Escola Tècnica Superior d'Enginyeria ETSE-UV

GRADO EN INGENIERÍA MULTIMEDIA

TRABAJO DE FIN DE GRADO

QUALIAPP360: APLICACIÓN PARA EVALUAR LA CALIDAD DE LOS VÍDEOS 360

AUTORA:
ESTEFANÍA ESCUDERO ESCOBEDO

TUTORÍA:
MIGUEL GARCÍA PINEDA

SEPTIEMBRE, 2021

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

GRADO EN INGENIERÍA MULTIMEDIA

TRABAJO DE FIN DE GRADO

QUALIAPP360: APLICACIÓN PARA EVALUAR LA CALIDAD DE LOS VÍDEOS 360

AUTORA:
ESTEFANÍA ESCUDERO ESCOBEDO

TUTORÍA:
MIGUEL GARCÍA PINEDA

TRIBUNAL:

PRESIDENTE/PRESIDENTA:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Agradecimientos

En primer lugar, quiero dar las gracias a mi tutor Miguel por haberme ayudado a desarrollar el Trabajo Final de Grado correctamente. Fue muy claro en lo que debía enfocarme a la hora de diseñar la aplicación y me facilitó los recursos que más tarde utilizaría en todas las fases de este proyecto.

Aparte quiero agradecer a los investigadores y estudiantes que han dedicado su tiempo y trabajo en obtener datos y algoritmos utilizados en la tecnología de vídeos 360º. Sin su trabajo no podría haber desarrollado este proyecto, ya que gracias a ellos he podido tomar distintas métricas y aplicarlas en mi aplicación.

Quiero dar las gracias también a la Universidad de Valencia (UV) y a la Escuela Técnica Superior de Ingeniería (ETSE) por ofrecer los medios e instalaciones para la obtención del conocimiento que me ha permitido llegar hasta donde estoy. Igualmente, agradezco a los distintos profesores que se han preocupado por el progreso de mi persona desde estudiante a ingeniera multimedia. En especial al profesor Vicente Fco Candela por animarme a no tirar la toalla y alentarme a conseguir pasar el bache por el que estaba pasando en esos años.

Finalmente, quiero dar las gracias a mi familia, a mis amigos y en especial a mi pareja por el apoyo incondicional.

Resumen

El uso y visualización de vídeos esféricos ha ido en aumento con el paso del tiempo a través de la llegada de dispositivos HMD (*Head-Mounted Display*) para el público. Aún siendo una nueva tecnología no asentada para la población general poco a poco se va formando un notable número de usuarios que están adoptando esta tecnología y que demandan contenido multimedia en 360º de calidad.

El presente documento plantea la investigación de distintas métricas para evaluar la calidad de vídeos 360º basándose en la comparativa de similitud entre dos vídeos con la misma resolución.

Tras la exposición del marco teórico sobre el ámbito de los vídeos omnidireccionales y las métricas VQA, haciendo hincapié en las métricas objetivas, se seguirá una planificación establecida apropiada al desarrollo de una aplicación en Python.

Se seguirá un método de actuación de ingeniería de software para conceptualizar todo el proceso y obtener una visión global de funcionamiento del proyecto.

Como punto de arranque del proyecto, la comprensión y selección de las métricas VQA objetivas a implementar tomaron un papel importante. Se han definido seis métricas de evaluación de calidad a implementar en esta aplicación: PSNR, SSIM, MSSIM, VMAF 360º, S-PSNR, y WS-PSNR. Con estas métricas podemos evaluar vídeos esféricos obteniendo datos precisos de la calidad objetiva.

Como conclusión, en lo que trata a resultados, esta aplicación genera una gráfica con la puntuación obtenida de las métricas seleccionadas. Además, se puede descargar un archivo con las puntuaciones en formato CSV que permite conocer en detalle la comparativa de la calidad de los archivos de vídeo. La extracción de este archivo CSV permite su uso posterior en trabajos de investigación y en un futuro no muy lejano en centros educativos.

Abstract

The use and display of spherical videos has been increasing over time through the arrival of HMD devices (Head-Mounted Display) for the public. Although this new technology is still not established for the general population, step by step, a notable number of users are adopting this technology and demand quality 360º multimedia content.

This document proposes the investigation of different metrics to evaluate the quality of 360º videos based on the comparison of similarity between two videos with the same resolution.

After the presentation of the theoretical framework in the field of omni-directional videos and VQA metrics, with an emphasis on objective metrics, we will follow a proper planning to develop a Python application.

A software engineering method of action will be followed to conceptualize the entire process and obtain a global vision of the project.

