picture Film.* Butterworth-Heinemann, 2000.

[36] The ultimate aspect ratio guide for filmmakers. http://vashivisuals.com/every-filmmaking-aspect-ratio-for-free/.

[37] Margaret Rouse. Ultra high-definition tv. http://whatis.techtarget.com/definition/Ultra-High-Definition-TV-UHDTV.

[38] Michelle S. Carlos. A comparison of scanning technologies for archival motion pictura film. https://www.academia.edu/5918000/A_COMPARISON_OF_SCANNING_TECHNOLOGIES_FOR_ARCHIVAL_MOTION_PICTURE_FILM.

[39] Renderstory blog. Log color in depth. http://renderstory.com/log-color-in-depth/.

[40] PC Magazine Encyclopedia. Espacio de color ycbcr. http://www.pcmag.com/encyclopedia/term/55147/ycbcr.

[41] Red Digital Cinema. Video chroma subsampling. http://www.red.com/learn/red-101/video-chroma-subsampling.

[42] USA National Digital Information. Digital moving-picture exchange (dpx), version 2.0. http://www.digitalpreservation.gov/formats/fdd/fdd000178.shtml.

[43] Tim Vitale. Digital image file formats. tiff, jpeg, jpeg2000, raw, dng, 2007. http://cool.conservation-us.org/coolaic/sg/emg/library/pdf/vitale/2007-07-vitale-digital_image_file_formats.pdf.

[44] Cineon. Cineon image file format draft. http://www.cineon.com/ff_draft.php.

[45] Giovanna Fossati. From grain to pixel. the archival life of film in transition., 2009.

[46] Emanuel Lorrain. Scart. website on audiovisual heritage - a short guide to choosing a digital format for video archiving masters, 2014. https://www.scart.be/?q=en/content/short-guide-choosing-digital-format-video-archiving-masters.

[47] Margaret Rouse. Telecine y pulldowns. http://whatis.techtarget.com/definition/cinema-pulldown-32-telecine.

[48] R. Matchell. Physical science, measurement and instrumentation, management and education - reviews. *IEE Proceedings A*, 129:445–453, 1982.

[49] Dansk kulturarv. http://www.dr.dk/Om_DR/Til+eksterne+producenter/01154147.htm/.

[50] Preparing an edit decision list. http://masteringfilm.com/preparing-an-edit-decision-list-edl/.

[51] Blackmagic cintel film scanner. https://www.blackmagicdesign.com/products/cintel.

[52] Arriscan technologies. http://www.arri.com/archive_technologies/arriscan/.

[53] Risen from the ashes. https://www.arri.com/news/news/risen-from-the-ashes/.

[54] Kinograph. http://thecreatorsproject.vice.com/blog/kinograph-may-be-the-savior-of-film.

[55] Film foundation. http://www.film-foundation.org/mission-statement.

[56] The film foundation. The story of movies educational program. http://www.storyofmovies.org/.

[57] National film preservation foundation. http://www.filmpreservation.org/.

[58] The library congress. https://www.loc.gov/.

[59] Presto space. http://prestospace.org/.

[60] Joanneum research institute. https://www.joanneum.at/.

[61] Werner Haas. Joanneum Research Institute Peter Schallauer, Axel Pinz. Automatic restoration algorithms for 35mm film. https://www.joanneum.at/uploads/tx_publicationlibrary/img1245.pdf.

[62] Peter Schallauer. Joanneum Research Institute. Digital image sequence restoration. https://www.joanneum.at/uploads/tx_publicationlibrary/img604.pdf.

[63] Ucla film & television archive. https://www.cinema.ucla.edu/restoration.

[64] UCLA. An interview with bob gitt, 2006. https://www.cinema.ucla.edu/restoration/interview-bob-gitt-2006.

[65] Roxanne Pena. Abs-cbn film restoration project. https://letterboxd.com/_roxannepena/list/abs-cbn-film-restoration/.

[66] The library of congress. Film schools & careers in preservation. https://www.loc.gov/programs/national-film-preservation-board/resources/film-schools-and-careers/.

[67] FIAF. Academic courses on film preservation. http://www.fiafnet.org/pages/Training/Other-Film-Preservation-Courses.html?PHPSESSID=3u93t4gonnlog7lrr9l4jgm8j5.

[68] Moving image archiving and preservation. Master of arts. http://www.fiafnet.org/pages/Training/Other-Film-Preservation-Courses.html?PHPSESSID=3u93t4gonnlog7lrr9l4jgm8j5.

[69] Selznick school of film preservation. Masters. http://selznickschool.eastmanhouse.org/masters_about.html.

[70] Gnu: Tipos de licencia de software. https://www.gnu.org/philosophy/categories.es.html.

[71] Free software foundation. http://www.fsf.org/.

[72] Gnu sistema operativo. https://www.gnu.org/home.

[73] Openbsd project. https://opensource.org/licenses/BSD-3-Clause.

[74] The debian free software guidelines (dfsg). https://people.debian.org/~bap/dfsg-faq.html.

[75] Gnu. ¿qué es el software libre? https://www.gnu.org/philosophy/free-sw.es.html.

[76] Open source definition, bsd. https://opensource.org/osd.

[77] Por qué el «código abierto» pierde de vista lo esencial del software libre. https://www.gnu.org/philosophy/open-source-misses-the-point.html.

[78] Gnu general public license. https://opensource.org/licenses/BSD-3-Clause.

[79] Licencias de software libre compatibles con gpl. https://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses.

[80] Vfx software. top 22 applications. http://www.pfind.com/vfx-software.

[81] After effects. http://www.adobe.com/uk/products/aftereffects.html.

[82] Hitfilm. https://hitfilm.com/.

[83] Adobe premiere. http://www.adobe.com/products/premiere.html.

[84] Final cut pro. http://www.apple.com/final-cut-pro/.

[85] Vegas pro. http://www.vegascreativesoftware.com/us/vegas-pro/.

[86] Davinci resolve. https://www.blackmagicdesign.com/products/davinciresolve.

[87] Nuke. https://www.thefoundry.co.uk/products/non-commercial/.

[88] Natron web oficial, 2016. http://natron.fr/.

[89] Pftrack. http://www.thepixelfarm.co.uk/pftrack/.

[90] Blender. https://www.blender.org/.

[91] Fusion. https://www.blackmagicdesign.com/products/fusion.

[92] Revisionfx. http://revisionfx.com/.

[93] Neatvideo. https://www.neatvideo.com/.

[94] Documentation : Ofx image effects api 1.4, 2017. http://openfx.sourceforge.net/Documentation/1.4/index.html.

[95] Inria. https://www.inria.fr/en/.

[96] Ejemplo de interfaz de natron. http://alternativeto.net/software/natron/.

[97] Cinepaint. http: //www.cinepaint.org/more/docs/wiki/CinePaintFilmRestoration.html.

[98] Openexr. http://www.openexr.com/.

[99] Gimp the free and open source image editor. https://www.gimp.org/.

[100] Carey Bunks. Grokking the gimp.
http://www.cinepaint.org/more/tutorial/Grokking-the-GIMP/.

[101] Licencia gnu. https://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[102] Processing. https://processing.org/.

[103] Kinograph. Image stabilization with blender 2.73.
https://www.youtube.com/watch?v=Y5o09uRTzdU.

[104] Kinograph. Kinograph image registration with fusion 7.
https://www.youtube.com/watch?v=EE_T-g8w2Pc.

[105] Arduino. https://www.arduino.cc/.

[106] Avisynth. http://avisynth.nl/index.php/Main_Page.

[107] Freddy Van de Putte. The power of avisynth: restoring old 8mm films.
http://forum.doom9.org/showthread.php?t=144271.

[108] Freddy Van de Putte. Improved avisynth 8mm film restoring script, 2010.
https://vimeo.com/13173031.

[109] Virtual dub. http://virtualdub.org/.

[110] The open effects association. http://openeffects.org/.

[111] Openfx: An open plug-in api for 2d visual effects.
http://openfx.sourceforge.net/.

[112] Tuttle ofx. http://www.tuttleofx.org/.

[113] Jean-Michel Morel Antoni Buades, Bartomeu Coll. Non-local means
denoising. http://www.ipol.im/pub/art/2011/bcm_nlm/.

[114] Opencv. http://opencv.org/.

[115] Cimg. http://cimg.eu/.

[116] Diamant film restauration.
http://www.hs-art.com/index.php/solutions/diamant-film.

[117] Dustbuster+. http://www.hs-art.com/index.php/solutions/dustbusterplus.

[118] The pixel farm. http://www.thepixelfarm.co.uk/pfclean/.

[119] Davinci revival. http://www.vision2see.de/davinci-revival-english.

[120] Phoenix film restauration.
http://www.digitalvision.tv/products/phoenix_film_restoration/.

[121] NeatVideo. Noise reduction for digital videos. https://www.neatvideo.com/.

[122] Hattarge A.M., Bandgar P.A., and Patil V.M. A survey on shot boundary detection algorithms and techniques. http://www.ijetae.com/files/Volume3Issue2/IJETAE_0213_80.pdf.

[123] Joan Cabestany, Ignacio Rojas, and Gonzalo Joya. Advances in computational intelligence: 11th international work-conference on artificial neural networks. *IWANN, Torremolinos-Málaga, Spain, June 8-10, Proceedings*, 2011.

[124] Rainer Lienhart. Comparison of automatic shot boundary detection algorithms. http: //www.vis.uky.edu/~cheung/courses/ee639_fall04/readings/spie99.pdf.

[125] Red Digital Cinema. Flicker-free video overview. http://www.red.com/learn/red-101/flicker-free-video-tutorial.

[126] Julie Delon and Agnés Desolneux. Flicker stabilization in image sequences. https://hal.archives-ouvertes.fr/file/index/docid/407796/filename/ flicker_hal_juillet2009.pdf.

[127] Thierry Guillemot and Julie Delon. Implementation of the midway image equalization. http://www.ipol.im/pub/art/2016/140/.

[128] Julie Delon. Movie and video scale-time equalization. http://dev.ipol.im/~morel/LivreGMR/A%20CITER/Fichiers_Midway/ images_films/ieee_final.pdf.

[129] F. Kelly y A. Kokaram F. Pitie, R. Dahyot. A new robust technique for stabilizing brightness fluctuations, 2004.

[130] F. Kelly y A. Kokaram F. Pitie, R. Dahyot. Localised deflicker of moving images, 2006.

[131] T. Ohuchi y T. Seto T. Saito, T. Komatsu. Image processing for restoration of heavily-corrupted old film sequences, 2000.

[132] R. Lagendijk y J. Biemond P. van Roosmalen. Correction of intensity flicker in old film sequences, 1999.

[133] Antoni Buades, Julie Delon, Yann Gousseau, and Simon Masnou. Adaptive blotches detection for film restoration. 2016. http://math.univ-lyon1.fr/~masnou/fichiers/publications/blotch-icip.pdf.

[134] P.H.S.Torr and A. Zisserman. Adaptive blotches detection for film restoration. https: //www.robots.ox.ac.uk/~vgg/publications/2000/Torr00a/torr00a.pdf.

[135] M. Irani and P. Anandan. All about direct methods. http://pages.cs.wisc.edu/~dyer/ai-qual/irani-visalg00.pdf.

[136] C.Arunkumar Madhuvappan and Dr.J.Ramesh. Video compression motion estimation algorithms – a survey, 2014. http://www.ijser.org/researchpaper/Video-Compression-Motion-Estimation-Algorithms-A-Survey.pdf.

[137] M.Jakubowski and G. Pastuszak. Block-based motion estimation techniques - a survey, 2013. http://www.uta.edu/faculty/krrao/dip/Courses/EE5359/BBMESURVEY.PDF.

[138] L.C.Manikandan and Dr. R. K. Selvakumar. A new survey on block matching algorithms in video coding. *International Journal of Engineering Research*, 3(2):121–125, 2014. https://www.ijer.in/ijer/publication/v3s2/IJER_2014_218.pdf.

[139] P. M. B. Van Roosmalen. Restoration of archived film and video, phd thesis, delft university of technology, 1999.

[140] Kokaram. Detection and removal of line scratches in degraded motion picture sequences.

[141] A. Kokaram, F.Pitie, D.Corrigan, and V.Bruno et al. Advances in automated restoration of archived meterial, 2011.

[142] Anil Kokaram. Motion picture restoration. digital algorithms for artefact suppression in degraded motion picture film and video.

[143] P.J. Bones T. Bretschneider, O. Kao. Removal of vertical scratches in digitised historical film sequences using wavelet decomposition, 2000. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.3261&rep=rep1&type=pdf.

[144] E. Ardizzone, H. Dindo, O. Gambino, and G. Mazzola. Scratches removal in digitised aerial photos concerning sicilian territory., 2007. https://iris.unipa.it/retrieve/handle/10447/35044/125079/IWSSIP.pdf.

[145] S. Müller, J. Bühler, S. Weitbruch, C. Thebault, I. Doser, and O. Neisser. Scratch detection supported by coherency analysis of motion vector fields., 2009.

[146] Kyung tai Kim and Eun Yi Kim. Film line scratch detection using texture and shape information., 2010. https://www.researchgate.net/publication/220645446_Film_line_scratch_detection_using_texture_and_shape_information.

[147] Alasdair Newson, Patrick Pérez, Andrés Almansa, and Yann Gousseau. Robust automatic line scratch detection in films. http://perso.telecom-paristech.fr/~gousseau/Scratches.pdf.

[148] L. Joyeux, O.Buisson, B. Besserer, and S. Boukir. Detection and removal of line scratches in motion picture films. https://perso.limsi.fr/vezien/PAPIERS_ACS/detection_and_removal.pdf.

[149] Alasdair Newson, Patrick Pérez, Andrés Almansa, and Yann Gousseau. Adaptive line scratch detection in degraded films, 2014. http://perso.telecom-paristech.fr/~gousseau/scratches.pdf.

[150] Amit A. Bankar and Bharat S. Borkar. Survey of automatic line scratch detection and removal in digitized film sequence. http://www.ijarcsms.com/docs/paper/volume3/issue6/V3I7-0037.pdf.

[151] V. Bruni and D.Vitulano. Removal of colour scratches from old motion picture films exploiting human perception, 2008. https://perso.limsi.fr/vezien/PAPIERS_ACS/detection_and_removal.pdf.

[152] Shanthini.B and Mahalakshmi.A. A survey on image deblurring, 2015. https://www.ijarcsse.com/docs/papers/Volume_5/11_November2015/V5I11-0192.pdf.

[153] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Process. Mag.*, 13:43–64, 1995.

[154] J. Sunghyun Cho, Jue Wang, and Seungyong Lee. Video deblurring for hand-held cameras, using patch-based synthesis. http://cg.postech.ac.kr/research/video_deblur/.

[155] Guillermo Sapiro Mauricio Delbracio. Hand-held video deblurring via efficient fourier aggregation. https://arxiv.org/abs/1509.05251.

[156] Marcelo Bertalmío, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000.

[157] Marcelo Bertalmío. Image processing for cinema, 2014.

[158] Marcelo Bertalmío, Guillermo Sapiro, and Andrea Bertozzi. Navier-stokes, fluid dynamics, and image and video inpainting, 2001. http://www.math.ucla.edu/~bertozzi/papers/cvpr01.pdf.

[159] Marcelo Bertalmío, Guillermo Sapiro, and Andrea Bertozzi. Strong continuation, contrast invariant inpainting with a 3rd order, optimal pd. *IEEE Transactions of Image Processing*, 15:1934–1938, 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.7448&rep=rep1&type=pdf.

[160] Colomba Ballester, Vicent Caselles, and Marcelo Bertalmío. Filling-in by joint interpolation of vector fields and gray levels. *IEEE International conference of Image Processing*, 10, 2001.

