UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

EDVR, super resolución en secuencias de vídeo para su mejora de calidad/restauración

Presentado por Gonzalo Murillo Montes
en Universidad de Burgos — 6 de julio de 2021
Tutores: Dr. Pedro Latorre Carmona y Dr. César Ignacio García Osorio

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va a exponer la planificación temporal, la parte más importante de un proyecto. Indicando las tareas que se realizaron y cuándo se realizaron. También se incluye un análisis de la viabilidad tanto económica como legal del proyecto.

A.2. Planificación temporal

La planificación temporal se ha realizado implementando una metodología ágil, el método *Scrum*.

- El desarrollo del proyecto se ha dividido en *sprints*, con una duración de una semana o dos.
- Los *sprints* contienen las tareas que se llevaron a cabo en el intervalo de tiempo definido.
- Cada tarea tiene asociada un coste aproximado basado en la dificultad o tiempo estimado de realización.
- En caso de que la estimación planificada fuese inexacta, ya sea mayor o menor, se modificó reflejando el tiempo real dedicado.
- Al finalizar cada *sprint* se realizaba una reunión junto a los tutores para explicar las dificultades encontradas, dudas y planificar los siguientes *sprints*.

Sprint 1

Fecha: 17/11/2020 - 1/12/2020

El primer *sprint* consistió en realizar una exploración bibliográfica inicial sobre el estado del arte e instalación en el equipo personal.

Tareas	**Estimado**	**Final**
Exploración bibliográfica inicial	8	10
Instalación EDVR	6	4

Tabla A.1: Tareas del sprint 1

Sprint 2

Fecha: 1/12/2020 - 15/12/2020

Se continuó con la exploración bibliográfica centrándose en los documentos de EDVR. También se estudiaron las posibilidades de instalación ante la incompatibilidad con mi equipo.

Tareas	**Estimado**	**Final**
Continuación del estudio	10	10
Exploración de alternativas	2	2
Instalación de VPN	1	1

Tabla A.2: Tareas del sprint 2

Sprint 3

Fecha: 15/12/2020 - 29/12/2020

Se comenzó la instalación en el equipo remoto y también la descarga de los conjuntos de datos REDS y Vimeo 90K.

Tareas	**Estimado**	**Final**
Instalación en el equipo remoto	8	8
Comienzo de descarga de los conjuntos de datos	5	13

Tabla A.3: Tareas del sprint 3

Sprint 4

Fecha: 29/12/2020 - 12/1/2021

La instalación del conjunto de datos REDS dio más problemas de los esperados y se continuo con su descarga. También con Vimeo 90K descargado, se empezaron a hacer pruebas de ejecución con ese conjunto de datos.

Tareas	**Estimado**	**Final**
Continuación de descarga de REDS	10	10
Primeras ejecuciones	4	6

Tabla A.4: Tareas del sprint 4

Sprint 5

Fecha: 12/1/2021 - 27/1/2021 Se continuo con las ejecuciones con Vimeo 90K y se empieza con REDS.

Tareas	**Estimado**	**Final**
Continuación con Vimeo 90K	4	4
Comienza la ejecución con REDS	4	4

Tabla A.5: Tareas del sprint 5

Sprint 6

Fecha: 27/1/2021 - 3/2/2021 Se continuo con la ejecución de REDS.

Tareas	**Estimado**	**Final**
Continuación con REDS	3	4

Tabla A.6: Tareas del sprint 6

Sprint 7

Fecha: 3/2/2021 - 17/2/2021

Se comenzó a trabajar en un cuaderno de Jupyter, con la instalación de EDVR y explicaciones para entenderlo y ejecutarlo. También se creó un repositorio de GitHub.

Tareas	Estimado	Final
Comienzo cuaderno de demostración Jupyter	5	5
Creación del repositorio en GitHub	2	4

Tabla A.7: Tareas del sprint 7

Sprint 8

Fecha: 17/2/2021 - 24/2/2021

Se continuó con el cuaderno de Jupyter, con más explicaciones y la ejecución. El repositorio de GitHub, se actualizó con la mayoría del trabajo realizado hasta la fecha.

Tareas	Estimado	Final
Continuación con Jupyter	5	5
Creación del repositorio en GitHub	3	4

Tabla A.8: Tareas del sprint 8

Sprint 9

Fecha: 24/2/2021 - 10/3/2021

Se continuó con el cuaderno de Jupyter, con la incorporación del mini conjunto de pruebas. Se empezó a estudiar como implementar un video propio para ser procesado.

Tareas	Estimado	Final
Continuación con Jupyter	6	5
Comienzo video propio	6	6

Tabla A.9: Tareas del sprint 9

Sprint 10

Fecha: 10/3/2021 - 24/3/2021

Se empezaron las pruebas con el video propio. Se añadieron los últimos retoques al cuaderno Jupyter.

Tareas	**Estimado**	**Final**
Pruebas con el video propio	8	12
Añadidos finales cuaderno	4	2

Tabla A.10: Tareas del sprint 10

Sprint 11

Fecha: 24/3/2021 - 31/3/2021

Se continuó con la ejecución del video propio, consiguiendo obtener los resultados deseados.

Tareas	**Estimado**	**Final**
Pruebas con el video propio	8	8

Tabla A.11: Tareas del sprint 11

Sprint 12

Fecha: 31/3/2021 - 14/4/2021

Se estudió la posibilidad de realizar el entrenamiento con videos propios y diferentes técnicas de métricas de clasificaciones de imágenes.

Tareas	**Estimado**	**Final**
Train	3	8
Métricas de clasificación	6	6

Tabla A.12: Tareas del sprint 12

Sprint 13

Fecha: 14/4/2021 - 21/4/2021

Se adquirieron más vídeos con distinto contenido para ser procesados. Se busca un método para convertir las imágenes procesadas a vídeo.

Tareas	Estimado	Final
Más vídeos	5	5
Fotograma a vídeo	4	4

Tabla A.13: Tareas del sprint 13

Sprint 14

Fecha: 21/4/2021 - 28/4/2021

Se empezó a explorar el tema de la interfaz, valorando distintas opciones y decidiéndose por PySimpleGUI. Estudio de la biblioteca.

Tareas	Estimado	Final
Opciones de interfaz	6	4
Estudio PySimpleGUI	8	3

Tabla A.14: Tareas del sprint 14

Sprint 15

Fecha: 28/4/2021 - 5/5/2021

Se comenzó a desarrollar la interfaz, convertir el vídeo en fotogramas, usando Matlab para transformar la imagen y fotogramas a vídeo.

Tareas	Estimado	Final
Opciones de interfaz	6	4
Vídeo a fotogramas	2	2
Matlab	4	7
Fotogramas a vídeo	3	3

Tabla A.15: Tareas del sprint 15

Sprint 16

Fecha: 5/5/2021 - 12/5/2021

Se busca e implementa una alternativa a Matlab. Empieza el estudio sobre cómo usar LaTeX

Tareas	**Estimado**	**Final**
Sustituto Matlab	4	7
Estudio LaTeX	3	3

Tabla A.16: Tareas del sprint 16

Sprint 17

Fecha: 12/5/2021 - 19/5/2021

Se comienza a escribir la memoria del trabajo de fin de grado.