As the starting point of the project, the understanding and selection of the objective VQA metrics played an important role. Six quality evaluation metrics have been defined to implement in this application: PSNR, SSIM, MSSIM, VMAF 360º, S-PSNR, and WS-PSNR. With these metrics we can evaluate spherical videos obtaining precise data of the objective quality.

In conclusion, in terms of results, this application generates a graph with the score obtained from the selected metrics. In addition, a file can be downloaded with the scores in CSV format that allows you to know in detail the comparison of the quality of the video files. The extraction of this CSV file allows its later use in research work and in educational centers in the near future.

Palabras clave

- 360VR
- Video omnidireccional
- Calidad
- Codificación
- Evaluación calidad visual
- Aplicación

Keywords

- 360VR
- Omnidirectional video
- Quality
- Video encoding
- Visual Quality Assessment
- Application

Índice

Agradecimientos **2**

Resumen **4**

Abstract **5**

Palabras clave **6**

Keywords **6**

Introducción **11**
Introducción 11
Motivación 13
Objetivos 13
Estructura del documento 14

Estado del arte **15**
Conexión entre vídeos e imágenes 360º y VR 15
HMD para la obtención de datos 16
Problemas generales 16
VQA: Evaluación de calidad visual 17
VQA Objetiva 17
PSNR: Peak Signal-to-Noise ratio 17
SSIM y MSSIM: Structural Similarity Index Measure 18
S-PSNR: Spherical PSNR 21
WS-PSNR: Weighted Spherical PSNR 23
VMAF: Video Multimethod Assessment Fusion 24
VQA Subjetiva 24
Enfoque I: Estimación de movimiento 24
Enfoque II: Corrección de densidad de muestreo 25
Aplicaciones existentes 26
Con interfaz 26
Sin interfaz 27
Conclusiones 28

Planificación **29**
Análisis de Requisitos 29
Requisitos funcionales 29
Requisitos no funcionales 33
Análisis de casos de uso 34
Diagrama de Casos de Uso 34

Especificación de Casos de Uso 35
Diagrama de clases 40
Diagrama de secuencia 42
Metodología en cascada 44
Tareas e hitos 45
Estimación temporal 45
Estimación de costes 47
Costes del personal 47
Costes de software 48
Costes de hardware 49
Costes totales 49
Estudio de viabilidad 49
Viabilidad técnica 50
Viabilidad estratégica 50
Viabilidad legal 50
Viabilidad económica 50
Análisis de riesgos 51
Estrategias de mitigación 52
Planes de contingencia 52
Desarrollo del proyecto 53
Tecnologías utilizadas 53
Visual Studio Code 53
GitHub 53
Visual Paradigm 54
Python 54
Librería OpenCV 55
Módulo NumPy 56
Módulo Pyplot 56
Módulo Interfaz: Tkinter 56
Módulo os 56
Módulo glob 57
Módulo re 57
Diseño 58
Interfaz gráfica 58
Implementación 63
Funciones importantes 63
getPSNRdata 63
getMSSIMdata 65
getSSIMdata 66

getVMAFdata 66
getSPSNRdata 68
getWSPSNRdata 69
Obstáculos en el desarrollo 72

Pruebas y resultados 72
Comparativa entre métricas 72
Puntuaciones obtenidas 72
Análisis de dichas métricas 76
Pruebas de rendimiento 76
Uso de CPU y de memoria 76
Pruebas de usabilidad 78
Test SUS 78
Resultados 78

Conclusiones 80
Conclusiones 80
Trabajo futuro 81

Anexos 82
Anexo I: Código fuente y ejecutable 82
Anexo II: Icono de la aplicación 82
Anexo III: Índice de tablas 83
Anexo IV: Índice de ilustraciones 84
Anexo V: Índice de ecuaciones 85

Bibliografía 86

1. Introducción

1.1. Introducción

En la actualidad la demanda de vídeos 360º, también llamados vídeos esféricos o vídeos inmersivos, se está incrementando gracias a la realidad virtual y los dispositivos que emplea para su funcionamiento como los HMD (Head-Mounted Display), se muestra un ejemplo en la ilustración 1.

Ilustración 1: Primera generación de HMD, Oculus Rift DK1. Fuente: xinreality

El consumidor medio se interesa cada vez más en el entretenimiento interactivo y novedoso, por ello cada vez hay más usuarios interesados en la tecnología de realidad virtual. Esto se debe a que por medio de los vídeos omnidireccionales se consigue revolucionar el entorno de visualización de contenido multimedia creando una experiencia totalmente diferente a lo anteriormente concebido.

Uno de los aspectos a la hora de mejorar la experiencia de usuario es la evaluación de la calidad de vídeo. Estas pruebas son necesarias debido a que si las imágenes que se muestran son de baja calidad se puede crear incomodidad o molestia en la vista como se aprecia en la ilustración 2. Incluso en casos más extremos pueden originarse mareos y náuseas. Y por ende, el consumidor deja de visualizar el vídeo y desarrolla una opinión desfavorable.