[161] Colomba Ballester, Vicent Caselles, and Joan Verdera. Disocclussion by joint interpolation on vector fields and gray levels, 2001. http://www.dtic.upf.edu/~jverdera/Publications/SIAMDisocMMS042245-1.pdf.

[162] N. Paragios, Y.Chen, and O.Faugeras. Mathematical models in computer vision: the handbook. capítulo: Pde-based image and surface inpainting. *Springer*, 2005.

[163] Alexandru Telea. An image inpainting techn,ique based on the fast marching method, 2003. https://pdfs.semanticscholar.org/622d/5f432e515da69f8f220fb92b17c8426d0427.pdf.

[164] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. *IEEE International Conference of COmputer Vision*, pages 1033–1038, 1999. http://www.wisdom.weizmann.ac.il/~vision/courses/2003_2/VariationalCompletion.pdf.

[165] Michael Elad and Michal Aharon. Texture synthesis by non-parametric sampling. *IEEE Transactions on Image Processing*, 15:3736–3745, 2006. http://www.egr.msu.edu/~aviyente/elad06.pdf.

[166] S. Roth and M.J. Black. Fields of experts: a framework for learning image priors. *CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:860–867, 2005.

[167] J.L. Starck, M. Elad, and D.L. Donoho. Image decomposition: Separation of texture from piecewise smooth content. *Proceedings SPIE Annual Meeting*, 2003. https://pdfs.semanticscholar.org/940f/d4c061fb9b32fe9ddfc374e936bb304520f2.pdf.

[168] Tao Ding, Mario Sznaier, and Octavia Camps. Robust identification of 2-d periodic systems with applications to texture synthesis and classification. *Proceedings of the 45th IEEE Conference on Decision & Control*, 2006. https://pdfs.semanticscholar.org/940f/d4c061fb9b32fe9ddfc374e936bb304520f2.pdf.

[169] Shantanu D. Rane, Guillermo Sapiro, and Marcelo Bertalmío. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. *IEEE Transactions on Image Processing*, 2003. https://iie.fing.edu.uy/publicaciones/2002/RSB02/Rsb02.pdf.

[170] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions of image processing*, 13(9), 2004. http://www.irisa.fr/vista/Papers/2004_ip_criminisi.pdf.

[171] V. Caselles A. Bugeau, M. Bertalmío. A comprehensive framework for image inpainting. *IEEE Transactions of image processing*, 19(10), 2010.

[172] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions of image processing*, 12(8), 2003. http://www.math.ucla.edu/~lvese/PAPERS/01217265.pdf.

[173] M. Daisy, D. Tschumperlé, and O. Lezoray. A fast spatial patch blending algorithm for artefact reduction in pattern-based image inpainting. *SIGGRAPH Asia Technical Briefs, Hong-Kong*, 2013. https://tschumperle.users.greyc.fr/publications/tschumperle_siggraphasia13.pdf.

[174] M. Daisy, P. Buyssens, D. Tschumperlé, and O. Lezoray. A smarter examplar-based inpainting algorithm using local and global heuristics for more geometric coherence. *IEEE International Conference on Image Processing (ICIP'14), Paris/Francia*, 2014. https://tschumperle.users.greyc.fr/publications/tschumperle_icip14.pdf.

[175] Peter Michael Bruce Van Roosmalen. Phdthesis: Restoration of archived film and video, 2000. http://homepage.tudelft.nl/c7c8y/Theses/PhDThesisRoosmalen.pdf.

[176] Foundry furnace. https://www.foundry.com/products/nuke/plug-ins/furnace.

[177] Foundry keylight. http://help.thefoundry.co.uk/nuke/content/reference_guide/keyer_nodes/keylight.html.

[178] Genarts sapphire. http://www.genarts.com/sapphire.

[179] Redgiant universe. https://www.redgiant.com/universe/.

[180] Natron read the docs: Denoisesharpen node. http://natron.readthedocs.io/en/master/plugins/net.sf.openfx.DenoiseSharpen.html.

[181] Adaptive image denoising by rigorous bayesshrink thresholding. http://ieeexplore.ieee.org/document/5967802/.

[182] Denoise comparison, natron and neat v4. https://www.youtube.com/watch?v=kC0bWy23b-k.

[183] Natron read the docs: Tracker node. http://natron.readthedocs.io/en/master/plugins/fr.inria.built-in.Tracker.html.

[184] Función inpaint de cimg, 2017. https://github.com/devernay/openfx-misc/tree/master/CImg/Inpaint.

[185] Natron programming guide. https://github.com/MrKepzie/Natron/wiki/OpenFX-plugin-programming-guide-(Basic-introduction).

[186] Natron programming guide - example.
https://github.com/MrKepzie/Natron/wiki/
OpenFX-plugin-programming-guide-(Invert-plugin-walkthrough).

[187] Especificacion ofx. http://openfx.sourceforge.net/Documentation/1.3/
ofxProgrammingReference.html#ImageEffectContexts.

[188] Especificacion ofx acciones. http://openfx.sourceforge.net/Documentation/
1.3/ofxProgrammingReference.html#id473661.

[189] Integración python-natron.
https://media.readthedocs.org/pdf/natron/python/natron.pdf.

[190] Instalación de opencv 2.4.13 en linux. http://docs.opencv.org/2.4/doc/
tutorials/introduction/linux_install/linux_install.html.

[191] Using opencv with gcc and cmake.
http://docs.opencv.org/2.4.13/doc/tutorials/introduction/
linux_gcc_cmake/linux_gcc_cmake.html#linux-gcc-usage.

[192] Imágenes y clips en ofx.
http://openfx.sourceforge.net/Documentation/1.0/Reference/ch08.html.

[193] Especificacion ofx - acciones, 2017.
https://media.readthedocs.org/pdf/natron/stable/natron.pdf.

[194] Ofx programming guide: Basic image processing. https://github.com/ofxa/
openfx/blob/master/Guide/Doc/ofxExample2_Invert.adoc.

[195] Mat, the basic image container. http://docs.opencv.org/2.4/doc/tutorials/
core/mat_the_basic_image_container/mat_the_basic_image_container.html.

[196] Old basic structures.
http://docs.opencv.org/2.4/modules/core/doc/old_basic_structures.html.

[197] Gary Bradski and Adrian Kaehler. Learning opencv. capítulo 3: Getting to
know opencv.
http://docs.opencv.org/2.4/modules/core/doc/old_basic_structures.html.

[198] Opencv .at method. http://docs.opencv.org/2.4.13.2/doc/user_guide/
ug_mat.html#accessing-pixel-intensity-values.

[199] Opencv basic structures.
http://docs.opencv.org/2.4/modules/core/doc/basic_structures.html.

[200] The efficient way. how to scan images, lookup tables and time measurement
with opencv. http://docs.opencv.org/2.4/doc/tutorials/core/
how_to_scan_images/how_to_scan_images.html.

[201] Valgrind. http://valgrind.org/info/tools.html.

[202] Callgrind. http://valgrind.org/docs/manual/cl-manual.html.

[203] Kcachegrind. http://kcachegrind.sourceforge.net/html/Home.html.

[204] Set of ofx plugins which perform image processing using opencv. https://github.com/devernay/openfx-opencv.

[205] Gnu. cómo elegir una licencia para su obra. https://www.gnu.org/licenses/license-recommendations.html.

[206] Licencia mozilla. https://www.mozilla.org/en-US/MPL/2.0/.

[207] Openbsd project. https://opensource.org/licenses/BSD-3-Clause.

[208] Preguntas frecuentes acerca de las licencias de gnu. https://www.gnu.org/licenses/gpl-faq.es.html#OrigBSD.

[209] El compilador gcc. tutorial. https://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm.

[210] Repositorio github de openfx. https://github.com/ofxa/openfx.

[211] Repositorio github de openfx-suportext. https://github.com/devernay/openfx-supportext.

[212] Documentación de Git. Git tools - submodules. https://git-scm.com/book/en/v2/Git-Tools-Submodules.

[213] Juan Andrés Friss de Kereki. Detección de cortes en películas. http://iie.fing.edu.uy/~agomez/__timag__tmp___/2016/shots/.

[214] Yann Gousseau. Robust automatic line scratch detections in degraded films, 2014. http://perso.telecom-paristech.fr/~gousseau/scratches/.

[215] IEEE. Movie and video scale-time equalization application to flicker reduction. http://ieeexplore.ieee.org/document/1556641/?arnumber=1556641.

[216] Julie Delon and Agnès Desolneux. Stabilization of flicker-like effects in image sequences through local contrast correction. http://epubs.siam.org/doi/10.1137/090766371.

[217] S. Ferradans, N. Papadakis, J. Rabin, G. Peyre, and J-F. Aujol. Regularized discrete optimal transport. *Proceedings of 4th International Conference on Scale Space and Variational Methods in Computer Vision*, 7893:428–439, 2013.

[218] C. Villani. Topics in optimal transportation. *Graduate Studies in Mathematics, American Mathematical Society*, 58, 2003.

[219] Martín Piñeyro, Julieta Keldjian, and Álvaro Pardo. Gpu based implementation of film flicker reduction algorithms. https://www.academia.edu/26180558/GPU_Based_Implementation_of_Film_Flicker_Reduction_Algorithms.

[220] Función de sort de opencv. http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html?#sort.

[221] Función de sortidx de opencv. http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html?#sortidx.

[222] Hsl space color. http://www.chaospro.de/documentation/html/paletteeditor/colorspace_hsl.htm.

[223] A. Desolneux, L. Moisan, and J. M. Morel. From gestalt theory to image analysis: A probabilistic approach. http://desolneux.perso.math.cnrs.fr/papers/DMM_Align_00.pdf.

[224] Transformada de hough de opencv. http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghlines.

[225] Función lineiterator de opencv. http://docs.opencv.org/ref/2.4/dc/dd2/classcv_1_1LineIterator.html.

[226] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(4):722–732, 2010.

[227] Options that control optimization in gcc. https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html.

[228] Función inpaint de opencv. http://docs.opencv.org/2.4/modules/photo/doc/inpainting.html.

[229] Documentación de OpenCV. Función cvinpaint, 2014. http://docs.opencv.org/2.4/modules/photo/doc/inpainting.html.

[230] Ofx future major changes: Metada proposed. http://openeffects.org/standard_changes/metadata.

[231] Javier Sánchez. Midway video equalization. http://www.ipol.im/pub/art/2017/181/.

Apéndice A

Glosario

2K / 4K resoluciones de video. 2K tiene 2048 píxeles de ancho, mientras que 4K tiene 4096.

Apple-ProRes es un codec o compresor de Apple que permite exportar videos con pocas pérdidas de información.

Artefactos digitales desajuste de señales debido a las técnicas de procesamiento digital que provocan una distorsión en el resultado final.

Betacam es una familia de productos profesionales de videocassette desarrollados por Sony en 1982.

BSD es una familia de licencias de software libre, que imponen restricciones mínimas sobre la redistribución del software que cubren.

Compositor aplicación que permite postproducir secuencias de video aplicando diferentes videos, imágenes y efectos.

Copia *master* película original a partir de la cual se pueden hacer otras copias.

Corte duro es una denominación para el corte clásico entre planos donde un fotograma corresponde a un plano, y el siguiente corresponde a otro.

DCP o *Digital Cinema Package* es un formato de proyección de cine.

DPX o *Digital Picture Exchange* es un formato de imagen.

DVCPro es un formato de video utilizado para capturar y editar video en HD.

Edit Decision List / EDL

Editor aplicación que permite realizar el montaje (edición) de una secuencia de video. Se trabaja con línea de tiempo para agregar, cortar o mover puntos de entrada y salida de los clips. Comúnmente utilizan efectos, transiciones, etc.

Edición no lineal (NLE) es un proceso de edición no destructivo que permite realizar tareas de edición accediendo a cualquier fotograma de la secuencia en cualquier orden (para editar el fotograma 100, no hay que pasar por el 1 al 99); se define en contraposición a la edición lineal o analógica. Las aplicaciones de edición digital actuales son editores NLE.

Escena se define como un conjunto de planos pertenecientes a una misma acción, en un mismo período de tiempo y espacio.

EXR u OpenEXR es un formato de imagen desarrollado por Industrial Light & Magic para uso en aplicaciones de procesamiento de imágenes.

Framerate / FPS significa la tasa de fotogramas por segundo.

Full HD también llamado 1080p, es una resolución de video que tiene 1920×1080 píxeles.

GCC o *The GNU Compiler Collection* es un sistema del *GNU Project* para compilar que soporta distintos lenguajes de programación.

GNU o *GNU's Not Unix!* es un sistema operativo.

GPL o *General Public License* es una licencia de software libre.

H264 es un codec estándar de baja compresión.

HD o *High Definition* (alta resolución) es una resolución de video que incluye 720p (12080×720), 1080i (1920×1080 con escaneo interlaceado) y 1080p o Full HD.

Host software, en este caso un editor o compositor de video. Algunos ejemplos son Natron, The Foundry Nuke, Assimilate Scratch, Sony Vegas, FilmLight, Baselight.

JPG o *Joint Photographic Experts Group* es un formato de imagen.

Jump cut es un tipo de corte donde se filma al mismo sujeto pero al siguiente fotograma del corte se lo ve en otro momento, manteniendo prácticamente o variando sutílmente la posición de la cámara. Esto genera un efecto de saltar hacia adelante en el tiempo.

LTO es una tecnología de cinta magnética utilizada para almacenar información, desarrollada a finales de la década del 90.

Natron es un software libre multi-plataforma para composición de video.

NTSC es un estándar de transmición de video utilizado en muchos países de América (en particular Uruguay no lo utiliza, si no que se utiliza PAL). Transmite 30 fotogramas por segundo, donde cada fotograma está formado por 525 líneas.

Nodo en Natron, un nodo representa una instancia de un *plugin* de procesamiento de imagen. Es la herramienta básica mediante la cual se ejecuta cualquier acción o modificación sobre una señal de entrada y a partir de éste se genera una señal de salida.

Open Effects (OpenFX o OFX) plataforma libre y estándar de desarrollo de *plugins* de efectos visuales para video. Interfaz de comunicación entre *hosts* y *plugins*.

OpenCV es una librería libre de *computer vision* escrita en C++ que incluye diversas funciones de procesamiento de imágenes.

PAL es un estándar de transmición de video que transmite 25 fotogramas por segundo, donde cada fotograma está formado por 625 líneas.

Plano serie de fotogramas consecutivos y contiguos tomados por una única cámara representando una acción continua en tiempo y espacio.

Plugin componentes de software que agregan alguna funcionalidad específica a una aplicación de postproducción.

PNG o *Portable Network Graphics* es un formato de imagen.

Profundidad de color (*Bitdeph*) representa la cantidad de memoria destinada por cada píxel para cada canal de color de la imagen. Por ende determina la precisión en la información de color.

PSD o *PhotoShop Document*, es un formato de imagen en capas que utiliza Adobe Photoshop.

RAW es un formato de imagen que no tiene compresión de información, se utiliza para guardar toda la información que un sensor captó.

Relación de aspecto es la proporción entre la anchura y la altura de una imagen. Por ejemplo, Una relación de aspecto de 4:3 significa que cada 4 píxeles de ancho, la imagen tiene 3 píxeles de alto.

Remasterizar refiere al proceso de aumentar la calidad de sonido, imagen, o ambas, de una copia *master*.

RGB / RGBA se utilizan para describir los canales de color de una imagen. La R representa *Red* (Rojo), la G representa *Green* (verde), la B representa *Blue* (azul) y la A representa *Alpha* (transparencia).

SD o *Standard Definition* incluye resoluciones que no son HD; en particular incluye las utilizadas por NTSC, PAL y SECOM.