Tareas	**Estimado**	**Final**
Escribir la memoria	6	4

Tabla A.17: Tareas del sprint 17

Sprint 18

Fecha: 19/5/2021 - 9/6/2021

Se continúa escribiendo la memoria del trabajo de fin de grado. Y se empieza con los anexos del trabajo de fin de grado.

Tareas	**Estimado**	**Final**
Continuar escribiendo la memoria	20	20
Empezar a escribir los anexos	9	8

Tabla A.18: Tareas del sprint 18

Sprint 19

Fecha: 9/6/2021 - 19/6/2021

Se continúa escribiendo los anexos del trabajo de fin de grado.

Tareas	**Estimado**	**Final**
Continuar escribiendo los anexos	19	15

Tabla A.19: Tareas del sprint 19

Sprint 20

Fecha: 19/6/2021 - 30/6/2021

Se acaba de escribir los anexos del trabajo de fin de grado. Se realizan un para de figuras para la memoria. Y se retoca el contenido de Git Hub actualizandolo.

Tareas	**Estimado**	**Final**
Fin los anexos	10	12
Figuras	8	8
Git hub	5	5

Tabla A.20: Tareas del sprint 19

Coste total de los *Sprints*

En la tabla A.21 se muestran los costes estimados y finales totales de cada sprint y la suma del coste final del proyecto.

Sprint	**Estimado**	**Final**
Sprint 1	14	14
Sprint 2	13	13
Sprint 3	13	21
Sprint 4	14	16
Sprint 5	8	8
Sprint 6	3	4
Sprint 7	7	9
Sprint 8	8	9
Sprint 9	12	11
Sprint 10	12	14
Sprint 11	8	8
Sprint 12	9	14
Sprint 13	9	9
Sprint 14	14	7
Sprint 15	15	16
Sprint 16	7	10
Sprint 17	6	4
Sprint 18	29	28
Sprint 19	19	15
Sprint 20	23	25
Total	243	245

Tabla A.21: Coste de cada sprint.

A.3. Estudio de viabilidad

Viabilidad económica

Para considerar la viabilidad económica del proyecto se deben calcular los costes derivados de su realización. Se van a tener en cuenta tanto el coste del personal como el del software y el hardware empleados.

Costes de personal

Siguiendo lo expuesto anteriormente el coste personal es:

Concepto	**Coste(€)**
Salario mensual neto	968,63
Retención IRPF (15 %)	256,24
Seguridad Social (28,3 %)	483,46
Salario mensual bruto	1 708,33
Total 7 meses	11 958,31

Tabla A.22: Costes de trabajador.

Costes de software

El único *software* de este trabajo que es de pago es Matlab para estudiantes, en mi caso la Universidad de Burgos nos provee con licencia. En este supuesto se incluirá el precio que se debería pagar por una licencia básica.

Concepto	**Coste(€)**
Licencia Matlab students	69
Total	69,00

Tabla A.23: Costes software.

Costes de hardware

Para el desarrollo de el trabajo no se ha adquirido ningún *hardware*, por lo que se incluye los costos del material con el que ya se contaba, asumiendo una amortización en 5 años y calculando el costo de la amortización en la duración del trabajo, 7 meses.

Concepto	**Coste(€)**	**Coste amortizado(€)**
Dispositivo móvil	150	17,5
Ordenador portátil	800	93,33
Total	950	110,83

Tabla A.24: Costes de *hardware*.

Coste total

Teniendo en cuenta los costes de personal, de *software* y de *hardware*, el coste económico total del proyecto asciende a:

Concepto	**Coste(€)**
Coste de personal	11 958,31
Coste de *software*	69,00
Coste del *hardware*	950,00
Total	**12 977,31**

Tabla A.25: Coste total.

Viabilidad legal

Para el estudio de la viabilidad del producto, se van a analizar las bibliotecas usadas en el proyecto, anotando las licencias de las que hacen uso. A continuación se listan:

- **Zero clause BSD**: Tmux.
- **Modified BSD**: Anaconda, Jupyter Notebook, Spyder, PyTorch.
- **Apache 2.0**: OpenCv.
- **GPLv3**: PySimpleGUI.
- **LGPL**: ffmpeg.
- **Comercial**: Matlab.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este apartado recoge los requisitos y objetivos del proyecto. Se detallarán los objetivos generales y tanto los requisitos funcionales como los no funcionales.

B.2. Objetivos generales

El principal de este proyecto es el estudio de la herramienta de super resolución y restauración de vídeo, implementando mis propios vídeos, así como de la implementación de una interfaz. Este desarrollo se usaría como una herramienta de pre-procesado para técnicas de clasificación de gestos y acciones humanas. Los objetivos generales del trabajo se exponen en la memoria.

B.3. Catalogo de requisitos

Requisitos funcionales

Aquí se enumeran los requisitos funcionales que han sido implementados en el trabajo realizado:

- **RF 1 Obtención de vídeos.** Los videos deben adecuarse a los tamaños soportados 720 x 1280 o 1280 x 720 para vídeos tipo REDS y 448 y 256 o 256 x 448 para vídeos tipo Vimeo.

    - **RF 1.1 Vídeos para la interfaz.** En el caso de la interfaz los videos deben de usarse son los de tipo REDS.

- **RF 2 Procesamiento de videos a mejorar.** Los vídeos deben descomponerse en fotogramas acorde a la estructura de directorios requerida por EDVR.

    - **RF 2.1 Obtención de datos.** Las dimensiónes, el número de fotogramas y la ubicacion de los fotogramas.

- **RF 3 Generación de fichero de configuración.** Para que EDVR pueda ejecutarse es necesario crear y rellenar un fichero que contenga entre otras cosas, la ubicación de las imágenes, si se quiere usar el modo *predeblur*, la ubicación de los modelos preentrenados.

- **RF 4 Recomposición de video procesado.** Los fotogramas procesados deben ser reconvertidos a vídeo para su visualización.

- **RF 5 Reproducción del vídeo.** El vídeo debe poder ser visualizado desde la interfaz.

Requisitos no funcionales

Aquí se enumeran los requisitos no funcionales que han sido implementados en el trabajo realizado:

- **RNF 1 Proceso automático.** Todo el proceso, desde el procesamiento de los vídeos hasta la reproducción de los mismos debe realizarse automáticamente, solo requiriendo hacer clic en un botón.

- **RNF 2 Facilidad de instalación.** La herramienta debe ser fácil de instalar y de puesta en marcha.

- **RNF 3 Usabilidad.** La herramienta debe cumplir estándares de usabilidad, siendo intuitiva y fácilmente utilizable.

- **RNF 4 Software libre.** La herramienta debe requerir del uso de *software* libre.