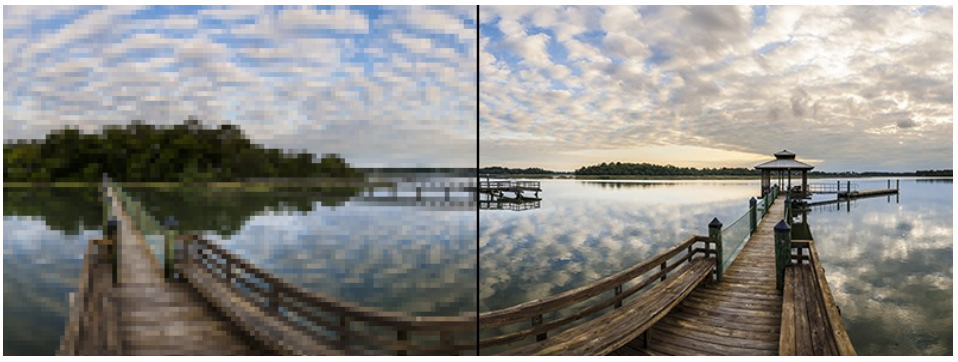

Ilustración 2: Comparativa de calidad entre dos vídeos 360. Fuente: medium

Otro factor a tener en cuenta es el tamaño de los vídeos esféricos. Se necesita transferir archivos muy pesados para su uso ya que la resolución de estos ficheros suele ser de 4K mínimo debido a la bajada de calidad de la imagen original a la que se muestra por el *viewport* del HMD del usuario. Para que un usuario visualice un vídeo 4K a través de su HMD, el vídeo original debería tener 16K de resolución. Creando consigo un tamaño excesivo por fichero. Esto se puede observar en la ilustración 3.

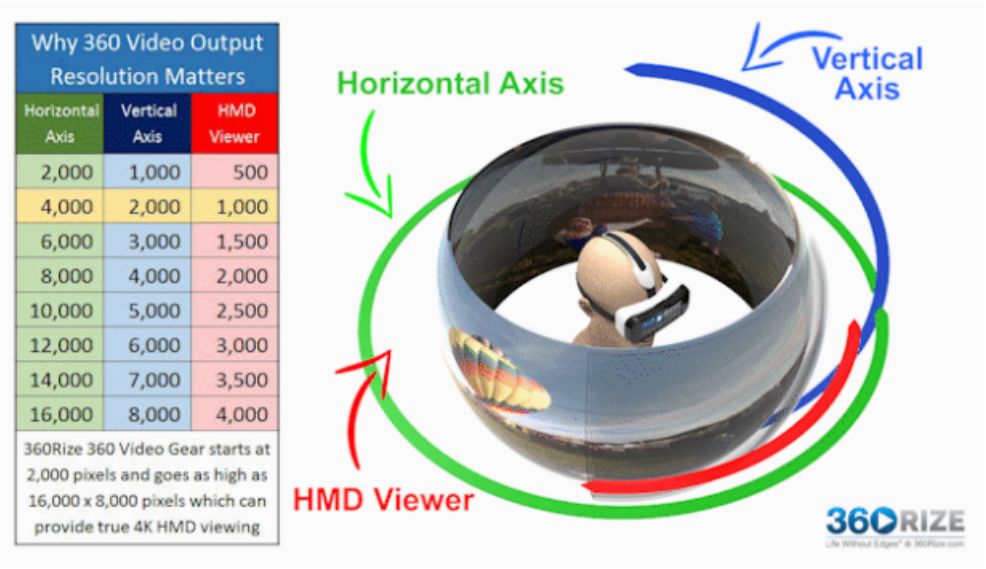

Ilustración 3: Transformación de resolución del vídeo al HMD. Fuente: 360rize

Una solución a este coste temporal de transferencia puede ser el *streaming* o transmisión de vídeo pero surge el problema de la bajada de calidad a causa del ancho de banda de la red. Además, debe existir un equilibrio entre la calidad de vídeo, la solidez frente a la aparición de errores o la pérdida de datos a través de la transmisión, la tasa de bits, y la posible complejidad de decodificación mediante algoritmos.

Frente a esto es notable la importancia que tiene el análisis de vídeos para evaluar la calidad y evitar proporcionar al usuario una mala experiencia causada por la baja calidad de imagen.

Pero dado que es una tecnología reciente surge un problema al no existir herramientas para tratar vídeos esféricos. Por lo tanto, este proyecto se va a basar en crear una aplicación que proporcione una solución a la inexistencia de programas para analizar la calidad de vídeos omnidireccionales. Suplir ese vacío con el desarrollo de la aplicación podría aportar información práctica a las investigaciones para un mejor avance.