SDA o Suma de Diferencias Absolutas, es un puntaje que se asigna según la diferencia entre dos imágenes.

SECAM es un estándar francés de transmición de video que al igual que PAL, transmite 25 fotogramas por segundo, donde cada fotograma está formado por 625 líneas. La diferencia con PAL radica en que transmite la información de color secuencialmente.

Secuencia de video sucesión de fotogramas.

Software libre refiere al software con autorización para que cualquiera pueda usarlo, copiarlo y/o distribuirlo, ya sea con o sin modificaciones, gratuitamente o mediante pago. Esto implica que el código fuente debe estar disponible.

SVG o *Scalable Vector Graphics* es un formato de imagen vectorial basado en XML con soporte para interactividad y animación.

Telecine refiere tanto al proceso como a los dispositivos mediante los cuales se puede digitalizar una película en formato analógico.

TIFF es un formato de imagen utilizado para intercambiar imágenes entre distintas aplicaciones, incluidas imágenes para *scanners*.

U-Matic es un formato de imagen analógico utilizado para grabar *videocassettes*.

UHD o *Ultra High Definition* es una resolución de video que incluye resoluciones mayores a 3840×2160 pixeles.

VHS o *Video Home System* es un estándar para grabar video de forma análogica en cassettes para consumo a nivel comercial.

Apéndice B

Manual de Usuario

Índice

1. Introducción

PARP, Plataforma Abierta de Restauración de Películas Deterioradas, es un proyecto de grado desarrollado en la Facultad de Ingeniería de la Universidad de la República de Uruguay, en el Instituto de Ingeniería Eléctrica (*"Profesor Agustín Cisa"*) [5] y en el Instituto de Computación (INCO) [4]. Fue desarrollado por los estudiantes de grado Sebastián Bugna y Juan Andrés Friss de Kereki.

Consiste en una serie de herramientas de efectos visuales implementadas para llevar a cabo la restauración de películas en formatos analógicos, que ya han sido digitalizadas. El código fuente se encuentra en el repositorio `https://github.com/SebastianBugna/PARP`. Las herramientas se ejecutan en Natron[3], una aplicación de distribución libre y gratuita, desarrollado en el instituto INRIA, con herramientas profesionales para composición de vídeo y efectos especiales.

Se implementaron algoritmos que permiten realizar la detección semi-automática de los cortes de una película, corregir del efecto de *flicker* de luminancia en secuencias, así como detectar y restaurar *scratches*. Es posible utilizarlas en sistemas operativos *Linux*.

En este manual de usuario cubriremos todos los pasos para restaurar una película con estas herramientas: desde la instalación del software Natron, la instalación de los *plugins* de PARP, la importación de video, los diferentes efectos disponibles para restaurar secuencias, y finalmente las opciones de exportación.

Este manual está dirigido a usuarios con alguna experiencia en postproducción de video. Las secciones están ordenadas para que pueda utilizarse como un tutorial introductorio a Natron y a la restauración de secuencias. El tiempo aproximado que toma completar el tutorial es de 3 horas.

2. Instalación

Para instalar las herramientas de *PARP* es necesario instalar:

1. El software de postproducción Natron.
2. La librería de *computer vision* OpenCV.
3. Los *plugins* de restauración de PARP.

2.1. Natron v2.2.7

Se puede descargar e instalar Natron desde el siguiente enlace:
`https://natron.fr/download`.

Por problemas específicos durante la instalación, consultar la guía de instalación de Natron:
`http://natron.readthedocs.io/en/master/guide/linux.html`

2.2. OpenCV v2.4.13

Es posible instalar OpenCV y los paquetes que esta librería requiere, directamente desde el terminal de *Linux* ejecutando los comandos:

- compilador: *$ sudo apt-get install build-essential*
- requerido: *$ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev*
- opcional: *$ sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev*

Se descarga y se instala el código fuente mediante los siguientes comandos:

- *$ cd /<my_working_directory>*

- *$ git clone https://github.com/opencv/opencv/archive/2.4.13.zip*
- *$ sudo apt-get install ffmpeg*
- *$ sudo apt-get -y install libopencv-dev*

La documentación completa sobre cómo instalar OpenCV se puede consultar en : http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html.

2.3. *Plugins* de restauración PARP

Se incluyen los siguientes tres *plugins* de restauración:

ShotCutDetection Herramienta de detección de cortes

Deflicker Herramienta de corrección de *flicker* de luminancia

RemoveScratches Herramienta para detectar y restaurar *scratches*

Es posible descargar y compilar los *plugins* descargando el código fuente del repositorio y utilizando Makefiles, ejecutando los siguientes comandos:

- *$ git clone https://github.com/SebastianBugna/PARP*
- *$ cd PARP*
- *$ make CONFIG=release*

Esto compila todos los *plugins*. Es posible compilar cada uno por separado accediendo a su subdirectorio específico y ejecutando los mismos comandos.

Las opciones de compilación más comunes son:

CONFIG=release : permite compilar la verión final y optimizada.
CONFIG=debug : permite compilar una versión sin optimizaciones, para hacer debugging.
CONFIG=relwithdebinfo : permite compilar una versión optmizada y hacer debugging.

Al compilar los *plugins* se crean subdirectorios llamados, por ejemplos: "Linux-64-*realease*". En cada uno de estos subdiretorios se crea un directorio "*.*bundle*".
Se pueden cargar en Natron los *plugins*:

- moviendo los directorios "*.*bundle*" a la carpeta "/usr/OFX/Plugins".
- Alternativamente se puede utilizar la opción en la interfaz de Natron: *Edit>Preferences...>Plug-ins>OpenFX-Plugins search path*" donde se agrega la ruta a el directorio "*.bundle".

También se debe copiar el archivo "initGui.py" del repositorio, en la carpeta " /Natron2/Natron/Plugins/PyPlugs/"

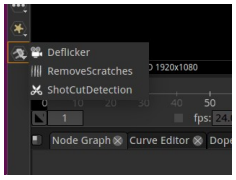


Figura 1: Grupo de *plugins* PARP cargados en Natron

Para asegurarse que los *plugins* se cargan correctamente al ejecutar Natron, verificar que se puede visualizar el siguiente grupo de *plugins* en la barra lateral izquierda en la interfaz de Natron, como se muestra en la Fig.1.

3. Uso de Natron

3.1. Entendiendo Natron

Para profundizar en los diferentes aspectos de Natron se recomienda consultar su documentación[2], así como su guía de usuario [14]. También existe una gran variedad de tutoriales [13].

3.1.1. Composición por nodos

El primer paso para entender Natron es que trabaja con nodos.

Un **nodo** es la herramienta básica mediante la cual se ejecuta cualquier acción o modificación sobre una señal de entrada y a partir de éste se genera una señal de salida. En la mayoría de los casos se tendrá como entrada y salida información de video.

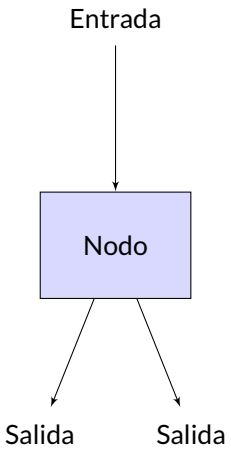


Figura 2

En Natron todo se hace a través de diferentes nodos: tanto la lectura y escritura de video, como los distintos efectos que se aplican.

Los diferentes nodos disponibles se encuentran clasificados en grupos en una barra vertical a la derecha de la interfaz de Natron.

Al pasar el *mouse* por encima de los iconos se puede ver el nombre del grupo. Al presionar en el icono de cada grupo, se despliegan los distintos nodos que contiene dicho grupo. Algunos de los grupos más utilizados son:

	*Image:* nodos básicos para cargar, exportar y visualizar videos.
	*Color:* nodos para aplicar diferentes ajustes de color a la imagen.
	*Transform:* nodos que permiten aplicar diferentes transformaciones espaciales, *e.g* escala y rotación.
	*Time:* nodos que permiten controlar los nodos temporalmente.
	*Other:* diferentes nodos para organizar la estructura de nodos. *E.g* agrupar un conjunto de nodos.

En las siguientes secciones se explica cómo utilizar algunos de estos nodos en tareas específicas.

3.2. Interfaz gráfica de Natron

En esta sección se brinda un primer acercamiento sobre la interfaz gráfica de Natron, explicando las distintas secciones y menúes que la componen. Algunos video tutoriales que explican esto mismo en mayor profundidad pueden encontrarse en [10] [12] y .

La interfaz es muy similar a la del programa comercial *Nuke* [9], por lo que si se ha trabajado con el mismo resultará sencillo de utilizar.

Se distinguen cinco secciones principales, que en la vista por defecto son las siguientes: el menú superior, el menú lateral izquierdo, el visor central con la salida de video y la línea de tiempo, la sección inferior con el grafo de nodos, y la sección de la derecha con los parámetros y las opciones de los nodos, como puede verse en la Figura 3.

1. **Menú superior**
   Aquí se puede crear un nuevo proyecto, guardar el proyecto actual, editar las preferencias de la aplicación, borrar la memoria caché, o mostrar el menú de ayuda.

2. **Menú lateral izquierdo**
   Es el menú que contiene todos los grupos de nodos que se pueden utilizar, clasificados según su funcionalidad.

3. **Vista central**
   Es el visor principal. Aquí se puede ver el video en el cual se está trabajando, y la línea del tiempo.

4. **Sección inferior**
   Por defecto la pestaña seleccionada es la de *Node Graph*, o grafo de nodos. En ella se pueden ver todos los distintos nodos del proyecto y sus conexiones.

5. **Sección derecha**
   En la sección derecha se tiene la información para cada nodo existente del proyecto, y para cada uno se pueden cambiar las opciones o parámetros que el nodo ofrezca.

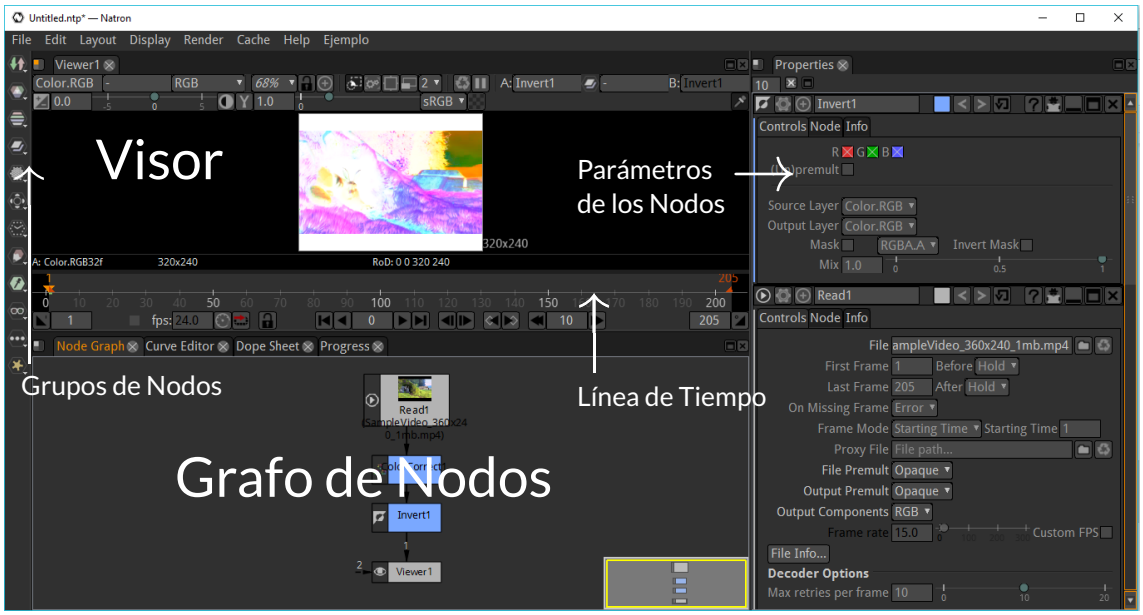


Figura 3: Interfaz gráfica de Natron y sus secciones principales

3.3. Cómo gestionar proyectos

Los proyectos de Natron son archivos de extensión .ntp. Los videos, imágenes, sonidos y materiales que se utilicen durante la composición se almacenan por separado en sus propios formatos originales.

3.4. Tareas básicas

3.4.1. Cómo crear, conectar y eliminar nodos.

Con la barra lateral donde se encuentran los diferentes grupos de nodos clasificados, es posible crear distintos nodos que se irán mostrando en el grafo de nodos de la interfaz.
Es posible posicionar y establecer conexiones entre los nodos a elección del usuario, para realizar las distintas tareas de composición y restauración.

Si se desea crear un nodo B conectado a otro nodo A existente, se sugiere seleccionar previamente el nodo A y luego seleccionar el nodo B desde el grupo de nodos en la barra lateral. De esta forma, al crear el nodo B, éste se conectará automáticamente con el nodo A. Un ejemplo práctico de esto se encuentra en la siguiente sección.

Por otro lado, si al crear el nodo B, éste no se encuentra correctamente conectado al resto de los nodos, ó aparece directamente desconectado del resto, es posible conectarlo y/o desconectarlo haciendo *click* y arrastrando las flechas de conexiones con el *mouse*.

Si en cambio dos nodos existentes A y B se encuentran conectados, y se desea conectar entre medio de A y B un nuevo nodo C, se recomienda lo siguiente:

1. Seleccionar el nodo que se desea conectar entre los otros dos nodos.
2. Presionar la tecla Ctrl . Aparecen circulos amarillos en la mitad de todas las conexiones existentes en el grafo de nodos.
3. Arrastrar el nodo seleccionado hacia la conexión deseada, en particular donde se encuentra el círculo amarillo. Se despliegan ahora flechas verdes que indican como resultará la nueva conexión, como se muestra en la Figura 4.

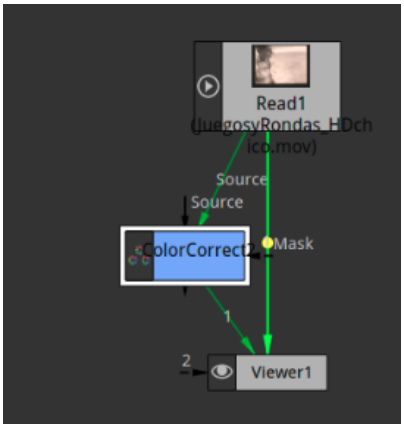


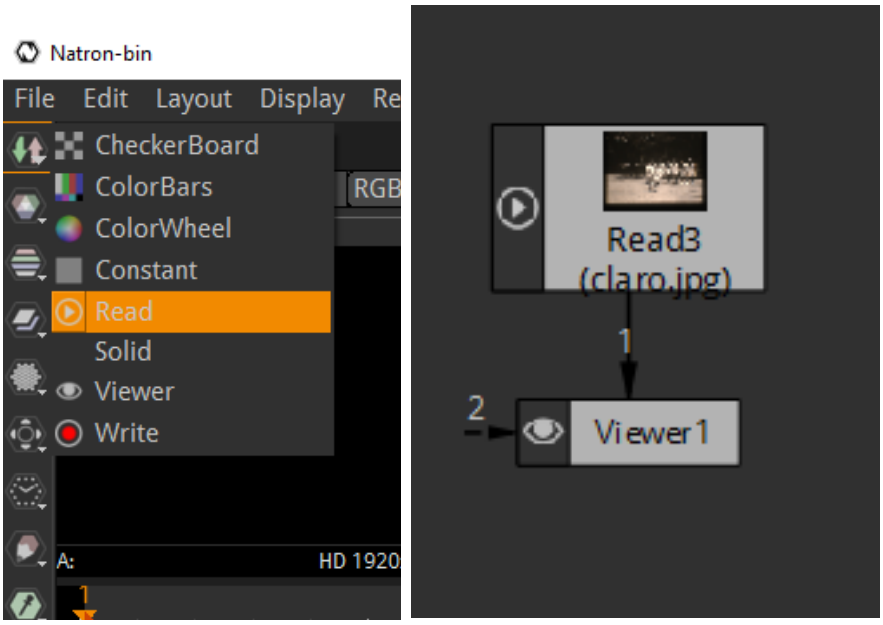
Figura 4: Parámetros por defecto del nodo *RemoveScratches*. Parámetro *Output Video* en *Detection Map*.