B.4. Especificación de requisitos

Diagramas de casos de uso

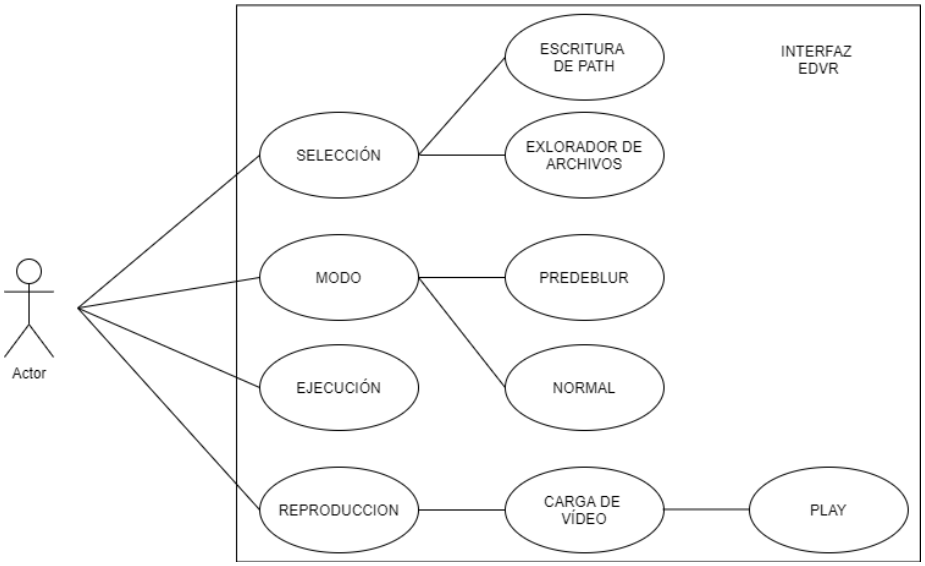

Figura B.1: Diagrama general de los casos de uso

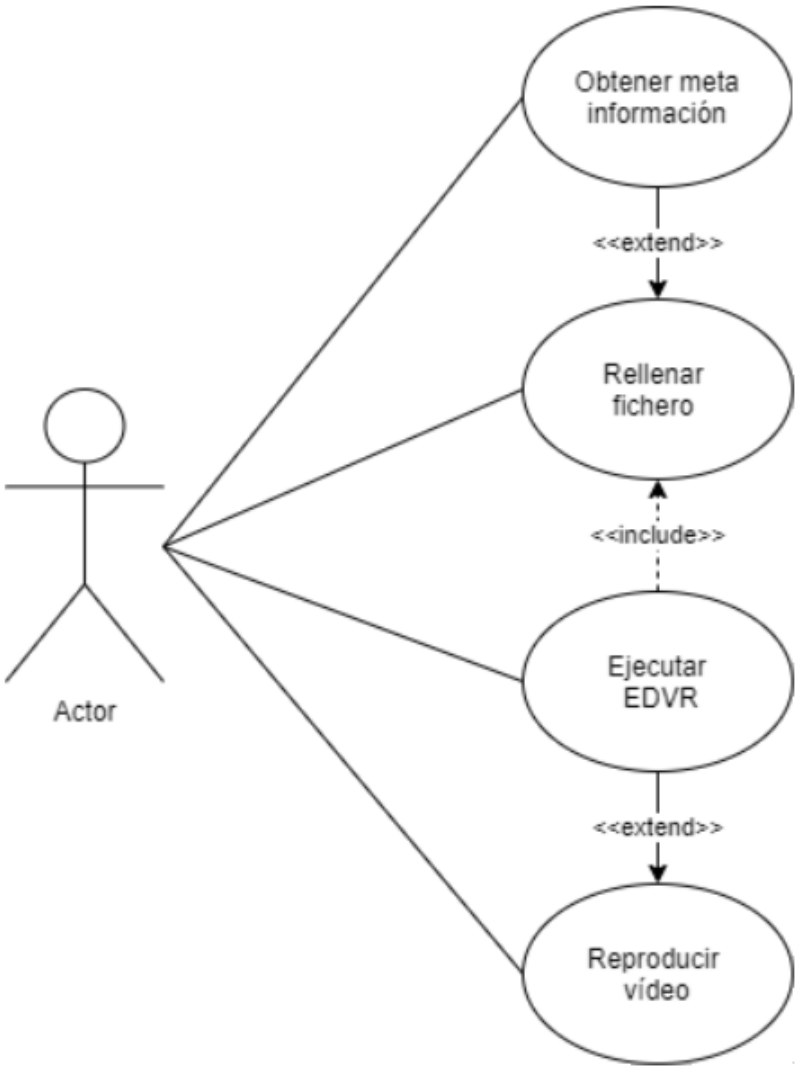

Figura B.2: Diagrama desglosado de los casos de uso del usuario

Caso de uso 1: Obtencion de la información previa.		
Descripción	El usuario obtiene información que se usará en la ejecución.	
Requisitos	RF 1	
	RF 1.1	
Precondiciones	Es necesario disponer de un vídeo para obtener los datos.	
Secuencia normal	Paso	Acción
	1	Obtener las dimensiones del vídeo.
	2	Transformar el vídeo a fotogramas.
	3	Contavilizar el número de fotogramas obtenidos.
	4	Guardar la ubicacion de los fotogramas.
Postcondiciones	La información se ha obtenido.	
Excepciones	No se ha podido obtener toda la información que se requiere.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.1: Caso de uso 1: Obtencion de la información previa.

Caso de uso 2: Inserción de la informacion para el prosesado.		
Descripción	Los dos ficheros necesarios para la ejecución son rellenados.	
Requisitos	RF 1	
	RF 1.1	
	RF 2	
	RF 2.1	
Precondiciones	Es necesario disponer de los datos del paso anterior.	
Secuencia normal	Paso	Acción
	1	Crear los ficheros.
	2	Rellenar los campos de la ubicación.
	3	Rellenar el campo de las dimensiones.
	4	Rellenar el campo del número de fotogramas.
	5	Rellenar el tipo de pretarined model y el campo predeblur.
Postcondiciones	Los ficheros se han rellenado correctamente.	
Excepciones	No se ha rellenado alguno de los campos.	
Importancia	Alta	
Urgencia	Media	

Tabla B.2: Caso de uso 2: Inserción de la informacion para el prosesado.

Caso de uso 3: Ejecución de EDVR.		
Descripción	Los fotogramas son procesados mejorando su calidad.	
Requisitos	RF 1	
	RF 1.1	
	RF 2	
	RF 2.2	
	RF 3	
Precondiciones	Los ficheros de ejecución deben haberse rellendo correctamete.	
Secuencia normal	Paso	Acción
	1	Se indica el PYTHONPATH.
	2	Se indica el numero de GPUs que se usan.
	3	Se indica el fichero pirncipal para la ejecución.
	4	Las imágenes son procesadas.
Postcondiciones	Los fotogramas son super resueltos.	
Excepciones	Error en el procesamiento.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.3: Caso de uso 3: Ejecución de EDVR.

Caso de uso 4: Reproducción de vídeo.		
Descripción	El vídeo procesado se reproduce.	
Requisitos	RF 1	
	RF 1.1	
	RF 2	
	RF 2.2	
	RF 3	
	RF 4	
Precondiciones	Los frames han sido procesados y recompuestos.	
Secuencia normal	Paso	Acción
	1	Cargar el vídeo se carga.
	2	El vídeo se reproduce.
Postcondiciones	El vídeo se reproduce correctamente.	
Excepciones	El vídeo no se reproduce.	
Importancia	Baja	
Urgencia	Baja	

Tabla B.4: Caso de uso 4: Reproducción de vídeo.