1.1. Motivación

A la hora de elegir el tema del TFG siempre tuve en mente que estuviera relacionado con el uso de diferentes elementos multimedia. Me llamó la atención el tema “Video encoding y métricas QOE” que el tutor hizo disponible y tras una reunión inicial se definió que basaría mi TFG en una aplicación para vídeos 360º.

Actualmente los vídeos omnidireccionales son una tecnología en crecimiento, y poder aportar y contribuir con este proyecto en las fases tempranas de su desarrollo me pareció muy interesante.

Aunque la documentación sea limitada, existen métricas para la evaluación de estos vídeos esféricos. O en caso de no existir en el lenguaje seleccionado o para vídeos 360º, se puede realizar una conversión desde otros proyectos existentes. Por lo que, aunque utilice algoritmos y métricas ya estudiadas y definidas, la aplicación es original y de las primeras en ser desarrolladas para vídeos omnidireccionales.

En una vista general tiene la misma estructura de funcionamiento que otras aplicaciones de evaluación de videos. La diferencia radica en los algoritmos usados para la obtención de métricas de calidad, en los archivos que trata, y su diseño de interfaz.

Esta aplicación servirá no solo para investigar sobre la calidad de vídeos en 360º sino también para mejorar las propias habilidades del lenguaje elegido.

1.2. Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación que recoja diferentes métricas para la evaluación de la calidad de vídeos 360º. Seleccionando un vídeo de referencia y su versión distorsionada se deberían obtener datos comparativos sobre la calidad de imagen.

Para su desarrollo se escogió el lenguaje Python por su gran importancia en la actualidad, sus librerías especializadas en vídeo 2D y 3D, y su extenso uso en empresas informáticas, además de ser un lenguaje que se ha aprendido en la carrera.

Como objetivo secundario, la herramienta debería presentar una interfaz amigable e intuitiva con la finalidad de agilizar su uso y evitar errores.

1.3. Estructura del documento

Se ha estructurado este documento en 7 capítulos. A continuación se definen los capítulos del presente trabajo y la respectiva descripción de su contenido.

Capítulo 1: Introducción, motivación y objetivos.
Se introduce el tema y los datos del estudio de forma clara y concisa para dar contexto al lector. También se exponen las razones por las que se ha decidido realizar este trabajo, que se pretende con ello, que métodos se han utilizado y qué resultados se esperan obtener.

Capítulo 2: Estado del arte.
Se engloban los conceptos fundamentales para entender el contexto de la aplicación a desarrollar. Se consigue a través de una justificación del entorno de desarrollo y librerías utilizadas además de la explicación teórica del estado actual de la tecnología de los vídeos 360º. Se compara con aplicaciones similares y se concretan las mejoras respecto a estas.

Capítulo 3: Especificación.
Una vez se han definido los objetivos a alcanzar, se determina el diseño y las características que tiene la aplicación de codificación de vídeos 360º.

Capítulo 4: Desarrollo del proyecto.
Se explica detalladamente la planificación y el desarrollo de la aplicación, definiendo los recursos y procesos necesarios para su elaboración.

Capítulo 5: Pruebas y resultados.
Tras la finalización de la aplicación, se debe someter a prueba para observar la mejora de calidad del vídeo comprimido.

Capítulo 6: Conclusiones y trabajo futuro.
En último término se resume el proceso de diseño y desarrollo de la aplicación y se determinará su utilidad en el campo de la realidad virtual a través de los beneficios o avances que puede originar por ser un ejercicio práctico.

2. Estado del arte

2.1. Conexión entre vídeos e imágenes 360º y VR

El continuo crecimiento de la tecnología VR ha derivado en la unión natural entre los vídeos esféricos y la realidad virtual. [1] Gracias a los vídeos e imágenes 360º se consigue obtener una experiencia más rica e inmersiva para el usuario frente a cualquier vídeo/imagen en 2D. Este nuevo tipo de multimedia se define como una esfera omnidireccional con un rango de 360 - 180 grados en el que el usuario puede moverse con total libertad.

Este contenido visual se divide en diferentes regiones completamente imperceptibles para el ojo humano. La utilidad de estos campos reside en determinar dónde se sitúa la vista del usuario y saber en qué región se puede enviar los datos con alta resolución.

La evaluación del movimiento de los ojos también se conoce como *eye-tracking.* Este procedimiento se realiza observando la actividad ocular en la pupila o en los reflejos de la córnea y se refleja en la región de interés (*region-of-interest, RoI*) dentro de la ventana gráfica o *viewport*.