Por último, si se desea eliminar un nodo del proyecto, se debe seleccionar dicho nodo en el grafo de nodos y presionar la tecla *delete*.

3.4.2. Cómo cargar y ver un video

Este será el primer paso en cualquier trabajo de composición. Para la carga del video se utiliza el nodo *Read*. Para visualizar un video se utiliza el nodo *Viewer*. Los pasos son:

1. Seleccionar un nodo *Viewer* en el *Node Graph* (este nodo se crea por defecto al crear un proyecto nuevo).
2. Hacer *click* en el grupo de nodos *Image* del menú lateral izquierdo, el cual es el de más arriba.
3. Hacer *click* en la opción *Read*. Ver la Figura 5a.
4. Aparecerá una ventana para seleccionar un archivo, seleccionar el archivo de video que se deseé. Se admiten la mayoría de los formatos y resoluciones estándar de video.



(a) Crear un nodo *Read*.

(b) Nodo *Read* conectado a un *Viewer* en el *Node Graph*.

Si por alguna razón no se tiene un nodo *Viewer*, se deberá crear de la siguiente manera (ver Figura 5b):

- Hacer *click* en el grupo de nodos *Image* del menú lateral izquierdo, el cual es el de más arriba.
- Hacer *click* en la opción *Viewer*.
- Conectar ambos nodos, arrastrando la flecha saliente del nodo *Read* a la flecha 1 entrante del nodo *Viewer*, para quedar con la configuración que muestra en la Figura 5b.

3.4.3. Cómo navegar en la línea de tiempo

Debajo de la línea de tiempo se pueden encontrar botones que permiten reproducir el video normalmente, hacia atrás, o avanzando fotograma a fotograma.
Al pasar el *mouse* por encima de los distintos botones se despliegan carteles que indican sus funcionalidades. Ver Figura 6.
El marcador en color naranja índica el número de fotograma que se está visualizando actualmente.

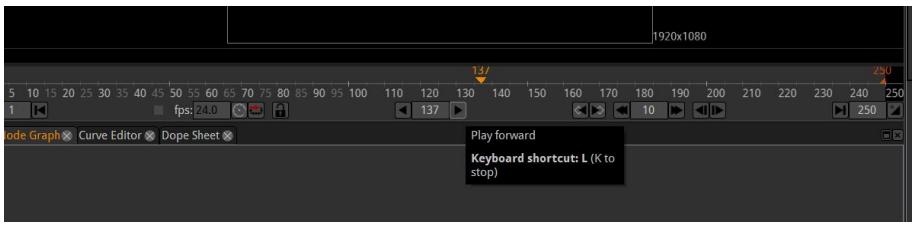


Figura 6: Herramientas de navegación en la línea de tiempo.

En la práctica se pueden realizar estas tareas directamente desde el teclado utilizando:

- K - detener el video
- L - reproducir el video
- J - reproducir marcha atrás el video
- Flechas de derecha e izquierda ←, → - avanzar de a un fotograma

También es posible maximizar los paneles presionando la barra espaciadora, por ejemplo si se desea ver el video en un mayor tamaño.

3.5. Aplicar efectos

3.5.1. Cómo aplicar efectos encadenados

Para aplicar efectos en cadena a una secuencia dada, simplemente se deben encadenar los nodos, es decir, conectar la salida de uno con la entrada de otro.

En la Figura 7, primero se aplica una corrección de color, y luego se invierten los canales de color de la imagen. Estos efectos se aplican a toda la secuencia cargada en el nodo *Read*, y el resultado se visualiza en el visor a través del nodo *Viewer1*.

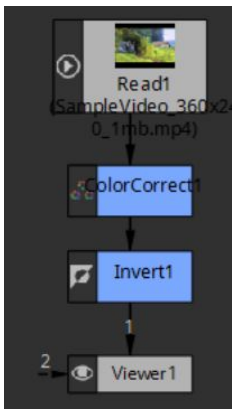


Figura 7: Aplicación de distintos efectos en encadenados.

3.5.2. Cómo aplicar distintos efectos a distintas secciones de video

Para aplicar ciertos efectos a distintos fragmentos, y no al video en su totalidad, se deben utilizar nodos *FrameRange*. Este se selecciona bajo el grupo *Time*, como se muestra en la Figura 8.

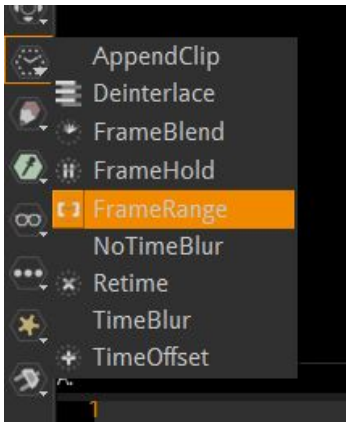


Figura 8: Nodo *FrameRange* en el grupo *Time*.

Cada nodo *FrameRange* tiene un fotograma de inicio y otro de fin.

Por ejemplo, en la Figura 9, se importó un video de prueba de 205 cuadros. Para separarlo en dos fragmentos (e.g. del fotograma 1 al 100, y del 101 al 205) se hace lo siguiente:

1. Crear dos nodos *FrameRange*.
2. Conectar ambos al nodo *Read* como muestra la Figura 9.
3. En el panel de propiedades, para cada nodo, seteamos el fotograma de inicio y fin.
4. A cada nodo, que representa una secuencia, se le aplican los efectos encadenados que se quiera.

5. Conectar estas salidas a un nodo *Merge* (se encuentra bajo el grupo *Merge*). Este nodo se encarga de unir ambos fragmentos.

6. Finalmente, conectar el nodo *Merge* a un *Viewer*.

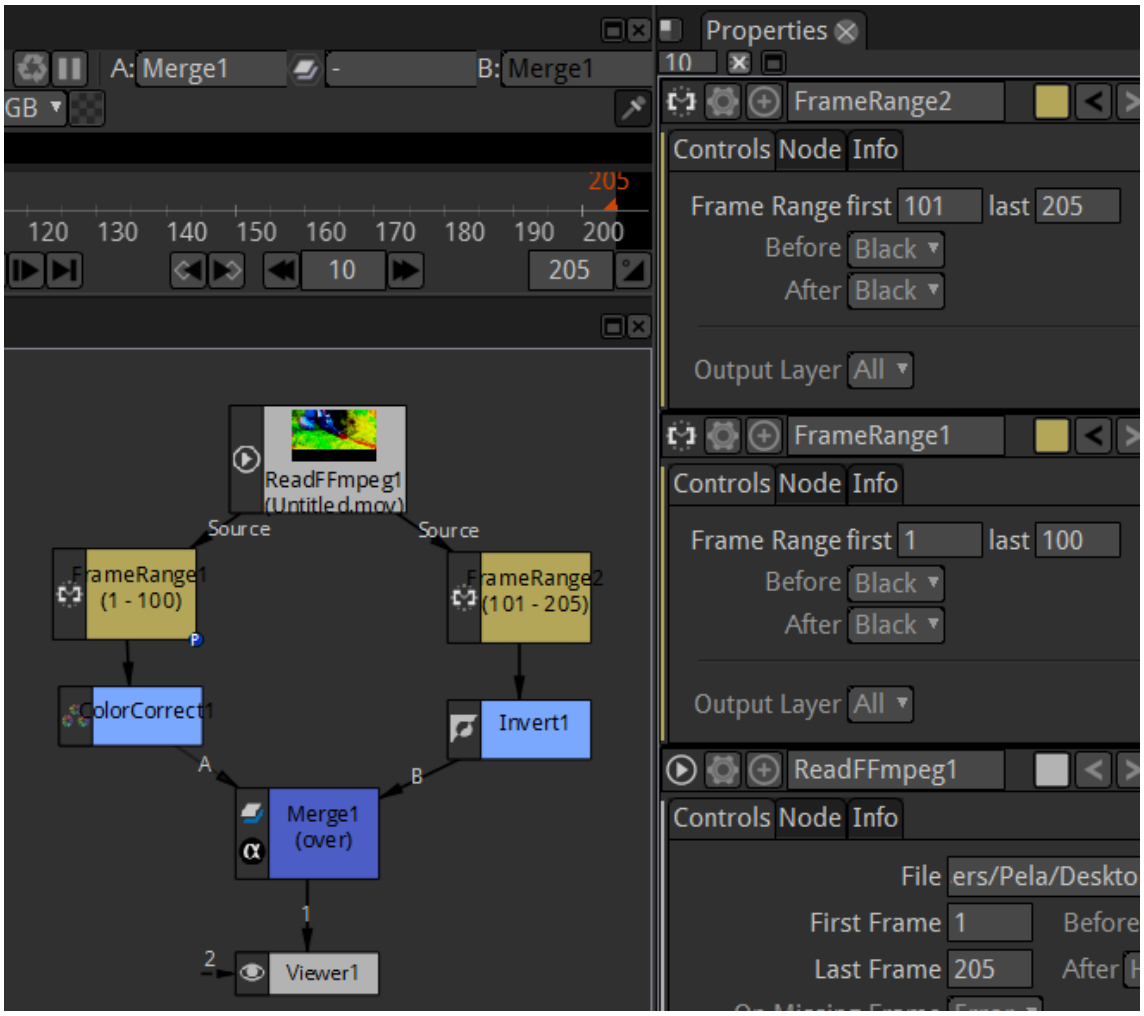
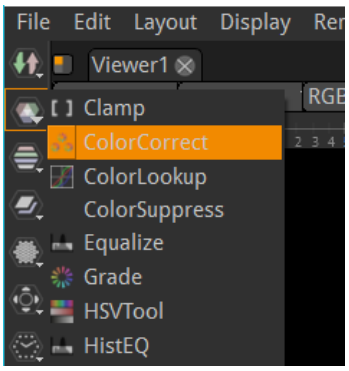


Figura 9: Aplicación de distintos efectos a distintas secciones del video.

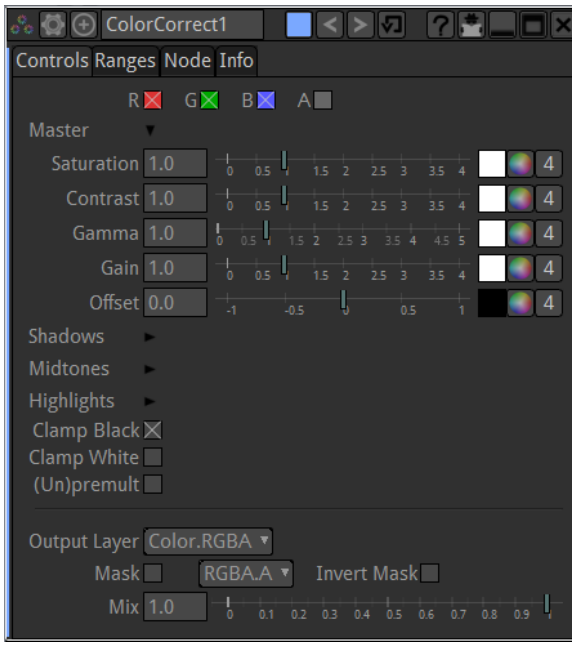
3.5.3. Corrección de color, saturación y contraste

Un solo nodo, llamado *ColorCorrect*, realiza estas tres funciones, entre otras. El mismo se encuentra bajo el grupo de *plugins Color*, como muestra la Figura 10a.

En el panel derecho, se tienen las opciones como parámetros para modificar la saturación, contraste, etc. Ver la Figura 10b. Basta con cambiar los valores y se podrá ir viendo en directo cómo afecta la imagen.



(a) Nodo *ColorCorrect* en el grupo de nodos *Color*.



(b) Parámetros del nodo *ColorCorrect*.

3.6. Cómo modificar la interfaz de trabajo para restaurar películas

La interfaz de Natron es completamente modificable, para que pueda ajustarse a distintos proyectos. Estas modificaciones se pueden guardar y cargar por defecto cada vez que se vuelva a abrir el programa.

Es muy aconsejable en el caso de restaurar películas modificar la interfaz de Natron para poder visualizar simultáneamente la versión original y la versión restaurada de la película.

En este ejemplo se cargó un video y se aplicó un nodo *ColorCorrect* con el cual aumentamos la saturación y el contraste del video original. Para cumplir con el objetivo debemos seguir los siguientes pasos:

1. Se importa un video con un nodo *Read* y se aplica algún efecto de imagen. En la la Figura 11 se aplico el nodo *ColorCorrect* a modo de ejemplo.

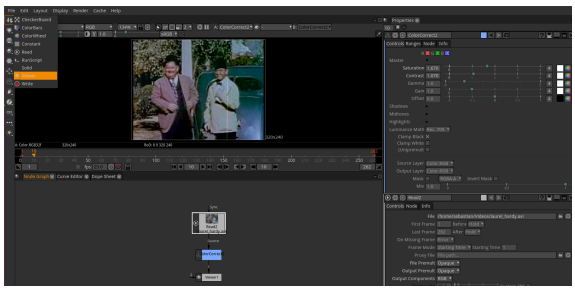
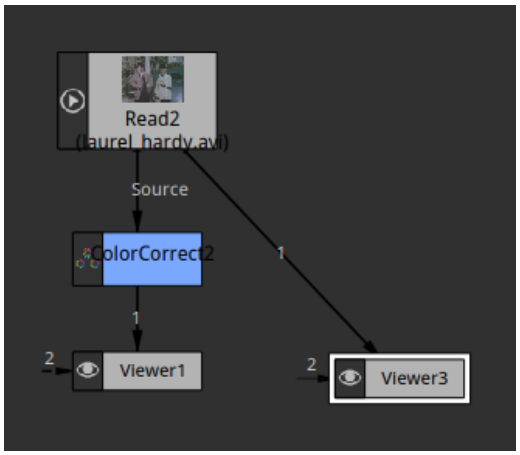
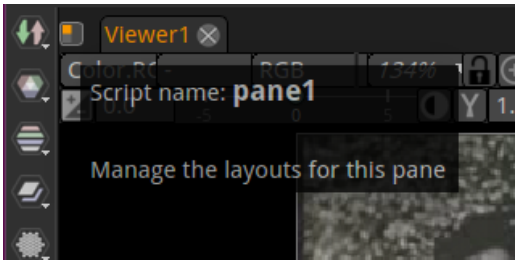


Figura 11: Se importa un video y se aplica corrección de color con nodo *ColorCorrect*.

2. Crear un nuevo nodo *Viewer*.
3. Conectarlo directamente al video original en el nodo *Read*, como muestra la figura 12a.
4. *Click* en el botón que se encuentra en la esquina superior derecha del panel central, donde se despliega el video correspondiente al nodo *Viewer1*. Ver la figura 12b.



(a) Crear nuevo nodo *Viewer* conectado al video original.



(b) Se señala botón naranja a izquierda del visor *Viewer1 ColorCorrect*.

Figura 12

5. Se despliega un menú de opciones. *click* en la opción *Split Horizontal*, como muestra la figura 13. De esta forma ahora tenemos la vista central dividida en dos columnas.