Apéndice C

Especificación de diseño

C.1. Introducción

Este apartado se encarga de recoger los diferentes diseños que han sido llevados a cabo para la realización del proyecto y cumplir de forma satisfactoria los requisitos y objetivos anteriormente tratados.

- Primero se hará una breve mención al diseño de los datos, usados por EDVR.
- El siguiente apartado trata sobre la estructura del código usado.
- A continuación se desglosa las funciones que se han implementado.
- Por último, se exponen los prototipos de la interfaz de usuario, previos a el desarrollo de la interfaz.

C.2. Diseño de datos

Para la ejecución de la interfaz, tanto de EDVR por la línea de comandos es requisito indispensable que los vídeos a procesar sean de la siguiente resolución 720×1280 píxeles o viceversa. Para la ejecución por línea también se puede usar 448×256 píxeles o viceversa, cambiando por el modelo pre entrenado correspondiente a Vimeo. Dado que es una resolución no muy común se dejo de trabajar pronto con ella para centrarse en la otra disponible.

Todos los vídeos usados durante el proyecto son capturados con un móvil a resolución 720×1280 píxeles y en formato .mp4. Los fotogramas que se

obtienen a partir de esos vídeos de la resolución correspondiente a los vídeos y en formato .png, al igual que los conjuntos de datos utilizados REDS y Vimeo90K.

Esto es todo respecto desarrollo de la interfaz, pero EDVR interacciona con imágenes, en este manejo de los fotogramas, sucesivas transformaciones y comparaciones, EDVR se basa en la técnica de comparación de fotogramas adyacentes y la fusión de sus características.

Para lograr estos hitos las imágenes son divididas en múltiples matrices de tamaño n, en el ejemplo de tamaño 3:

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Una vez en este formato es sencillo buscar el mismo punto h_{00} en las imágenes adyacentes ${h^n}_{00}$ y comprobar de cuál de ellos se obtiene la mayor información para el proceso de mejora. Esto es una explicación básica de los dos módulos de super resolución que EDVR usa, para más detalles dirigirse a la memoria.

C.3. Diseño procedimental

Todos los eventos que pueden ocurrir durante el desarrollo de la interfaz son concurrentes, es decir pueden ejecutarse en cualquier orden ya que todos están activos en todo momento, pero el orden que se muestra a continuación es el ideal.

```
while No clic Exit o cerrar do
    if Rellenar path a mano then
        Ubicación rellenada
    end
    if Clic Buscar then
        Open Buscador
    end
    if Modo Predeblur True then
        Predeblur=True
    end
    if Modeo Predeblur False then
        Predeblur=False
    end
    if Clic Comenzar then
        Open Confirmar
    end
end
```

Algoritmo 1: Proceso de Selección de vídeo

```
if Clic Cancel o cerrar then
    Volver Selección
end
if Clic Browse then
    Explorador
end
if Clic Ok then
    Ubicación rellenada Volver Selección
end
```

Algoritmo 2: Proceso de Búsqueda de vídeos

```
if Clic Exit o cerrar then
    Volver Selección
end
if Clic Generar then
    Ejecución
end
```

Algoritmo 3: Proceso de Confirmación de ejecución

if *Clic Exit o cerrar* **then**
| Volver Selección
end
if *Clic Si* **then**
| Reprodución
end

Algoritmo 4: Proceso de Ejecución

if *Clic cerrar* **then**
| Volver Ejecucíon
end
if *Clic cargar* **then**
| Cargar vídeo
end
if *Clic paly* **then**
| Reproduce vídeo
end
if *Clic pause* **then**
| Para vídeo
end
if *Clic stop* **then**
| Interrumpe vídeo
end

Algoritmo 5: Proceso de Reproducción del vídeo procesado

C.4. Diseño arquitectónico

La arquitectura de la interfaz es muy sencilla, esta dividida en ocho funciones, de las cuales tres forman la base de la interfaz home, ejecución y reproducir_video. Con esto me refiero a que solo estas tres clases son las que usan código de PysimpleGUI y forman la interfaz, el resto de las funciones contienen código para preparar la ejecución y su posterior procesado.

- **Función home:** Es la primera ventana de la ejecución y la del explorador de archivos. En ella se obtienen los valores de la ubicación del vídeo y del uso o no del modo *predeblur*.

- **Función ejecucion:** La clase que se encarga de llamar a el resto de las funciones que requiere EDVR. Compuesta por la ventana de

confirmación y de progreso de la ejecución. Durante su ejecución obtiene los valores de la dimensión del vídeo y el número de fotogramas.

- **Función hacer_frames:** Función encargada de usar FFmpeg para transformar el vídeo de entrada en fotogramas.
- **Función hacer_LR3:** Función que transforma los fotogramas obtenidos en la anterior a baja resolución.
- **Función ejecutar_EDVR:** Función que crea y rellena los ficheros .txt e .yml necesarios para la ejecución. También lanza la ejecución.
- **Función hacer_video:** Función que partiendo de los fotogramas obtenidos por EDVR, los transforma a vídeo.
- **Función btn:** Función que sirve para definir el tamaño de los botones que se usan en el reproductor.
- **Función reproducir_video:** Función que usa el reproductor VLC para reproducir el vídeo procesado.