La principal diferencia de los vídeos tridimensionales frente a los vídeos 2D reside en el mecanismo de atención visual innato de los humanos. Para los vídeos bidimensionales hay una tendencia a fijar la vista en el centro de la pantalla. Mientras que con los vídeos en 3D hay una variación, se sigue manteniendo la inclinación a mantener el interés en el ecuador del vídeo.

Aún con la propensión a mantenerse en el centro o en el ecuador del medio, se pueden recoger datos sobre qué parte del contenido es más atractivo a la audiencia a través de movimientos de cabeza. Con estos datos se permite conocer en qué posición y en qué elemento se está concentrando la persona.

Por lo tanto, se necesita predecir tanto el movimiento de cabeza como el ocular para predecir la atención visual y la forma de percibir el entorno. Aunque hay que tener cuidado ya que los resultados de la cabeza y oculares pueden no ser idénticos. El uso de estos dispositivos posibilita la mejora total de la inmersión del usuario en los vídeos 360º, a través del movimiento de la cabeza, simulando la vida real.

2.2. HMD para la obtención de datos

Para llevar a cabo la recogida de datos de movimiento, se pueden utilizar dispositivos de realidad virtual como el *Head-mounted display* (HMD).

Al estar basado en la postura del usuario, las acciones realizadas con la cabeza y los movimientos generales pueden ser grabados con el kit de desarrollo de software (SDK).

Por otro lado, existen dos problemas que surgen a partir del uso de HMD: La bajada de calidad de la ventana gráfica es más notable en contenido 360º ya que es una pequeña fracción de todo el vídeo o imagen. Aparte, existe una gran redundancia en la codificación de los bits a causa de toda parte que no es visible para el usuario, la parte fuera del *viewport*.

Se debe tener en cuenta la monitorización y evaluación del desarrollo, la percepción del propio usuario, y la compresión de los archivos. Y junto a ello otros factores externos como la conexión del usuario a internet, o el procesamiento del contenido.

2.3. Problemas generales

No obstante, la transmisión de vídeos 360º tiene el problema del gran peso que supone cada archivo. Se debe a las mínimas exigencias que deben cumplirse para que la experiencia del usuario no se vea comprometida. Para ello se debe transmitir con gran calidad, 4K e incluso más resolución, y alta fidelidad además de tener un *frame rate* alto para evitar la cinetosis (*motion sickness)*.

Asimismo, se requiere de forma urgente una compresión para la tasa de bits (*bitrate*) de los vídeos 360º. Actualmente las imágenes y videos 360º se proyectan desde una esfera a un plano 2D para de esa forma guardarse, ser procesados y transmitidos.

Dado que en 2D se utilizan planos y en los vídeos esféricos se utiliza una esfera no se pueden utilizar los mismos estándares ya que se crea una distorsión geométrica y redundancia a causa de un muestreo no uniforme.

Otra característica que se observa como problema es que las imágenes se crean sin bordes por lo que al proyectarse en un plano bidimensional se crean bordes artificiales y con ello discontinuidad.

En el siguiente punto se definirá la evaluación de calidad visual además de varios enfoques que se han propuesto para la obtención de soluciones.

2.4. VQA: Evaluación de calidad visual

En primer lugar, para evaluar el deterioro causado por la compresión se utiliza el proceso llamado *Visual quality assessment* (VQA). Esta evaluación de calidad se divide en dos tipos: subjetiva y objetiva.

En la VQA subjetiva, se requiere a los sujetos valorar las imágenes o vídeos visualizados para obtener una puntuación de calidad subjetiva. Este método se puede realizar con vídeos 2D y 3D ya que se basa en la experiencia del usuario al reproducir el medio.

En cambio, en la VQA objetiva para vídeos esféricos no se pueden usar los mismos métodos que en los vídeos 2D ya que se obtienen experiencias diferentes.

Al hacer las comprobaciones necesarias, se toman los ficheros esenciales del conjunto de datos o *dataset* VQA-ODV [2]. En esta agrupación de vídeos omnidireccionales para tratar el VQA se encuentra un total de 600 secuencias que se utilizarán para verificar el buen funcionamiento de la aplicación a desarrollar.

A continuación se hablará de los dos tipos existentes de VQA. Se hará más énfasis en la VQA Objetiva ya que es la que se utilizará para realizar el cálculo y evaluación de los vídeos.

2.5. VQA Objetiva

Con respecto a la obtención de métricas y resultados al comparar un vídeo original y el codificado, se dispone de diferentes enfoques. En este apartado se hablará de las métricas a utilizar en la aplicación: PSNR y SSIM.