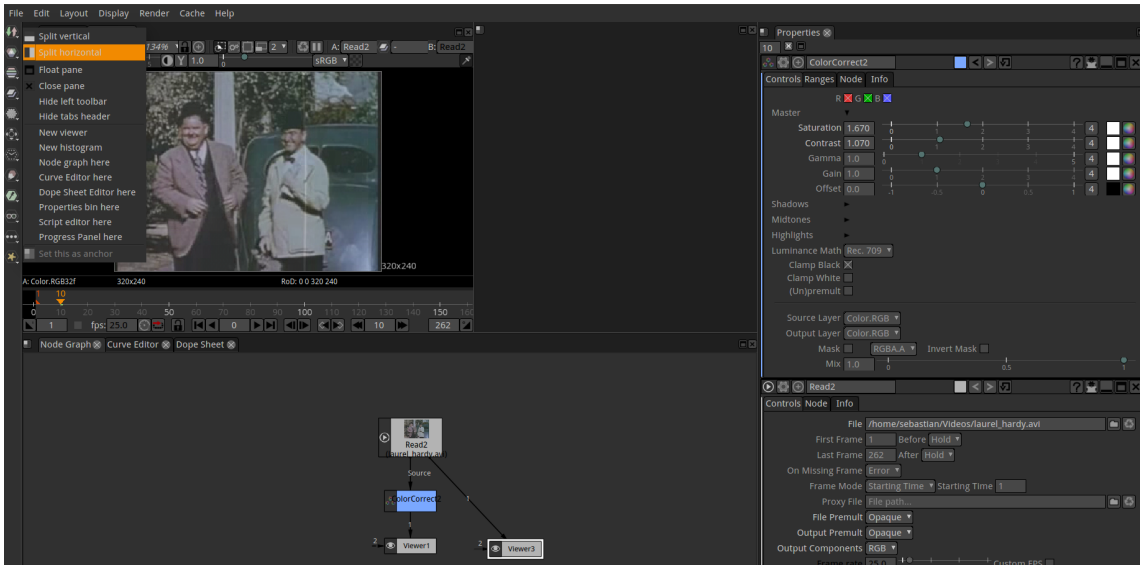


Figura 13: Seleccionar la opción *Split Horizontal* para dividir el visor principal en dos visores.

6. Arrastrar la pestaña del nodo *Viewer1* hacia el panel de la derecha.
   De esta forma en el panel central de la derecha visualizamos la versión corregida del video, y en el panel central izquierdo el video original, como se muestra en la figura 14.

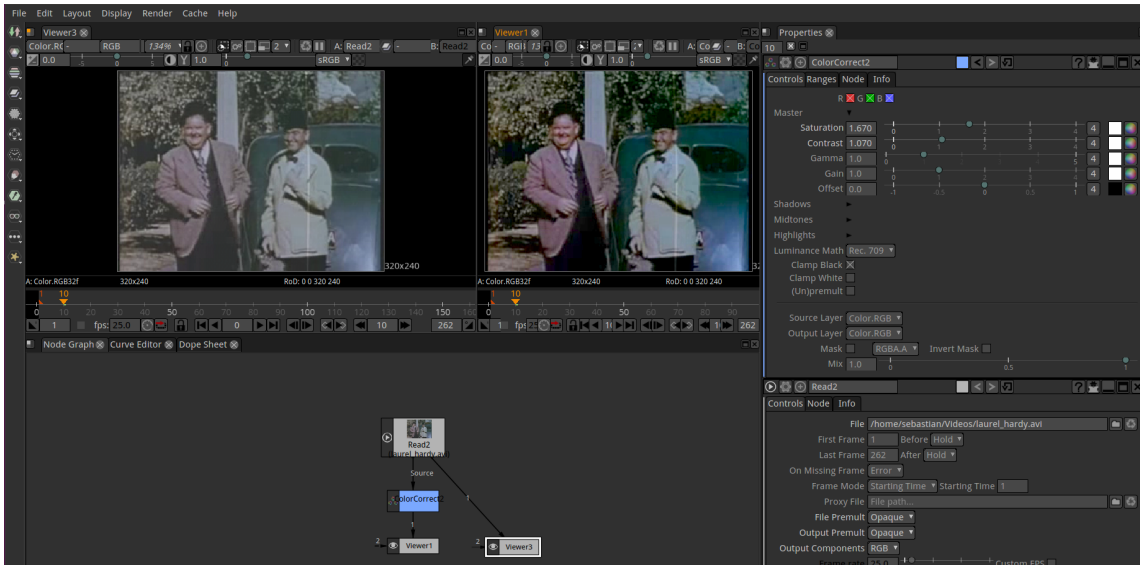
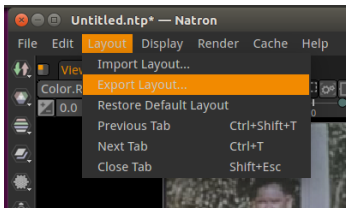


Figura 14: El espacio de trabajo ahora permite comparar la versión original con la versión restaurada.

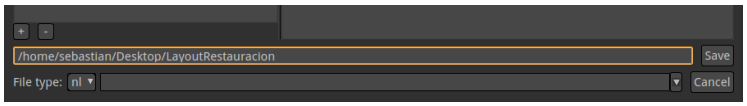
3.6.1. Guardar y configurar *Layout* por defecto de trabajo

Para no repetir este proceso cada vez que iniciamos un nuevo proyecto, es posible exportar nuestra interfaz y trabajar con la interfaz modificada cada vez que se ejecuta Natron. Se deben seguir los siguientes pasos:

1. *Click* en la opción *Layout* del menú superior.
2. *Click* en la opción *Export Layout*..., como muestra la figura 15a.



(a) Opción *Layout >Export Layout*.....



(b) Cuadro de dialogo para guardar *layout .nl*.

Figura 15

3. Se despliega un navegador para exportar nuestra interfaz, como muestra la figura 15b. Elegir un nombre y hacer *click* en *Save*. Los archivos se guardan con extensión *.nl (Natron Layout)*.
4. *Click* en *Edit* en el menú superior.
5. *Click* en *Preferences*... . Ver figura 16.

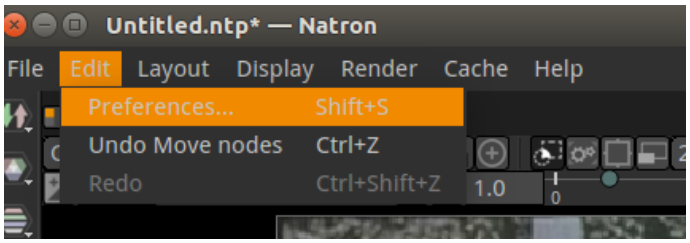


Figura 16: Opcíon *Edit >Preferences*... .

6. En el menú de preferencias hacer *click* en la opción *User Interface*.
7. En la opción *Default Layout* cargar el archivo exportado con extensión *.nl*.

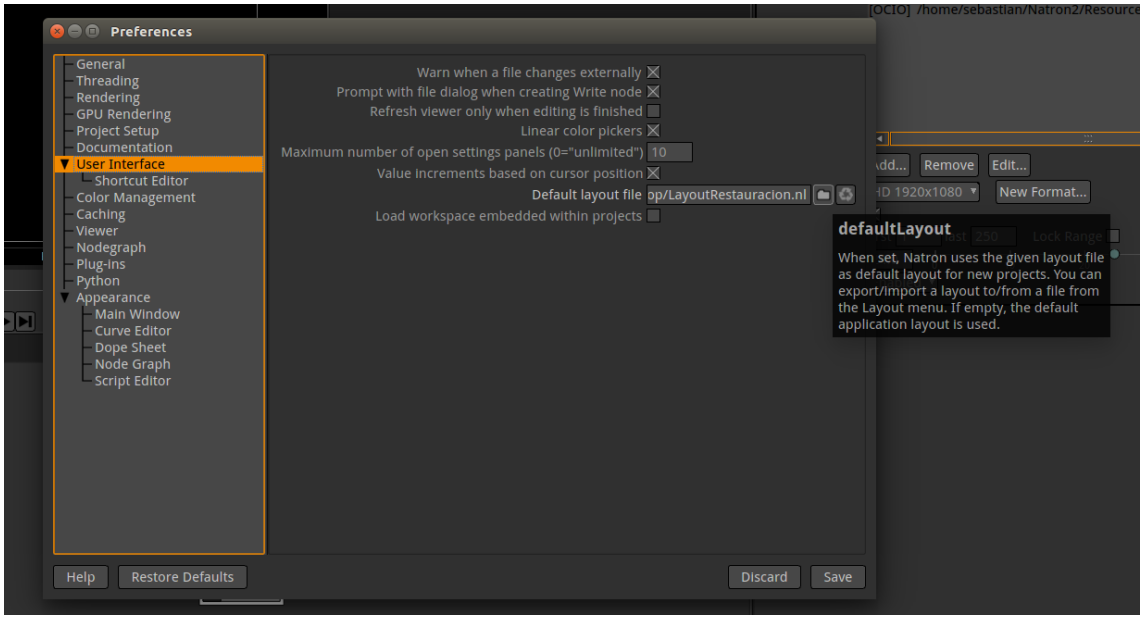


Figura 17: Menú de preferencias, sección *User Interface*.

De esta forma la próxima vez que se ejecute Natron, se cargará la vista central como dos columnas donde podremos monitorear diferentes nodos *Viewers* para guiar el proceso de restauración.

3.7. Cómo visualizar simultáneamente distintos fotogramas de un video

Durante la restauración de una película puede ser de interés comparar distintos fragmentos de la película de forma simultánea.

Esto se puede llevar a cabo mediante los siguientes pasos:

1. Se debe configurar la interfaz de trabajo para ver dos visores, como se explicó en la Sección 3.6.
2. Importar un video con un nodo *Read* y conectarlo a un nodo *Viewer1*, para visualizar su contenido.
3. Aplicar los efectos deseados al video. En el ejemplo utilizaremos un nodo *ColorCorrect*.
4. Crear un nuevo nodo *Viewer2*, que permita visualizar lo mismo que el *Viewer1*.
5. Crear un nodo *TimeOffset*.
6. Conectarlo entre el efecto y el *Viewer2*.
7. En el parámetro *Time Offset (Frames)* del nodo *TimeOffset* aplicar algún desfasaje temporal deseado. Puede ser útil utilizar el botón *scroll* del *mouse* para ir viendo los resultados en el visor.

   Se debe ingresar un número negativo de fotogramas con el que se desplaza la película que visualiza el nodo *Viewer2*, como se muestra en la Figura 18.

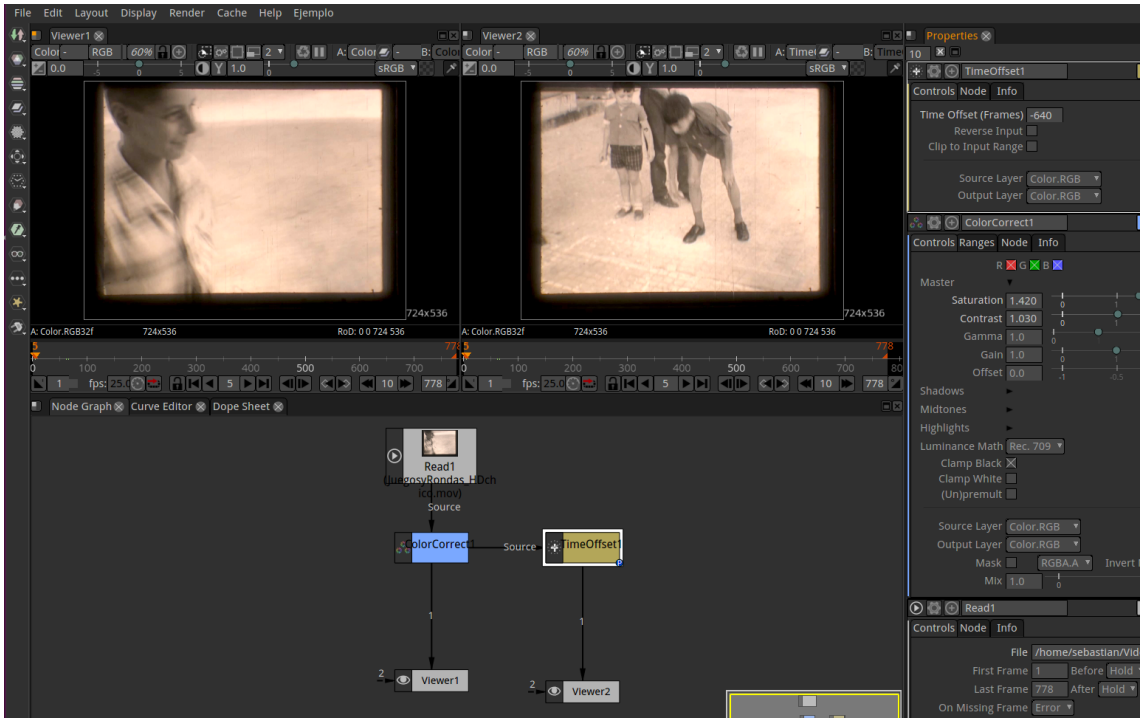


Figura 18: Visualización de dos frames de un video en forma simultánea.

4. Restauración de video

En esta sección se presentan las herramientas desarrolladas en el proyecto *PARP* para llevar a cabo la restauración de imperfectos específicos y comunes en películas deterioradas.
Se crearon tres *plugins* de restauración:

- Detección de Cortes: Permite detectar los cortes entre planos de una película.
- Detección y restauración de *scratches*: Detecta y repara los rayones presentes en las digitalizaciones de películas deterioradas.
- Corrección de *flicker* de luminancia: Corrige fluctuaciones de luminancia no naturales presentes en películas deterioradas o mal digitalizadas.

4.1. Cómo utilizar el *plugin* de detección de cortes entre escenas

Detectar los cortes entre planos de una secuencia puede ser útil para aplicar efectos solo en ciertos planos. En el caso de la restauración de material analógico deteriorado, muchas veces los filtros de restauración se aplican a una escena entera con varios planos, o incluso a la totalidad de la película. Por eso esta herramienta permite al usuario ajustar un parámetro de sensibilidad en la detección de los cortes.

Se puede detectar los cortes de una película seleccionando un nodo *Read* con la secuencia completa, y creando un nodo *ShotCutDetection* en el grupo de nodos *Parp*, en la barra lateral.

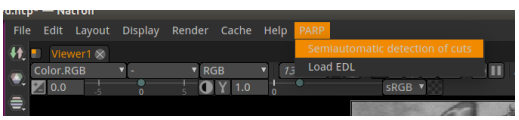
El *plugin* ofrece dos formas de funcionamiento: *Semiautomatic detection of cuts* y *Load EDL*, que explicaremos a continuación.

4.1.1. Detección semi-automática

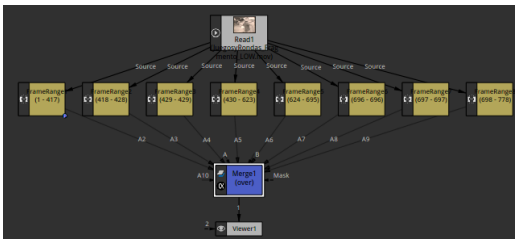
Con la opción *Semiautomatic detection of cuts* se realiza una detección automática que es supervisada por el usuario.

1. Seleccionar el nodo *ShotCutDetection*
2. Hacer *click* en el botón *Analyze Sequence*. Se desplegará el avance en una barra de progreso.
3. Al finalizar, recorrer la línea de tiempo a lo largo de la secuencia y observar que valor de pico toma el parámetro SDA en el entorno de alguno de los cortes que nos interesa detectar. Esto permite fijar el parámetro umbral *Threshold*. E.g si en varios cortes de interés el parámetro SDA tiene valores más altos que 0,2, sería razonable fijar el parámetro *threshold* a 0,2.
4. Hacer *click* en *PARP* en el menú de herramientas superior
5. Hacer *click* en *Semiautomatic detection of cuts* (ver Fig.19a).