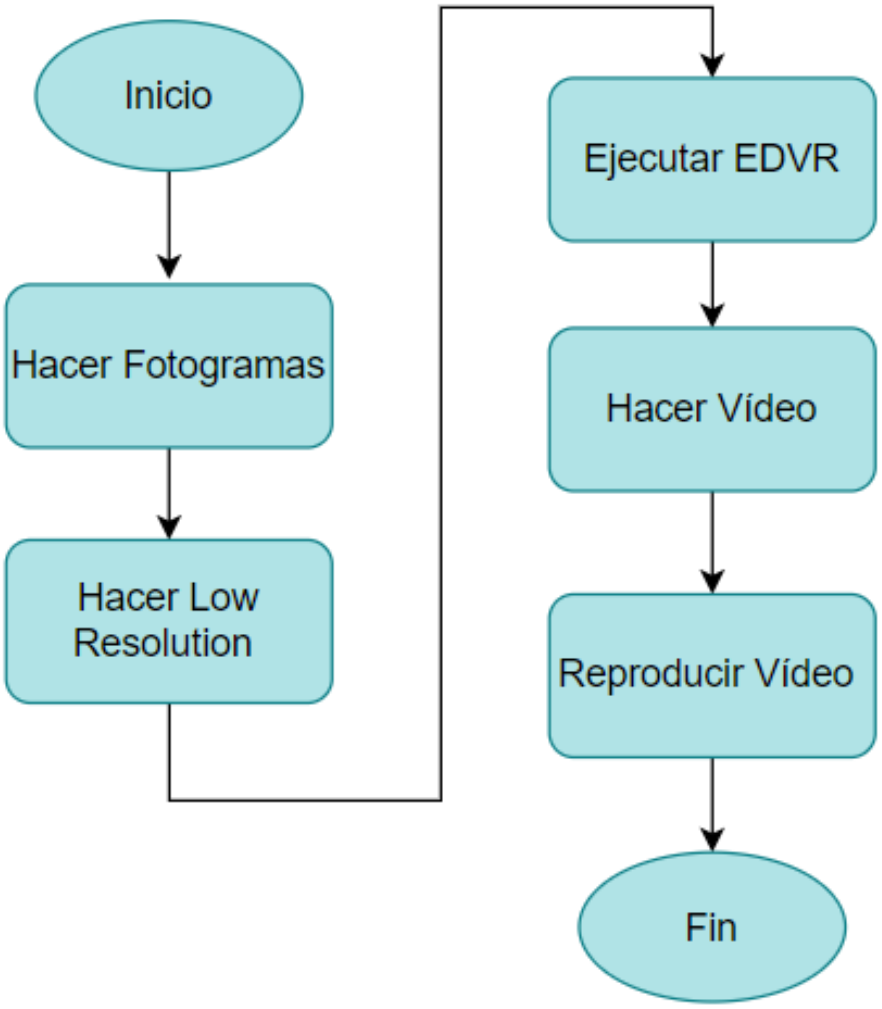

Figura C.1: Esquema de funciones para la ejecución de EDVR

C.5. Diseño de interfaces

Inicialmente se realizó un conjunto de prototipos básicos en los que se plasmaron las funcionalidades que debería tener la interfaz. Los diseños son bastante parecidos a los obtenidos finalmente.

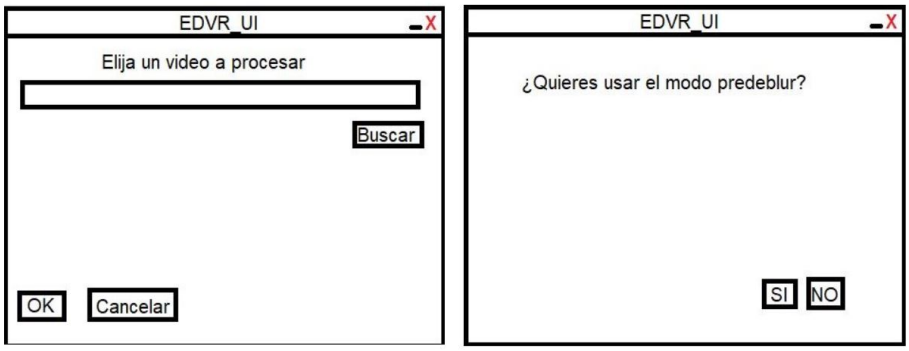

Figura C.2: Prototipos iniciales de las pantallas de seleccion de vídeo a procesar y de uso del módulo de predeblur.

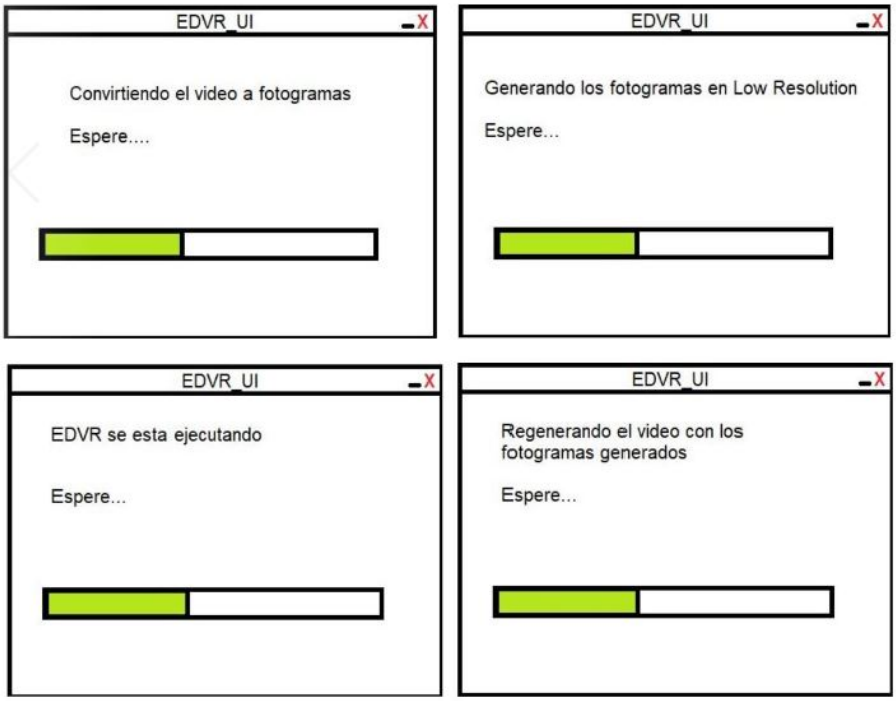

Figura C.3: Prototipos iniciales de las pantallas de carga durante el procesamiento.

Figura C.4: Prototipos iniciales de las pantallas de reproducción del vídeo procesado.

Durante el desarrollo de se decidió usar el tema oscuro "Dark" tema prediseñado en PySimpleGUI, siguiendo los cánones de desarrollo actuales. También durante el desarrollo se tomaron diferentes decisiones en el diseño, dando lugar a los resultados mostrados en la siguiente figura.

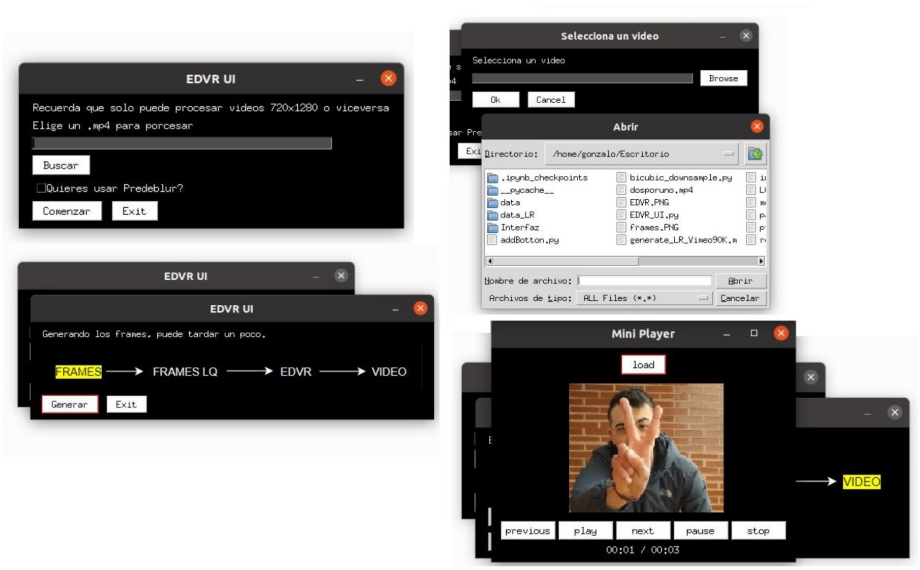

Figura C.5: Interfaces finales.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado se van a exponer todos los conceptos necesarios para comprender la estructura de proyecto, instalar el software necesario para su integración e importarlo en un nuevo equipo para ser ejecutado.