2.5.1. PSNR: Peak Signal-to-Noise ratio

Se puede definir a PSNR como la relación entre la potencia máxima posible de una imagen y el poder de ruido que corrompe y afecta a la calidad de su representación. Habitualmente se utiliza para cuantificar la calidad de reconstrucción de imágenes y vídeos con pérdida de calidad originada por compresión.

Para estimar el PSNR de una imagen, se necesita hacer comparación con una imagen limpia ideal con la máxima potencia posible. [3] Su definición se observa en la ecuación 1.

$$PSNR = 10log_{10}(\frac{(L-1)^2}{MSE}) = 20log_{10}(\frac{L-1}{RMSE}) \quad (1)$$

Se entiende como L el número de niveles de intensidad máximos posibles en una imagen. El nivel mínimo supuestamente debería ser 0.

MSE se describe como el error cuadrático medio y se define en la ecuación 2.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(O(i,j) - D(i,j))^2 \quad (2)$$

Se definen con O y D las matrices de datos de las imágenes a comparar. *Siendo* la imagen original O y la imagen dañada D.

A través de m e i se representa el número de filas de píxeles y el índice de esa fila dentro de la imagen respectivamente.

A su par, con n y j se representa el número de columnas de píxeles y el índice de esa columna dentro de la imagen respectivamente.

Una vez obtenido el PSNR, se necesita comprobar si es aceptable el valor. El rango típico de valores de PSNR en imágenes con pérdidas se encuentra entre 30 y 50 dB siempre que la profundidad de bits sea de 8 bits. En el caso de imágenes de 12 bits, se considera alto cuando es 60 dB o mayor. Cuanto más grande sea el valor, más eficiente es el método de compresión utilizado.

2.5.2. SSIM y MSSIM: Structural Similarity Index Measure

SSIM es una métrica que calcula la similitud entre dos imágenes. Se establece en un rango entre -1 y +1. El valor positivo indica que las imágenes son muy similares o idénticas, mientras que el valor negativo implica que las imágenes son muy diferentes. [4] Este valor se suele ajustar ocasionalmente a un rango de [0, 1] con el mismo significado que [-1, +1].

En la ilustración 4, se muestra el flujo y organización de SSIM. Con *Signal (x,y)* se refiere a las imágenes de referencia.

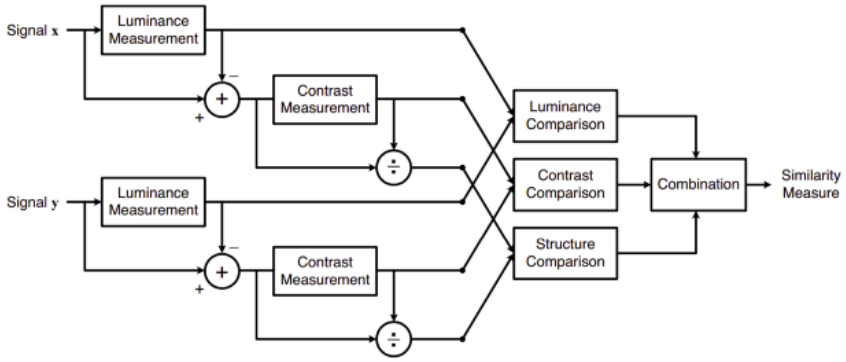

Ilustración 4: Flujo y organización de SSIM. Fuente: medium

Esta métrica extrae tres características claves de una imagen:

Luminancia: Densidad rectangular de la superficie del flujo de luz que incide, sale o pasa a través de una superficie en una dirección dada.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{3}$$

La comparación de la luminancia $l(x, y)$ es una función de μ_x y μ_y siendo μ el promedio.

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4}$$

Mediante la constante C_1 se garantiza estabilidad por si el denominador tiene un valor de 0 y se define en la siguiente figura. *L* se entiende como el rango dinámico para los valores de los píxeles.

$$C_1 = (K_1 L)^2 \tag{5}$$

Contraste: Disparidad de intensidad entre un punto de una imagen. El contraste es cero si la claridad o luminosidad es la misma.

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)^2 \right)^{\frac{1}{2}}. \quad (6)$$

La comparación del contraste es $c(x, y)$ es una función de σ_x y σ_y siendo σ la desviación estándar.

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (7)$$

En el caso de C_2 sucede lo mismo que con C_1. Además, K_1 y K_2 son constantes normales.

$$C_2 = (K_2 L)^2 \quad (8)$$

Estructura: Comparación estructural que se calcula al dividir la señal de entrada con la desviación estándar. Se representa con los valores obtenidos de x (la imagen), μ_x (la luminancia) y σ_x (el contraste).

$$(\mathbf{x} - \mu_x)/\sigma_x \quad (9)$$

La función de comparación estructural se define con $s(x, y)$. Las variables (x, y) son las dos imágenes que se están comparando.