Se crearán en el grafo de nodos, varios nodos *FrameRanges* conectados a un nodo *Merge* y a un *Viewer*. Cada nodo *FrameRange* corresponde a un plano de la película detectado. De ahora en más se puede aplicar procesamiento diferenciando distintas secciones del video de acuerdo a los cortes detectados (ver Fig. 19b).



(a) Menú de PARP en el menú de herramientas de Natron.



(b) Nodos generados para una secuencia de ejemplo al utilizar el algoritmo de detección de cortes.

Figura 19

4.1.2. Importar *EDL (Edit Decision List)*

Las listas estándar de edición en la industria son de tipo *EDL* o *Edit Decision List*. En primer lugar se debe tener o generar un archivo de este tipo, con al menos los siguientes campos: Número de Escena, Primer Fotograma, Último Fotograma. Un ejemplo de archivo válido para una secuencia de 70 fotogramas se presenta a continuación.

Listing 1: EDL.txt

```
Shotnumber  FirstFrame  LastFrame

1  1  10
2  11  25
3  26  70
```

Para importar el archivo y generar los nodos a partir del mismo, se debe hacer *click* en *Parp* en el menú de herramientas superior y luego hacer *click* en la opción *Load EDL*. Se abrirá una ventana para seleccionar un archivo, se selecciona el archivo deseado y con esto se crearán los nodos según la lista brindada.

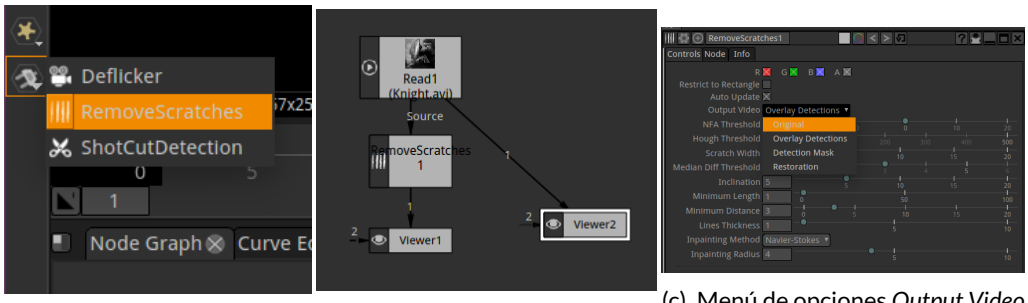
4.1.3. Resultados esperados

- La herramienta semi-automática de detección de cortes puede detectar falsos cortes si la secuencia contiene fluctuaciones de luminancia violentas. Se recomienda en estos casos ajustar el parámetro *Sensitivity* para evitar realizar esto cortes a la película.
- Se recomienda no generar demasiados nodos *FrameRange* en un mismo proyecto, ya que al sobrecargar el grafo de nodos el proyecto se vuelve poco fluido. En el caso de que fuera necesario se sugiere dividir el trabajo en varios proyectos donde se trabajan diferentes secciones de la película.

4.2. Cómo utilizar el *plugin* de detección y corrección de *scratches*

Para detectar y restaurar *scratches* se debe aplicar al nodo de video el *plugin RemoveScratches* bajo el grupo PARP en el menú lateral izquierdo. Ver la figura 20a.

Se puede aplicar a secuencias largas con cortes entre planos, ya que el algoritmo trabaja fotograma por fotograma, y es robusto a los cambios entre escenas.



(a) Nodo *RemoveScratches* en el grupo de nodos *Parp*

(b) *RemoveScratches* conectado a un *Viewer*.

(c) Menú de opciones *Output Video* y otros parámetros del *plugin*

Figura 20

El *plugin* se conecta a un nodo *Read*, como en la Figura 20b, y se utiliza en dos instancias. En la primer instancia el usuario debe supervisar la detección adecuada de los *scratches* que se desean restaurar y en la segunda se restauran los *scratches*.

Para ello se utiliza el parámetro *Output Video*, en el cual se pueden seleccionar cuatro opciones, como se muestra en la Figura 20c:

- *Original:* Se muestra el video original.
- *Overlay Detection:* Se imprimen sobre el video y en color azul los *scratches* detectados.
- *Detection Mask:* Se imprimen únicamente las detecciones sobre fondo negro.
- *Restoration:* Se restauran los *scratches* detectados.

Por defecto el *plugin* utiliza la opción de salida *Overlay Detections*, para que el usuario comience ajustado la detección. Si bien el algoritmo funciona de forma automática, es posible mejorar los resultados ajustando los diferentes parámetros que se describen a continuación:

- *Scratch Width*: ancho del *scratch* en píxeles. Para gran parte de los videos es adecuado un valor de 5, sin embargo para resoluciones mayores a Full HD los *scratches* pueden tener anchos mayores. Este valor puede ajustarse entre 2 y 10.
- *Median Difference Threshold*: umbral para la diferencia entre el nivel de gris del píxel y su mediana horizontal. Existen casos en que este valor para un *scratch* puede estar por debajo del umbral en ciertos fotogramas, y esto genera inestabilidades temporales en los resultados. Este valor puede ajustarse para valores entre 1 y 6.
- *Hough Threshold*: umbral para las detecciones de la transformada de Hough. Este valor puede ajustarse para limitar la cantidad de detecciones, y recuperar falsos negativos. Es posible ajustarlo entre valores de 10 a 500.
- *Inclination*: inclinación máxima (en grados) considerada para los *scratches* con respecto a la dirección vertical. Este parámetro puede ajustarse para eliminar falsas detecciones con ángulos mayores al de los *scratches*. Se pueden utilizar valores entre 1 y 20 grados.
- *Minimum Length*: largo mínimo considerado para los *scratches*. Se fija automáticamente como la décima parte de la altura de la imagen, pero puede ser ajustado para obtener *scratches* con largos mínimos entre 2 y 500 píxeles.
- *NFA Threshold*: impone cota al número esperado de falsas alarmas bajo el modelo de fondo. Se fija en 0 pero puede ser ajustado por el usuario en Natron, para obtener mayor o menor número de detecciones, dependiendo si el interés es mejorar la precisión o el *recall* del algoritmo. Se modifica utilizando potencias de 10, con valores entre 10^{-20} y 10^{20}.
- *Minimum Distance*: distancia mínima aceptada entre las detecciones (en píxeles). Puede ajustarse para eliminar detecciones muy cercanas entre sí, con valores entre 0 y 20.
- *Line Thickness*: permite al usuario ensanchar o afinar el grosor de los segmentos detectados. Es útil en casos en que el perfil horizontal del *scratch* es irregular y para extender la región donde se realiza el *inpainting*. Se pueden ajustar entre 1 y 5 píxeles.
- *Restrict to Rectangle*: [Opcional] por defecto se encuentra desactivado. Al activarlo el usuario puede restringir el área en el que se realizan las detecciones, directamente en el visor de Natron. Esto es útil para *scratches* localizados, o cuando existen regiones de la imagen donde se generan falsos positivos.

Cuadro 1: Parámetros de entrada de usuario y sus valores por defecto para el *plugin RemoveScratches*. También existe el parámetro *booleano* de ventana ajustable, que por defecto está desactivado.

Parámetro	Scratch Width	Median Diff Thresh	Hough Thresh	Inclination	Min. Length	NFA Thresh	Min. Dist.	Line Thickness
Valor por defecto	5	3	90	10	$\frac{M}{10}$	1	3	1
Rango de variación	[2..10]	[2..6]	[10..500]	[1..20]	[2..500]	$[10^{-20}..10^{20}]$	[0..20]	[1..10]

Ajustando estos parámetros es posible mejorar las detecciones a cada caso específico. Ver la Figura 21.

Figura 21: Parámetros por defecto del nodo *RemoveScratches*. Parámetro *Output Video* en *Detection Map*.

4.2.1. Restauración de *scratches* mediante *video inpainting*

Por último para restaurar los *scratches* detectados se debe elegir en el parámetro *Output Video* la opción *Restoration*, como muestra la Figura 22. Se utilizan técnicas de *video inpainting* para rellenar los *scratches* detectados.

Figura 22: *scratches* restaurados con el parámetro *Output Video* en *Restoration*

Para ajustar la restauración se utilizan los siguientes parámetros:

- *Inpainting Method*: permite elegir entre dos métodos para realizar *image inpainting*: *Navier-Stokes* o *Fast Marching Squares*. Por defecto se elige usar *Navier-Stokes* porque provee resultados más precisos, sin embargo puede generar mayores incoherencias temporales en las secuencias.
- *Inpainting Radius*: determina los píxeles en una vecindad circular a cada punto que será restaurado. Al aumentar el radio se consideran más píxeles vecinos para realizar la interpolación. Por defecto el valor es de 4 píxeles. Se puede ajustar para valores entre 1 y 20 píxeles.

4.2.2. Ventana de procesamiento interactiva

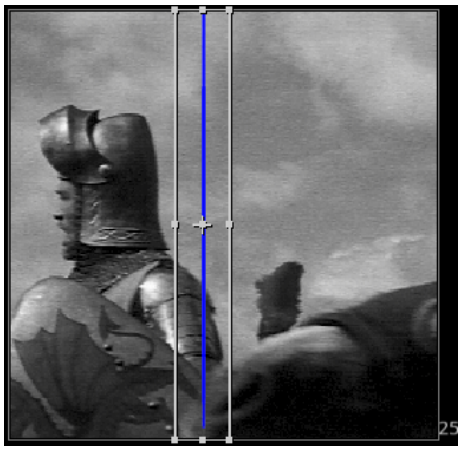


Figura 23: Ajuste de ventana de procesamiento

Otra herramienta útil durante la instancia de detección es la opción *Restrict to Rectangle*, que por defecto se encuentra habilitada, como se muestra en la figura 21. Esta opción permite al usuario ajustar una ventana interactiva dentro de la cual realizar las detecciones.

La ventana interactiva puede ajustarse directamente sobre el video, como muestra la figura 23, ó con los parámetros:

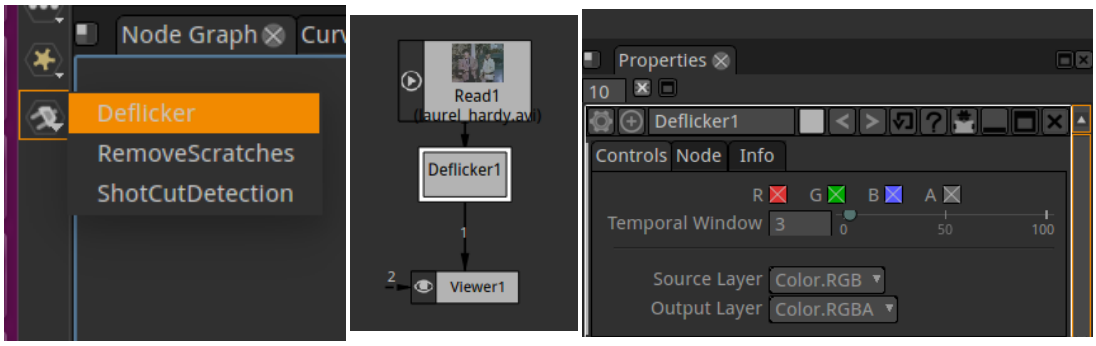
- *Bottom Left :* Posición de la esquina inferior izquierda.
- *Size w :* Ancho deseado en pixeles.

4.2.3. Resultados esperados

- Es posible mediante el ajuste de parámetros detectar la gran mayoría de los *scratches* que aparecen en las secuencias.
- Si los parámetros se ajustan de manera demasiado permisiva, la herramienta podría confundir con *scratches* líneas o segmentos presentes en la textura natural de la imagen.
- Puede ocurrir que un mismo juego de parámetros no sea útil durante toda la duración de un plano. Es posible crear fotogramas claves en los parámetros para solucionar este problema. Ver documentación de Natron [2].
- La restauración de los *scratches* detectados se realiza mediante técnicas de *image inpaiting*, que pueden generar incoherencias temporales detectables en el resultado. Si bien el algoritmo puede no dar un resultado satisfactorio en regiones de la imagen con mucho detalle, funciona adecuadamente en zonas con texturas suaves y uniformes.

4.3. Cómo utilizar el *plugin* de *deflicker*

Para quitar el efecto de *flicker* de un video, se debe utilizar el nodo *Deflicker* que se encuentra bajo el grupo de nodos *Parp*, como muestra la Figura 24a.



(a) Nodo *Deflicker* en el grupo de nodos *Parp*.

(b) *Deflicker* conectado a un *Viewer*.

(c) Parámetro *Temporal Window* ajustado a 3 fotogramas.

Figura 24

El *plugin* se utiliza igual que la mayoría, se conecta a una entrada y una salida, como se muestra en la figura 24b.

Tiene un único parámetro, *Temporal Window*, que indica la cantidad de frames totales que tomará en cuenta el algoritmo para hacer el cálculo del histograma común. Ver la figura 24c.
Cuanto mayor sea este parámetro, mejores resultados pero mayor tiempo de procesamiento. Es un entero que se setea con una barra deslizadora o se escribe en las opciones del parámetro.
Se recomienda empezar con valores chicos (e.g 3 ó 5), e incrementarlo hasta que se desee, observando cualitativamente los resultados de la restauración.

4.3.1. Resultados esperados

- Para utilizar correctamente esta herramienta, se debe previamente realizar la detección de cortes entre planos.
- La herramienta *Deflicker* no soluciona problemas locales de *flicker*, sino que se aplica globalmente a toda la imagen.

4.4. Herramientas de restauración disponibles en Natron

Si bien Natron no es una aplicación de restauración estrictamente, cuenta con una serie de *plugins* útiles para restaurar determinados deterioros presentes en películas deterioradas. Se resumen a continuación dichas herramientas:

***Sharpen* y reducción de ruido [6]** permite eliminar ruido y/o agregar nitidez (*sharpness*) a las imágenes, utilizando un método basado en *wavelets*, y con resultado eficientes dentro del estado del arte[1].

Estabilización de imagen [8] permite rastrear (“trackear”) uno o más puntos 2D y utilizarlos para estabilizar la imagen, en caso de que la imagen de entrada presente vibraciones de posición y rotación.

***Image Inpainting* [11]** es un *plugin* en versión *beta* para realizar *image inpainting*. La implementación utiliza la librería abierta CImg.

Este puede ser útil para remover manchas y roturas grandes en fotogramas particulares de una secuencia (ver Fig. 25). De todas formas, no se trata de un algoritmo automático para video. Para proveer la máscara donde se realiza el rellenado se puede por ejemplo utilizar el nodo *Roto* [7] en Natron para dibujar diferentes figuras.

Figura 25: Proyecto de Natron donde se trabaja con la película de ejemplo. A la izquierda en el visor de Natron, el fotograma original. A la derecha, *zoom* de la imagen con una mancha restaurada mediante el *plugin* de *image inpainting* de Natron.

5. Exportación de video

1. *Click* en el grupo de nodos *Image* del menú lateral izquierdo, es el primer grupo de la barra lateral izquierda.
2. *Click* en la opción *Write*.
3. Se abrirá una ventana para seleccionar la extensión, ruta y nombre del archivo a crear. Completar estos campos y presionar aceptar. Ver la figura 26.
4. Conectar el nodo *Write* al nodo de salida que se quiera. Ver la figura 27.
5. En el panel de la derecha, luego de setear las opciones del nodo que se deseen, hacer *click* en el botón *Render*, dentro del cuadro con los parámetros del nodo *Write*. Se desplegará el avance de la exportación en una barra de progreso.