D.2. Estructura de directorios

A continuación, se muestra la estructura de directorios en la que se distribuye el proyecto:

- **\Interfaz:** Carpeta principal del proyecto donde se encuentran todos los elementos relacionados con la interfaz.
    - **\Estados:** Carpeta que contiene las imágenes que se usan en la interfaz para reflejar el estado de la ejecución.
        - **frames.png:** Imagen del proceso de convertir el vídeo a fotogramas.
        - **LQ.png:** Imagen del proceso de transformar los fotogramas a baja calidad.
        - **EDVR.png:** Imagen del proceso de ejecución de EDVR.
        - **video.png:** Imagen del proceso de recomposición de los fotogramas procesados a vídeo.

- **\Resultado:** Carpeta donde se guarda el vídeo procesado.
- **\fotogramas:** Carpeta donde se guardan todos los fotogramas del vídeo.
  - **\originales:** Carpeta donde se guardan los fotogramas originales.
  - **\lr:** Carpeta donde se guardan los fotogramas en baja calidad.
- **EDVR_UI.py:** Código de la interfaz.
- **meta_info_MIO.txt:** Fichero en el que se guardan los datos de: carpeta de localización de los fotogramas, el número de fotogramas y la resolución de estos.
- **test_EDVR_L_x4_SR_Mio.yml:** Fichero que contiene los datos para la ejecución, entre otros la localización todos los fotogramas, el uso o no del modo *predeblur* y el modelo preentrenado usado.
- **EDVR_L_x4_SRblur_REDS_official-983d7b83.pth:** modelo entrenado para los casos que usen el modo *predeblur*.
- **EDVR_L_x4_SR_REDS_official-9f5f5039.pth:** modelo entrenado para los casos que no usen el modo *predeblur*.

- **\ejemplos:** Imágenes que se usan en el *notebook*.

- **\MiniConjunto:** Carpeta con los elementos que usa el *notebook* para la ejecución de EDVR.

  - **\train_blur_bicubic:** Imágenes en baja resolución del conjunto REDS.
  - **\train_sharp:** Imágenes en originales del conjunto REDS.
  - **test_EDVR_L_x4_SRblur_REDS.yml:** Fichero que contiene los datos para la ejecución del mini conjunto.
  - **meta_info_REDS4_test_GT.txt :** Fichero en el que se guardan los datos de los vídeos y fotogramas del mini conjunto.

- **demostración.ipynb:** Cuaderno de *Jupyter* que contiene una explicación de EDVR y un ejemplo funcional de una ejecución de EDVR.

D.3. Manual del programador

A la hora de configurar una ejecución de EDVR son necesarias tres cosas, contar con los fotogramas, tener el fichero .txt con los datos relativos a los fotogramas y tener un fichero. yml con los datos de la ejecución de EDVR. Es en este último archivo donde se pueden configurar más en profundidad las ejecuciones. Durante la demostración y la interfaz que se presentan en este desarrollo no se cambian demasiados aspectos, tan solo los necesarios para hacerse funcionar con los datos de los vídeos proporcionados. Pero se puede ahondar más en la personalización. Tomaremos como ejemplo el archivo situado en el repositorio original link.

```
name: EDVR_L_x4_REDS_SR_official
model_type: EDVRModel
scale: 4
num_gpu: 4
manual_seed: 0

datasets:
  test:
    name: REDS4
    type: VideoTestDataset
    dataroot_gt: datasets/REDS/train_sharp
    dataroot_lq: datasets/REDS/train_sharp_bicubic
    meta_info_file: basicsr/data/meta_info/
    meta_info_REDS4_test_GT.txt
    io_backend:
      type: disk

    cache_data: false
    num_frame: 5
    padding: reflection_circle

# network structures
network_g:
  type: EDVR
  num_in_ch: 3
  num_out_ch: 3
  num_feat: 128
  num_frame: 5
  deformable_groups: 8
```

```
    num_extract_block: 5
    num_reconstruct_block: 40
    center_frame_idx: ~
    hr_in: false
    with_predeblur: true
    with_tsa: true

  # path
  path:
    pretrain_network_g: experiments/pretrained_models/
    EDVR_L_x4_SRblur_REDS_official-9f5f5039.pth
    strict_load_g: true

  # validation settings
  val:
    save_img: true
    suffix: ~

    metrics:
      psnr: # metric name, can be arbitrary
        type: calculate_psnr
        crop_border: 0
        test_y_channel: false
```

Lo primero que podemos modificar es el número de gpus, dependiendo del número de las que se dispongan, también se puede usar un modo cpu poniendo un 0 en este apartado.

La siguiente característica cuya modificación puede ser interesante es el de *type* en el apartado *dataset*, que es la manera de almacenar los fotogramas de los vídeos, hay tres opciones disc, LMDB o memcached.

- **disc:** Los fotogramas se almacenan en el disco duro.
- **LMDB:**(*Lightning Memory-Mapped Database*)usa una técnica para incrementar la velocidad de descompresión de la cpu. Para la realización de los tests no es muy recomendable ya que no suele ser mucha la información a procesar. Requiere generar los ficheros LMBD[1].
- **memcached:** Aporta un mejor rendimiento ya que usa la memoria caché. Requiere que el equipo soporte Memcached[2] y requiere más campos en el fichero.yml:

- server_list_cfg:
- client_cfg:
- sys_path:

Cada uno con sus respectivas localizaciones.

Para la ejecución de EDVR, el apartado *network structures* contiene la información de los parámetros que se usan en la ejecución del algoritmo de super resolución:

Descripción de la estructura de procesamiento interno en EDVR		
Parámetro	**Valor**	**Descripción**
num_in_ch	3	Número de canales de entrada
num_out_ch	3	Número de canales de salida
num_feat	128	Número de canales para las características
num_frame	5	Número de fotogramas de entrada
deformable_groups	8	Número de grupos en el que se dividirá la compensación
num_extract_block	5	Número de bloques para la adquisición de las caracteísticas
num_reconstruct_block	40	Número de bloques para la reconstrucción
center_frame_idx	none	El índice del fotograma central
hr_in	false	Si los frames a procesar son en alta resolución

Tabla D.1: Estructura de procesamiento interno en EDVR

Por último, si se disponen de fotogramas ideales, estos pueden ser usados para calcular dos métricas que ofrecen una calificación cuantitativa de la mejora de los fotogramas mejorados comparados con la imagen ideal. PSNR(Peak Signal-to-Noise Ratio) [3] y SSIM(Structural Similarity Index Measure) [4].

Los fotogramas deben colocarse en el apartado *dataset* en el campo *dataroot_gt*. Los dos resultados de estas métricas se proporcionan al final de la ejecución en el terminal o se pueden consultar en los logs generados de cada ejecución ubicados en la carpeta /results. Cuanto su valor sea más alto mejor será el resultado, esto en cuanto a PSNR y en relación a SSIM cuanto mas cerca este el valor de 1 mejor será el resultado.

D.4. Compilación, instalación y ejecución del proyecto

En este aparatado se indica como instalar el repositorio de EDVR y como ejecutarlo. También se explica cómo instalar la interfaz con todos sus requerimientos.