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \quad (10)$$

siendo C_3 y σ_{xy} equivalente a:

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y). \quad (11)$$

Por último, la puntuación SSIM se calcula con la siguiente fórmula:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma} \qquad (12)$$

Sabiendo que $\alpha > 0$, $\beta > 0$, $\gamma > 0$, si se asume que $\alpha = \beta = \gamma = 1$y que $C_3 = C_2/2$ se obtiene:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}. \qquad (13)$$

Para la evaluación de calidad de imagen se recomienda aplicar las métricas por zonas pequeñas de la imagen y después calcular MSSIM, es decir, la media total.

Esta generación de la media se calcula a través de la ecuación 14.

$$\mathrm{MSSIM}(\mathbf{X},\mathbf{Y}) = \frac{1}{M}\sum_{j=1}^{M} \mathrm{SSIM}(\mathbf{x}_j,\mathbf{y}_j) \qquad (14)$$

2.5.3. S-PSNR: Spherical PSNR

Al observar el problema de la densidad de muestreo no uniforme de la proyección se determina como solución la re-proyección del vídeo o imagen. Este paso se realiza en otros dominios donde exista una densidad de muestreo uniforme.

Uno de los primeros trabajos en el PSNR esférico, llamado S-PSNR, concreta el cálculo de PSNR en un dominio esférico. Se puede definir con la fórmula 15 si se reemplaza (i, j) por una serie de puntos $\mathbb{P}$ uniformemente distribuidos en una esfera, y Npor el número de estos puntos N_{S-PSNR}.

$$Q = g\left(\frac{1}{N}\sum_{(i,j)} D(i,j)\right) \qquad (15)$$

Su cálculo se basa en una serie de puntos $\mathbb{P}$ uniformemente distribuidos en dos esferas unitarias generadas por la imagen de referencia y la imagen a testar. De esta forma, se puede comparar dos imágenes con distinta resolución.

Para cada posición del punto p en una esfera, obtenemos su valor en la correspondiente ubicación del píxel en el plano proyectado original a través de sus vecinos o de métodos de interpolación bilineales. [5] Tras este proceso se generan dos variantes, S-PSNR-NN y S-PSNR-I.

Se denomina S-PSNR-NN cuando la posición en una esfera unitaria se redondea a la posición vecina más cercana en la correspondiente imagen.
Por otro lado, se expresa a través de S-PSNR-I cuando el valor de la señal de la posición en la esfera se deduce a causa de la interpolación de posiciones vecinas en la respectiva imagen.

Aun así, en la implementación oficial de S-PSNR el valor que toma $N_{S-PSNR} = 655362$. Si se compara con el número de píxeles habitual en una imagen o vídeo de 360º con una resolución mayor a 2K, es un valor menor.

2.5.4. WS-PSNR: Weighted Spherical PSNR

A través de la reproyección se resuelve completamente el problema de la densidad de muestreo no uniforme. Aun así, se requiere mucho tiempo ya que el procedimiento es extremadamente complejo.

Por lo tanto, otros enfoques han aparecido incluyendo la asignación de ponderación o de peso en el momento de calcular el PSNR y el SSIM para balancear la contribución de distorsión en diferentes posiciones de los píxeles.

Se entiende como peso o ponderación a la asignación de importancia que tiene un elemento en un conjunto dependiendo de ciertos criterios.

Este enfoque se conoce como WS-PSNR y debido a que es un método de referencia completa es similar a PSNR debido a que ambas métricas se deben utilizar solamente con vídeos o imágenes que tengan la misma resolución.

WS-PSNR se define como una métrica para medir la distorsión de un vídeo reconstruido en un dominio esférico. La distorsión de cada fotograma se entiende como el error acumulativo de todos los píxeles. Este error se pondera por el pertinente área esférica de acuerdo a la función de mapeo entre la esfera y el formato definido.

Este caso se puede observar en la ecuación 16, donde $w(i,j)$ se define como la asignación de ponderación de la posición del píxel en (i,j).

$$Q = g\left(\frac{\sum_{(i,j)} w(i,j)D(i,j)}{\sum_{(i,j)} w(i,j)}\right) \qquad (16)$$

Existe un caso especial en el que $w(i,j)$ es igual a una constante, por lo tanto se considera a todos los píxeles con la misma importancia al calcular PSNR y SSIM.

Ahora bien, se observa un exceso de peso en la distorsión de las ubicaciones de los píxeles con una densidad de muestreo grande. En el caso contrario, donde se denota una densidad de muestreo pequeña, se percibe la escasez de peso.