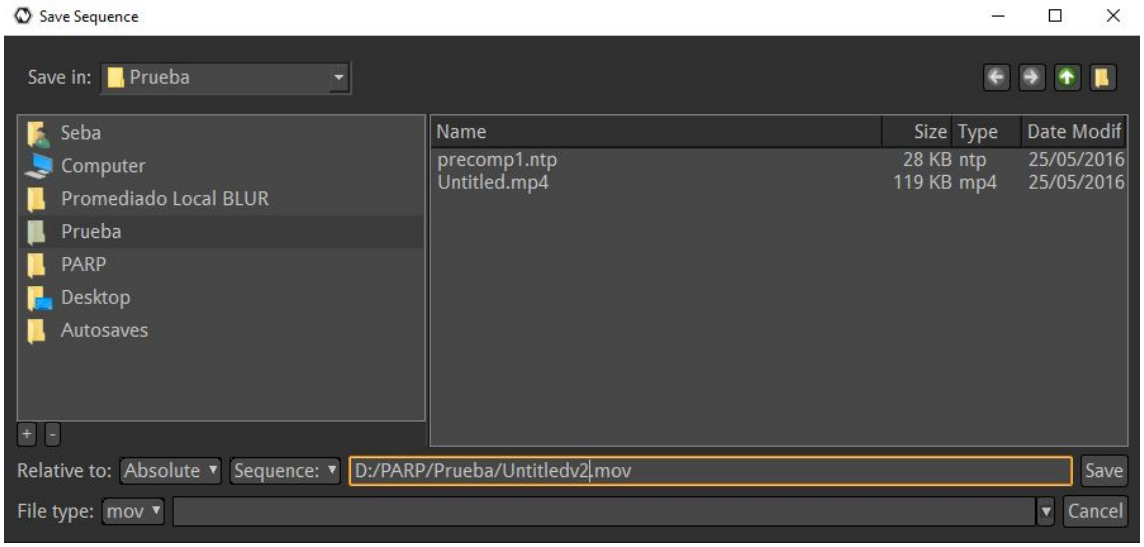


Figura 26: Cuadro de diálogo al exportar video con nodo *Write*.

En el ejemplo de la figura 27 se eligió formato de exportación *Quicktime* (extensión *.mov*), *compresión Apple Pro Res 444*.

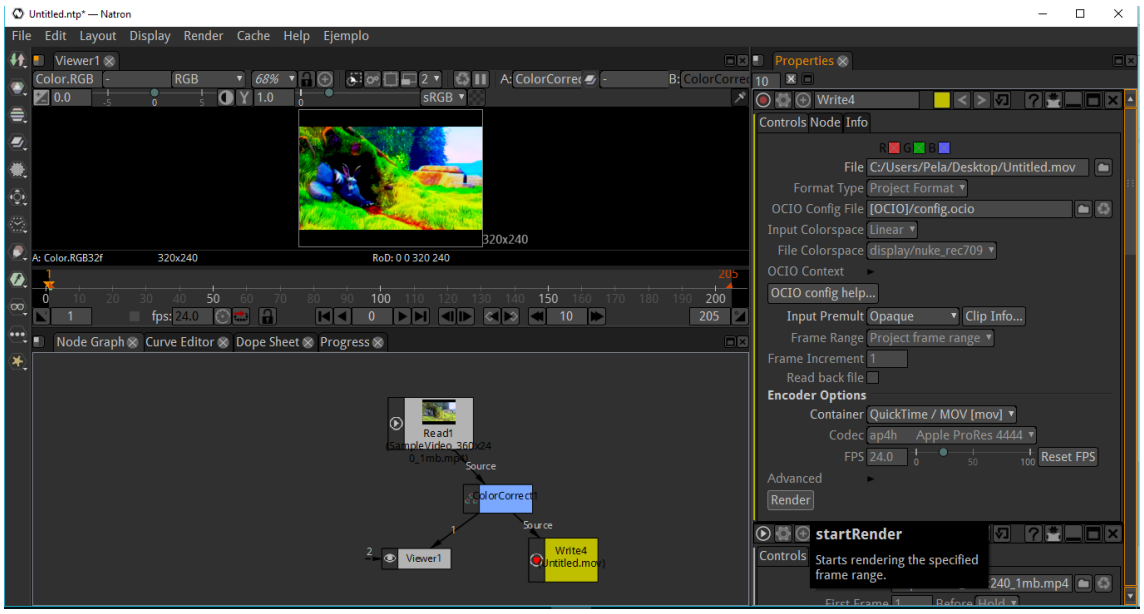


Figura 27: Exportación de video mediante nodo *Write*.

Referencias

[1] Denoise comparison, natron and neat v4. https://www.youtube.com/watch?v=kC0bWy23b-k.

[2] Natron documentation v3.0. URL: https://natron.readthedocs.io/en/master/.

[3] Natron. INRIA. URL: http://natron.fr/.

[4] UdelaR. Instituto de Computación (INCO). URL: https://www.fing.edu.uy/inco/inicio.

[5] UdelaR. Instituto de Ingeniería Eléctrica (Prof. Agustín Cisa). URL: https://iie.fing.edu.uy/.

[6] Natron read the docs: Denoisesharpen node. http://natron.readthedocs.io/en/master/plugins/net.sf.openfx.DenoiseSharpen.html.

[7] Natron read the docs: Roto node. http://natron.readthedocs.io/en/master/plugins/fr.inria.built-in.Roto.html.

[8] Natron read the docs: Tracker node. http://natron.readthedocs.io/en/master/plugins/fr.inria.built-in.Tracker.html.

[9] The Foundry Nuke. URL: https://www.thefoundry.co.uk/products/non-commercial/nuke-non-commercial/.

[10] Natron Tutorial Primeros Pasos. URL: https://www.youtube.com/watch?v=TUGkvRdlGTU.

[11] utilizando la librería Cimg *Plugin* de de inpainting para Natron. Función *CimgInpaint*, 2017. https://github.com/devernay/openfx-misc/tree/master/CImg/Inpaint.

[12] Natron Interface Overview Tutorial. URL: https://www.youtube.com/watch?v=dGA09_qzuqk.

[13] Natron tutorials. URL: https://forum.natron.fr/c/tutorials.

[14] Natron user guide. URL: https://natron.readthedocs.io/en/master/guide/index.html#user-guide.

Apéndice C

Plan de Proyecto

Índice

1. Resumen

Estudiantes

Sebastián Bugna	IIE	47474167	sebastian.bugna@gmail.com
Juan Andrés Friss de Kereki	INCO	46631887	juanfkt93@gmail.com

Cliente

AGU - Archivo General de la UdelaR

Tutores

Gregory Randall (IIE), Eduardo Fernández (INCO).

Fecha prevista de finalización

15/03/2017

Total de horas a realizar previstas

1000 hs.

Fecha y descripción de los entregables intermedios

15/09/2016 Hito1
Documentación detallada hasta la fecha.
Análisis con respecto a diagrama de Gantt, riesgos y planificación en general.

15/02/2017 Hito2
Informe final (Tesis)
Demostración de aplicación de la plataforma sobre una película corta
API, Documentación técnica del software

2. Descripción del proyecto

El deterioro inevitable del celuloide provoca la pérdida de películas que son parte de una herencia cultural. Estas degradaciones incluyen suciedades que se adhieren al fílmico, roturas, flickeos, rayones y pérdida de color, entre otros.

La restauración manual mediante técnicas de postproducción es tediosa y consume una inmensa cantidad de tiempo. Es deseable encontrar métodos automatizables capaces de restaurar estos imperfectos mediante procesamiento de señales.

El Archivo General de la Universidad (AGU), cuenta con decenas de películas uruguayas de carácter histórico que han sido rescatadas en condiciones diversas de degradación.

En este proyecto nos proponemos realizar una búsqueda de diferentes tecnologías disponibles, con el objetivo de programar una interfaz gráfica libre que permita visualizar, restaurar y exportar una película de forma ergonómica, así como explorar distintos algoritmos de restauración digital.

3. Objetivo general

El principal objetivo es diseñar una plataforma libre que permita al usuario aplicar diferentes técnicas de procesamiento de video para restaurar películas deterioradas previamente digitalizadas.

4. Alcance

El alcance del proyecto abarca los siguientes puntos:

- Montar y dejar en funcionamiento una plataforma libre, con buenas condiciones de reusabilidad y evolucionabilidad.
- Estudio e implementación de algoritmos de restauración de las distintas degradaciones típicas.
- Poder evaluar la aplicación con una película corta que reúna algunos de estos problemas.

5. Criterios de éxito

Se proponen los siguientes criterios de éxito con el fin de medir cumplimiento el del proyecto:

- Restaurar al menos dos de los problemas de degradación propuestos.
- Poder exportar e importar en al menos un formato apropiado de video.
- Poder instalar y utilizar la plataforma en un sistema operativo.

6. Actores

- Equipo de Proyecto: Juan Andrés Friss de Kereki (INCO), Sebastián Bugna (IIE).
- Tutor IIE - Gregory Randall
- Tutor INCO - Eduardo Fernandéz
- Tutor IIE - Mauricio Delbracio
- Contacto con el AGU: Mariel Balas, Isabel Wschebor, Ignacio Seimanas
- Consultas sobre restauración digital y herramientas de programación: Pablo Musé, Juan Cardelino

7. Supuestos

- El AGU suministrará películas deterioradas ya digitalizadas, que previamente serán elegidas por el equipo de trabajo.
- El AGU suministrará un escaneo en alta resolución de una película corta antes de finalizar el año.
- Se intentará encontrar opciones libres disponibles para el diseño de la plataforma y para la implementación de los algoritmos, por lo que muchas de las tareas se podrían resumir.

8. Restricciones

- Se utilizaran únicamente películas de prueba suministradas por el AGU.
- La plataforma será diseñada para un único sistema operativo.

9. Especificación funcional del proyecto

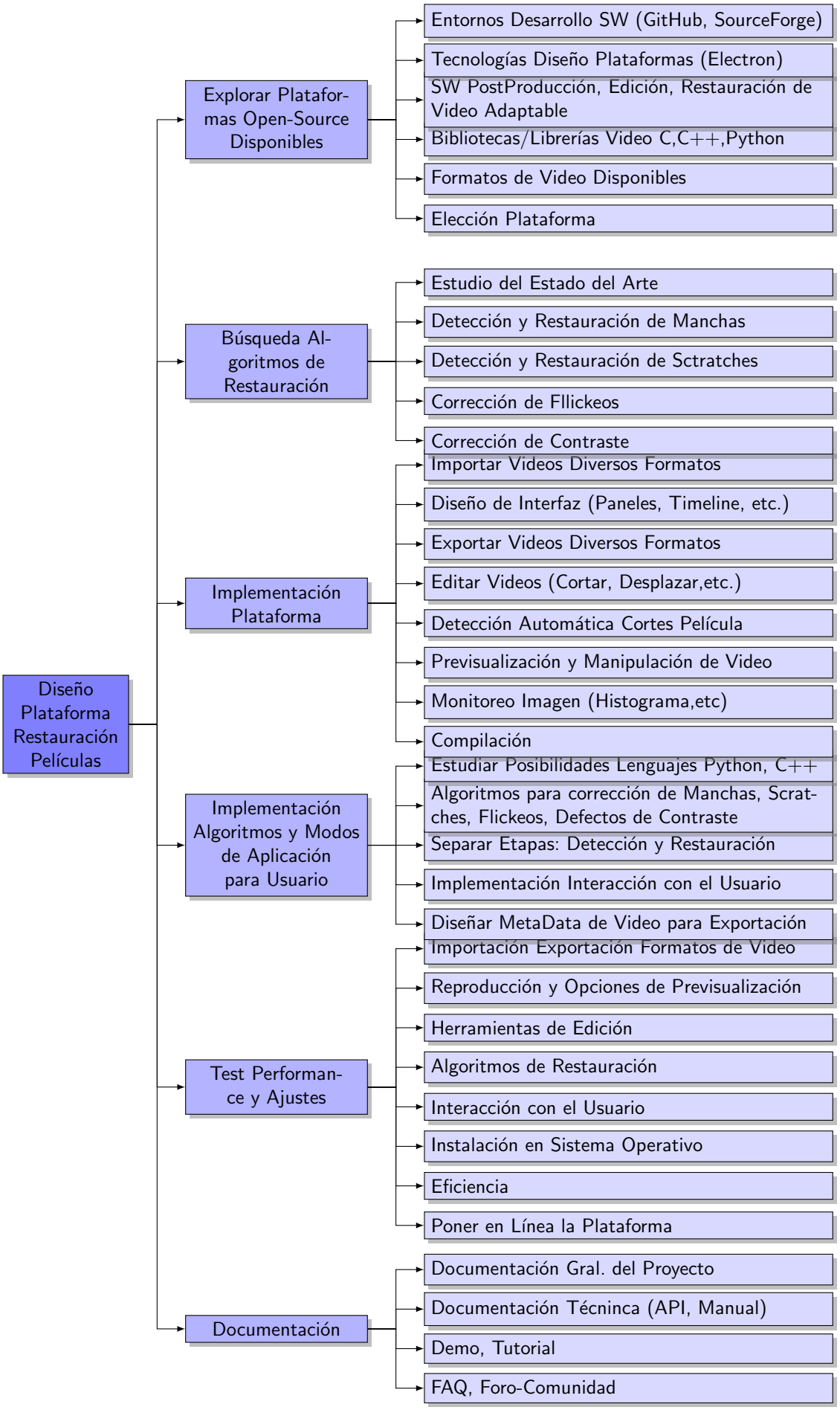
Plataforma Abierta de Restauración de Video			
Importar/ Manipular video	Filtros de restauración	Exportación de video	Documentación técnica
Opciones de importación	Manchas	Formatos estándar. Baja compresión	API
Monitoreo de aspectos técnicos de video	Scratches	Meta Data especial para archivado	Tutorial / FAQs
Herramientas básicas de edición	Flickeos		Foro/Comunidad

10. Objetivos específicos

Entre los objetivos específicos se enumeran los siguientes:

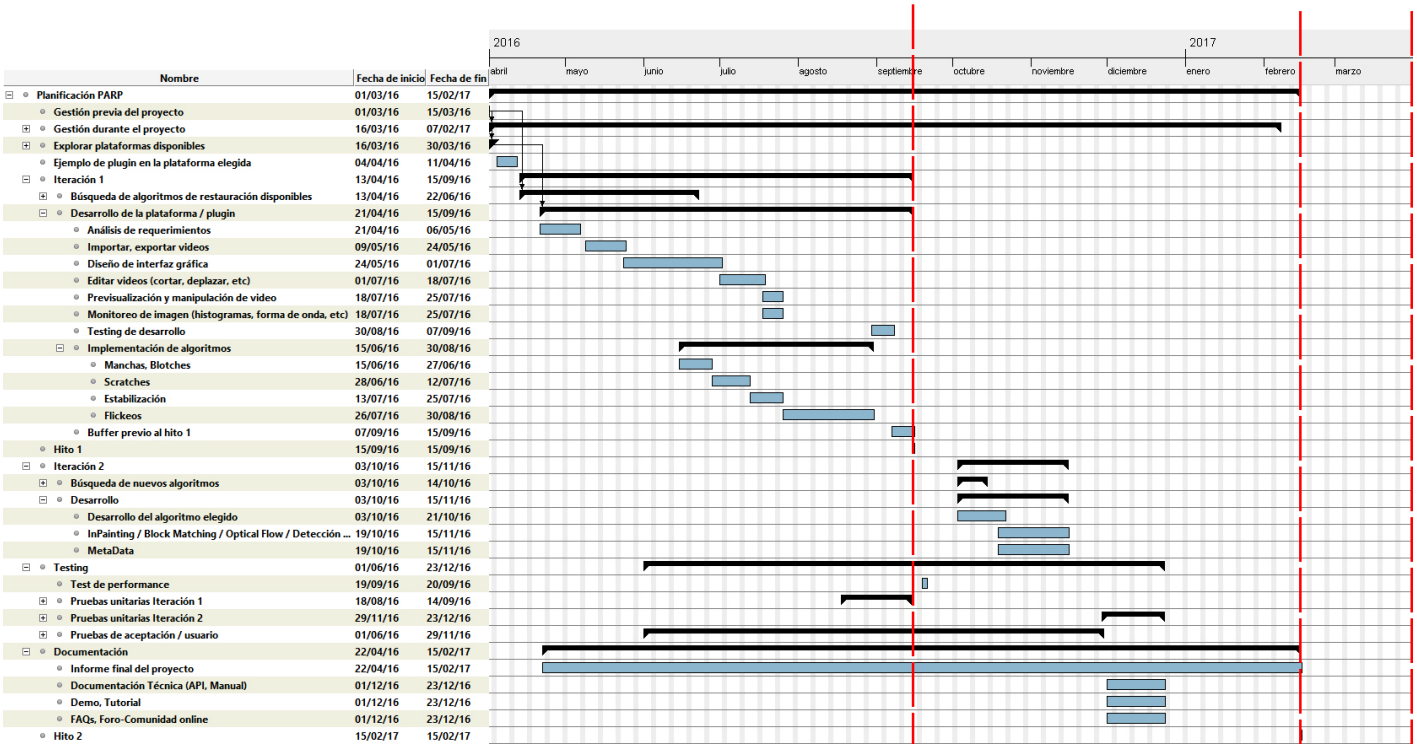
- Elegir una plataforma libre apropiada sobre la cual programar la interfaz y los algoritmos de restauración.
- Poder abordar la restauración de las siguientes degradaciones: *scratches*, manchas, problemas de contraste, *flickeos* de imagen.
- Exportar en formatos de video apropiados en cuanto a resolución, compresión y *metadata*, para un correcto archivado.
- Documentación adecuada que permita la futura incorporación y optimización de nuevos módulos.
- Mantener acotados los tiempos de procesamiento y la eficiencia general del programa.