Instalación

Antes de instalar los repositorios es importante comprobar que se cumplen los requisitos para hacer funcionar EDVR cumpliendo o obteniendo los siguientes requisitos:

- Python $\geq$ 3.7 recomendando Anaconda.
- PyTorch $\geq$ 1.3
- Tarjeta gráfica NVIDIA y CUDA

Una vez cumplidos es importante instalar el repositorio original de EDVR https://github.com/xinntao/EDVR para que posteriormente la interfaz funcione correctamente, ya que si no habrá conflictos con las ubicaciones de los archivos.

Lo primero es clonar el repositorio:

```
git clone https://github.com/xinntao/BasicSR.git
```

Una vez clonado hay que instalar las dependencias con:

```
pip install -r requirements.txt
```

Y por último para instalar completamente todo este comando:

```
python setup.py develop
```

Con estos pasos ya esta EDVR listo para usarse, ahora pasamos a instalar el repositorio de este proyecto. Puede ser clonado en el mismo directorio donde se localiza todo lo descargado e instalado con el repositorio de EDVR.

Para ello usamos el siguiente comando:

```
https://github.com/gonmurillo/TFG_EDVR.git
```

En el cuaderno de Jupyter hay una miniguía de instalación y ejecución, que puede ser usada nada más descargar el repositorio del proyecto.

Las siguientes bibliotecas deben ser descargadas, junto al programa VLC:

```
pip install ffmpeg-python
pip install opencv-python
pip install PySimpleGUI
pip install python-vlc
sudo apt install vlc
```

En nuestro caso se instaló VLC con el comando apt ya que con snap no se instalaba un codec que es necesario. La interfaz y e cuaderno Jupyter están configurados de tal manera que la mayoría de sus necesidades para funcionar están en el repositorio que se proporciona, pero hay ciertos aspectos en los que son dependientes del repositorio, como para la ejecución del algoritmo. Los *paths* solo funcionaran correctamente si se la carpeta del repositorio se coloca con el resto de las carpetas de EDVR. Si no se hace así habría que modificar varias partes del código para la nueva ubicación.

Con todos estos pasos realizados ya se puede ejecutar tanto EDVR de manera normal, desde el cuaderno Jupyter y desde la interfaz.

Para hacer funcionar la interfaz solo en necesario desplazarse hasta la ubicación del archivo y ejecutarlo con:

```
python EDVR_UI.py
```

Ejecución

A la hora de ejecutar EDVR se ofrecen 2 opciones, hacerlo por la consola de comandos, pero realizando los pasos previos de configuración manualmente, o usar la aplicación partiendo solo desde un vídeo.

Para la primera opción, una vez todos los pasos previos están realizados basta con adaptar estas tres líneas de código:

```
PYTHONPATH="./:${PYTHONPATH}"
CUDA_VISIBLE_DEVICES=0
python basicsr/test.py -opt options/test/EDVR/test_EDVR_L_x4_SRblur_REDS.yml
```

Pudiendo personalizar el número de gpus que participarán en la ejecución y cambiando entre los ficheros que contienen los datos de la ejecución(.yml). Si se usa varias gpus la línea que especifica el número y la que ejecuta el archivo de Python cambian, en este ejemplo se usan 4 gpus:

```
PYTHONPATH="./:${PYTHONPATH}" \
CUDA_VISIBLE_DEVICES=0,1,2,3 \
python -m torch.distributed.launch --nproc_per_node=4 --master_port=4321
basicsr/test.py -opt options/test/EDVR/test_EDVR_L_x4_SRblur_REDS.yml
--launcher pytorch
```

La segunda opción, es mucho más sencilla pero no ofrece tanto control sobre la ejecución:

```
python EDVR_UI.py
```

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección se tratará de explicar todo el proceso de utilización de las herramientas que conforman el proyecto. Se tratará de especificar los requisitos técnicos necesarios para el buen funcionamiento, el proceso de instalación, y por último, un manual de uso de las herramientas.

E.2. Requisitos de usuarios

A continuación, se listan los requisitos técnicos necesarios para el correcto funcionamiento del proyecto, hay algunos requisitos que no son obligatorios, pero si recomendables para explorar la totalidad del proyecto:

- **Sistema operativo:** Para todo el desarrollo del proyecto se ha usado Linux, más en concreto Ubuntu, en un principio todas las librerías y programas son compatibles con Windows, EDVR no esta probado en esta plataforma pero no debería haber problemas usando Windows WSL with CUDA supports https://docs.microsoft.com/en-us/windows/win32/direct3d12/gpu-cuda-in-wsl.
- **Tarjeta gráfica Nvidia:** Para que EDVR pueda funcionar es necesario que la tarjeta gráfica que se use sea de Nvidia y compatible con CUDA.
- **Anaconda:** Para la instalación de Python, Anaconda es una opción, es la que recomiendo ya que aporta otras herramientas como Jupyter que se usa en el proyecto.

- **Librerias:** Las librerías son necesarias para tanto EDVR como para la interfaz. Las librerías son:
  - **pytorch**
  - **ffmpeg-python**
  - **python-vlc**
  - **opencv-python**
  - **PySimpleGUI**
- **VLC:** Reproductor de vídeos que es necesario para la interfaz.
- **Repositorios:** Los dos repositorios son imprescindibles tanto el de EDVR https://github.com/xinntao/EDVR como el propio https://github.com/gonmurillo/TFG_EDVR/tree/main.

E.3. Instalación

En este apartado se detalla el proceso de instalación necesario para el correcto funcionamiento del proyecto:

- **Python:** Para instalar Python como ya expuse antes lo ideal es instalar Anaconda, para ello nos dirigimos a la web https://www.anaconda.com/products/individual#linux y descargamos el instalador. Una vez descargado nos desplazamos a la carpeta y ejecutamos el siguiente comando.

  ```
  sh Anaconda3-5.0.0.1-Linux-x86_64.sh
  ```

  Tras aceptar la licencia y configurar algunos aspectos ya tenemos Python en el equipo.
- **PyTorch:** La instalación de PyTorch se realiza desde la siguiente web https://pytorch.org/, aquí seleccionamos las características de nuestra máquina y uso que le daremos y nos proporciona el comando adecuado para la instalación.

Figura E.1: Instalación de PyTorch usada en mi equipo.

- **CUDA:** La intsalacion de CUDA es muy parecida a la de PyTorch y se hace desde el siguiente link `https://developer.nvidia.com/cuda-downloads`, se seleccionan las características del equipo y las deseadas y se proporciona el comando para la instalación.

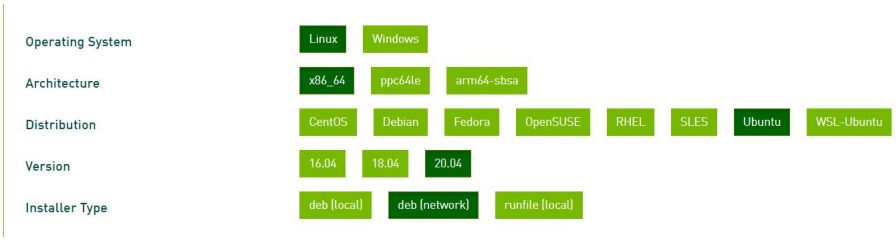

Figura E.2: Instalación de CUDA usada en mi equipo.

Para el resto de bibliotecas, programas y repositorios, la instalación se encuentra en el apartado instalación de la documentación técnica de programación.

E.4. Manual del usuario

Este apartado se dividirá en dos secciones, la primera dedicada a usar EDVR sin la interfaz, y la segunda usando la interfaz.

EDVR por línea de comandos

La ventaja de este método frente a la interfaz, es que puedes procesar varios vídeos a la vez, eso sí, con un incremento en el tiempo de ejecución, cuantos más fotogramas, más tiempo llevará.

Asumimos que el usuario ya tiene un vídeo dividido en fotogramas teniendo también los fotogramas en baja resolución. Los ficheros creados se guardan dentro del directorio BasicSR. Se usará la siguiente estructura de carpetas para no inducir a la confusión.

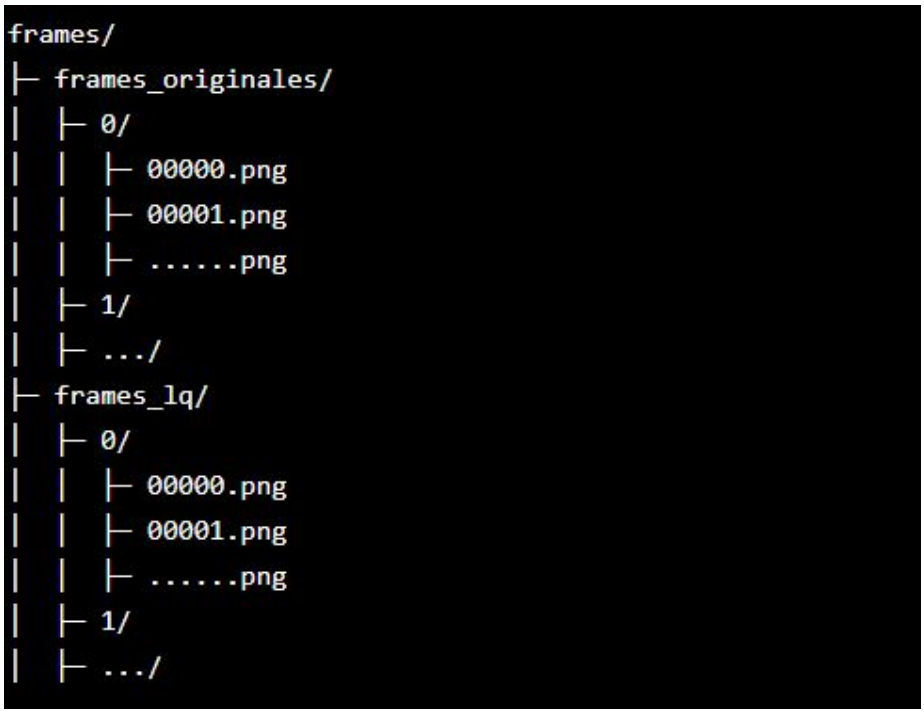

Figura E.3: Esquema de la estructura de carpetas para los fotogramas.

El primer paso es crear el fichero meta_info.txt de metadatos del vídeo o vídeos, siendo el primer dato la ubicación, el segundo el número de fotogramas y el tercero la resolución junto a 3 si es una imagen a color o 2 si es en blanco y negro.

```
000 208 (1280,720,3)
001 305 (720,1280,3)
002 70 (1280,720,3)
...
```

El siguiente paso es rellenar el fichero EDVR.yml con los datos para la ejecución. Tomamos como referencia el fichero del manual del programador cambiamos los campos de:

```
  dataroot_gt: frames/frames_originales
  dataroot_lq: frames/frames_lq
  meta_info_file: meta_info.txt
```

Ya solo tenemos que ejecutar EDVR, desde la carpeta principal de BasicSR:

```
PYTHONPATH="./:${PYTHONPATH}"
CUDA_VISIBLE_DEVICES=0
python basicsr/test.py -opt EDVR.yml
```

EDVR con interfaz

A la hora de usar la interfaz es mucho más sencillo ya que solo hay que desplazarse hasta la carpeta Interfaz y ejecutar el siguiente comando:

```
Python EDVR_UI.py
```

Todos los pasos son muy sencillos e intuitivos, dependiendo de la capacidad de cómputo y la duración del vídeo tardará más o menos. Los fotogramas procesados y el *log* se encuentra en el mismo lugar, la carpeta *results* y el vídeo reconstruido en la carpeta Resultado dentro de Interfaz.

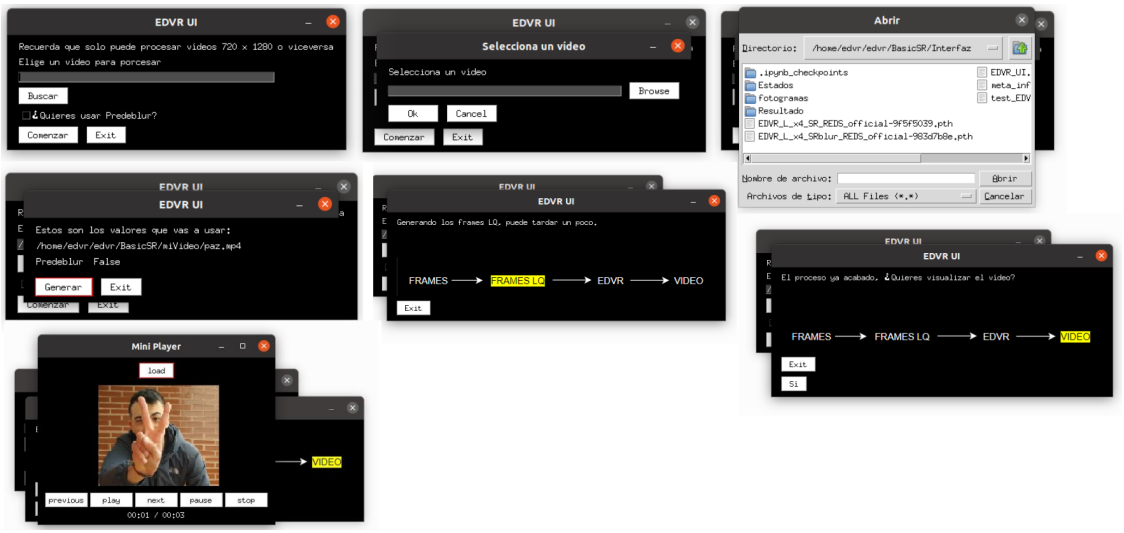

Figura E.4: Pestañas de la interfaz durante la ejecución.

Bibliografía

[1] Martin Hedenfalk. Lightning memory-mapped database manager (lmdb), 2010. [Internet; descargado 10-junio-2021].

[2] Memcached. Memcached, 2018. [Internet; descargado 10-junio-2021].

[3] Wikipedia. Psnr — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 11-junio-2021].

[4] Wikipedia. Structural similarity — wikipedia, the free encyclopedia, 2021. [Internet; descargado 11-junio-2021].