11. Tareas


Diseño Plataforma Restauración Películas
Explorar Plataformas Open-Source Disponibles
Entornos Desarrollo SW (GitHub, SourceForge)
Tecnologías Diseño Plataformas (Electron)
SW PostProducción, Edición, Restauración de Video Adaptable
Bibliotecas/Librerías Video C,C++,Python
Formatos de Video Disponibles
Elección Plataforma
Búsqueda Algoritmos de Restauración
Estudio del Estado del Arte
Detección y Restauración de Manchas
Detección y Restauración de Sctratches
Corrección de Fllickeos
Corrección de Contraste
Implementación Plataforma
Importar Videos Diversos Formatos
Diseño de Interfaz (Paneles, Timeline, etc.)
Exportar Videos Diversos Formatos
Editar Videos (Cortar, Desplazar,etc.)
Detección Automática Cortes Película
Previsualización y Manipulación de Video
Monitoreo Imagen (Histograma,etc)
Compilación
Implementación Algoritmos y Modos de Aplicación para Usuario
Estudiar Posibilidades Lenguajes Python, C++
Algoritmos para corrección de Manchas, Scratches, Flickeos, Defectos de Contraste
Separar Etapas: Detección y Restauración
Implementación Interacción con el Usuario
Diseñar MetaData de Video para Exportación
Test Performance y Ajustes
Importación Exportación Formatos de Video
Reproducción y Opciones de Previsualización
Herramientas de Edición
Algoritmos de Restauración
Interacción con el Usuario
Instalación en Sistema Operativo
Eficiencia
Poner en Línea la Plataforma
Documentación
Documentación Gral. del Proyecto
Documentación Técninca (API, Manual)
Demo, Tutorial
FAQ, Foro-Comunidad

12. Cronograma detallado del proyecto

En la Figura 1 se puede ver un diagrama de Gantt realizado al principio del proyecto para presentar un cronograma detallado del mismo. En rojo se marcan los hitos uno y dos y la fecha prevista de finalización.



Nombre	Fecha de inicio	Fecha de fin
Planificación PARP	01/03/16	15/02/17
Gestión previa del proyecto	01/03/16	15/03/16
Gestión durante el proyecto	16/03/16	07/02/17
Explorar plataformas disponibles	16/03/16	30/03/16
Ejemplo de plugin en la plataforma elegida	04/04/16	11/04/16
Iteración 1	13/04/16	15/09/16
Búsqueda de algoritmos de restauración disponibles	13/04/16	22/06/16
Desarrollo de la plataforma / plugin	21/04/16	15/09/16
Análisis de requerimientos	21/04/16	06/05/16
Importar, exportar videos	09/05/16	24/05/16
Diseño de interfaz gráfica	24/05/16	01/07/16
Editar videos (cortar, deplazar, etc)	01/07/16	18/07/16
Previsualización y manipulación de video	18/07/16	25/07/16
Monitoreo de imagen (histogramas, forma de onda, etc)	18/07/16	25/07/16
Testing de desarrollo	30/08/16	07/09/16
Implementación de algoritmos	15/06/16	30/08/16
Manchas, Blotches	15/06/16	27/06/16
Scratches	28/06/16	12/07/16
Estabilización	13/07/16	25/07/16
Flickeos	26/07/16	30/08/16
Buffer previo al hito 1	07/09/16	15/09/16
Hito 1	15/09/16	15/09/16
Iteración 2	03/10/16	15/11/16
Búsqueda de nuevos algoritmos	03/10/16	14/10/16
Desarrollo	03/10/16	15/11/16
Desarrollo del algoritmo elegido	03/10/16	21/10/16
InPainting / Block Matching / Optical Flow / Detección ...	19/10/16	15/11/16
MetaData	19/10/16	15/11/16
Testing	01/06/16	23/12/16
Test de performance	19/09/16	20/09/16
Pruebas unitarias Iteración 1	18/08/16	14/09/16
Pruebas unitarias Iteración 2	29/11/16	23/12/16
Pruebas de aceptación / usuario	01/06/16	29/11/16
Documentación	22/04/16	15/02/17
Informe final del proyecto	22/04/16	15/02/17
Documentación Técnica (API, Manual)	01/12/16	23/12/16
Demo, Tutorial	01/12/16	23/12/16
FAQs, Foro-Comunidad online	01/12/16	23/12/16
Hito 2	15/02/17	15/02/17

Figura 1: Primera versión de diagrama de Gantt del proyecto.

Este se modificó con el correr del proyecto, por los cambios de alcance y complicaciones que surgieron. La última versión del Gantt se muestra en la Figura 2.

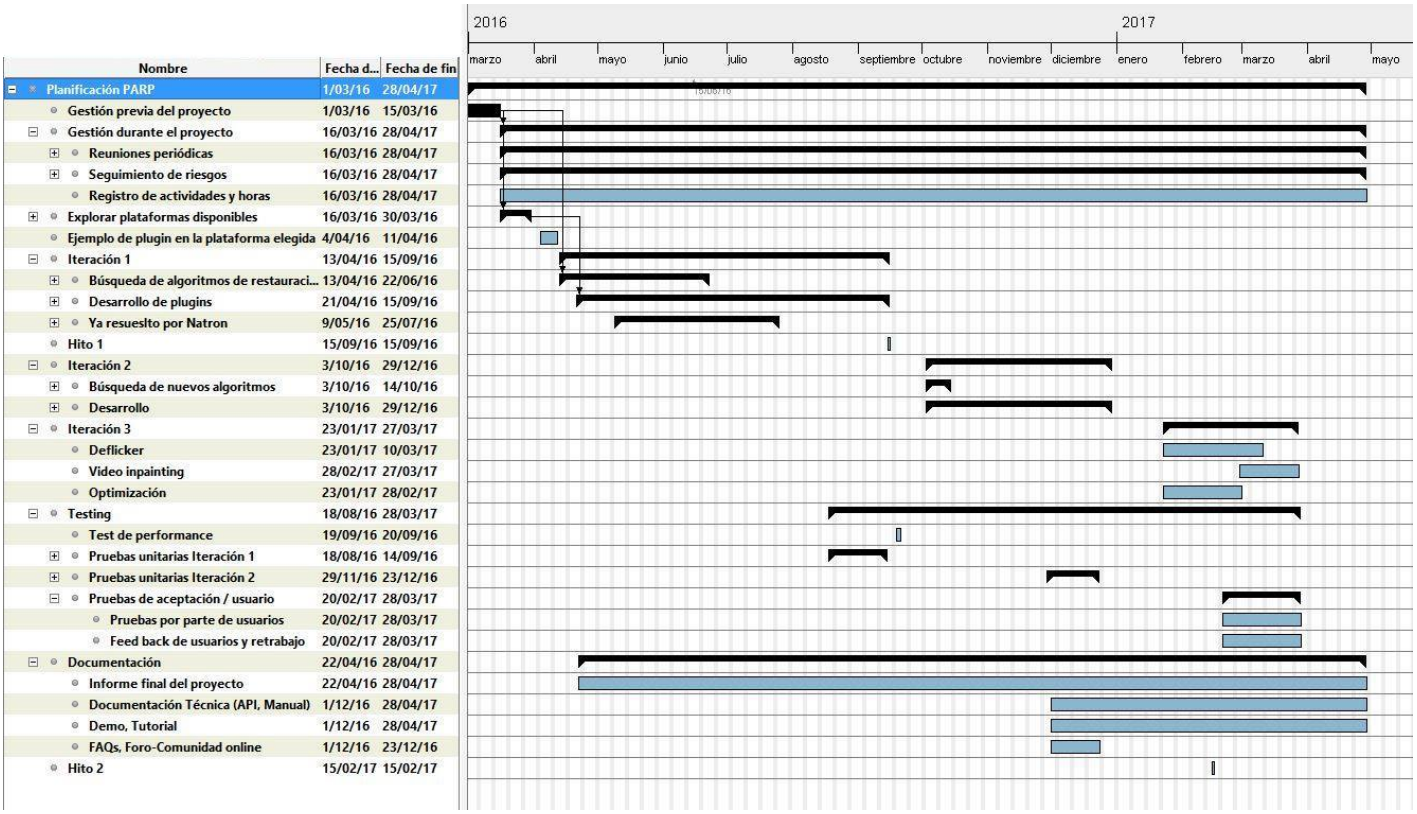


Figura 2: Última versión de diagrama de Gantt del proyecto.

13. Análisis de costos

En este proyecto, no poseemos costos ni de insumos, ni de importaciones o trámites, ni tampoco de servicios. Por lo tanto solo se especifican a continuación los costos en horas hombre. En total, según la primera estimación el costo del proyecto será de 1000 horas hombre. Suponiendo un monto de 200 pesos la hora, el monto total del proyecto asciende a 200.000 pesos.

14. Análisis de riesgos

Se enumeran los siguientes riesgos para nuestro proyecto:

1. Tutor se va de viaje y repercute en el rendimiento del proyecto.
2. Plataforma elegida resulta no apropiada.
3. Tener más exámenes que los planificados.
4. Error en la estimación de tiempos de tareas.
5. No encontrar implementaciones existentes de algoritmos durante la Iteración 1.
6. Un algoritmo presente complicaciones imprevistas para su implementación.
7. Testing de usuarios implica replanificaciones
8. No obtener tiempos de ejecución razonables
9. Actualización de *plugins* a Natron 2.2

Para cada uno de ellos, se especifican acciones a realizar en caso de que ocurran.

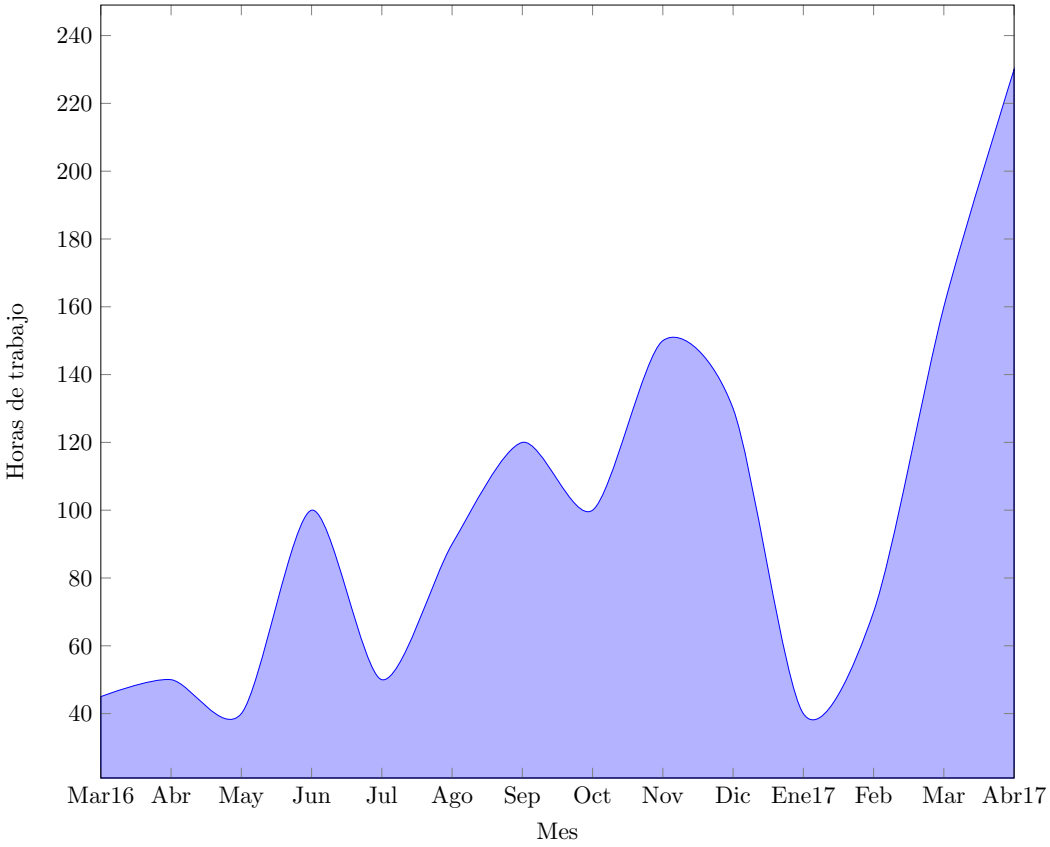
1. Incorporar co-tutor (Mauricio Delbracio) y charlas Skype con tutor
2. Elección de otra plataforma y replanificación. (Por esto se hará un ejemplo de prueba antes del comienzo de la Iteración 1).

3. Se previó que vamos a contar con pocas horas durante estos períodos.
4. Se crearon *buffers* al fin de las grandes tareas para cubrir este riesgo.
5. Buscar otras maneras de implementar el algoritmo.
6. Se elimina este problema y se busca resolver otro que esté dentro del alcance del proyecto.
7. Se crea un *buffer* luego de los tests de usuario planificados para retrabajar.
8. Se intentan optimizar distintas funciones del algoritmo (principalmente las más costosas).
9. En el peor de los casos, si no se logra actualizar los *plugins* al publicarse una nueva versión de Natron, se los publica para la versión anterior.

Finalmente, se realiza una tabla con la estimación final de los riesgos y categoriza según su probabilidad de ocurrencia y nivel de impacto en la realización del proyecto.

		Probabilidad de Ocurrencia		
		Poco probable	Moderado	Muy probable
Nivel de Impacto	Ninguno			
	Bajo		R3, R8, R9	
	Medio	R6	R4, R7	R1, R5
	Alto		R2	
	Extremo			

15. Horas de trabajo



Las horas totales de trabajo en este proyecto fueron 1375.

.