2.5.5. VMAF: Video Multimethod Assessment Fusion

VMAF [6] se define como un algoritmo de evaluación de calidad de vídeo perceptual. Su funcionamiento se basa en evaluar un vídeo distorsionado, re-codificado o modificado al ser comparado con una referencia original.

Ahora bien, para este proyecto se ha utilizado la versión VMAF 360 [7]. Esta versión es una modificación de la métrica original para ser utilizada con vídeos esféricos.

Para calcular la calificación se deben tener en cuenta las dependencias de la percepción humana de la calidad de imagen. Esta puntuación se forma mediante máquinas de vectores de soporte. Estas máquinas son un conjunto de algoritmos de aprendizaje automático.

Se combinan tres métricas para predecir la calidad de vídeo:

- VIF: Mide la pérdida de fidelidad de la información
- DLM: Considera la pérdida en deficiencias que distraen la atención del espectador
- MCPD: Mide basándose en la luminancia de la diferencia temporal entre fotogramas

La puntuación de VMAF se limita de 0 a 100, siendo 0 la calidad más baja y 100 la calidad más alta. Una calificación de 20 supone una calidad muy mala, 40 indica que es calidad mala, 60 es regular, 80 se considera una calidad buena y 100 es excelente.

Todas estas puntuaciones se agrupan en una secuencia utilizando la media aritmética para dar como resultado una puntuación media de opinión diferencial también conocida como DMOS.

2.6. VQA Subjetiva

Aunque la VQA que se tratará en la aplicación es la rama objetiva, también es interesante conocer enfoques subjetivos.

2.6.1. Enfoque I: Estimación de movimiento

La estimación de movimiento y la compensación de movimiento se definen como dos pasos importantes en la compresión de vídeos para reducir la redundancia temporal en los fotogramas. Estos pasos suelen ser tratados por la mayoría de los estándares de compresión como H.264 o H.265, entre otros.

Primeramente, la estimación de movimiento examina el movimiento de los objetos existentes en una secuencia para obtener vectores que representen esa estimación.

Por otro lado, la compensación de movimiento utiliza el conocimiento obtenido del movimiento de un objeto para conseguir comprimir los datos.

Con el resultado producido gracias a estos dos pasos, se puede aplicar una técnica llamada Predicción Interframe. Este procedimiento permite codificar vídeos con el menor número posible de bits tras explotar la alta correlación temporal entre fotogramas sucesivos.

Su funcionamiento se define como la predicción de fotogramas consecutivos. Para ello se aplica a estos frames el vector de movimiento obtenido con los anteriores pasos.

Aún así, su uso en vídeos 360º no es del todo apropiado. La implementación de estos modelos tradicionales de movimiento afectan directamente al medio y aumenta la diferencia de los vectores de movimiento entre bloques, y en consecuencia, aumenta el bitrate.

Debido a los problemas que se originan, se han definido varios enfoques para modificar el proceso de la estimación de movimiento en los vídeos 360º. Una solución se basa en aplicar el proceso en un dominio esférico en lugar de ser proyectado en un plano 2D.

2.6.2. Enfoque II: Corrección de densidad de muestreo

Como se ha mencionado anteriormente, se origina distorsión geométrica y un exceso de bitrate extra para las áreas sobremuestreadas en el plano 2D. A continuación se habla brevemente de dos soluciones:

La primera solución se fundamenta en aplicar suavizado gaussiano en las regiones redundantes para disminuir la distorsión en los bordes. Con la segunda se incorpora una corrección de densidad de muestreo en el proceso de cuantificación.

2.7. Aplicaciones existentes

En el ámbito de aplicaciones que tratan con métricas de calidad se pueden encontrar numerosas que trabajan con vídeos 2D y en menor medida que permiten trabajar con vídeos 360º.

Sin embargo, en estos programas siempre existe una carencia de una característica importante.

Cuando la herramienta tiene interfaz carece de opción para tratar con vídeos esféricos. Mientras que si la aplicación permite cargar vídeos 360 carece de interfaz y por ende su ejecución se realiza a través de una línea de comando en la consola. Esta forma de ejecutar el programa supone una baja accesibilidad a usuarios sin conocimientos de informática.

Por lo tanto el objetivo de este proyecto es suplir las necesidades originadas por las dos aplicaciones analizadas. Crear una aplicación con interfaz que permita el tratamiento de vídeos omnidireccionales.

A continuación se analizarán algunas herramientas para observar las principales diferencias con la aplicación desarrollada.

2.7.1. Con interfaz

BVQM: Batch Video Quality Metric [8] es un software que evalúa de forma objetiva la calidad de vídeo 2D. A través de su uso se obtienen métricas tales como registro temporal, registro espacial, escalado espacial, entre otras como se muestra en la ilustración 5.

Una vez realizada la evaluación, se muestran los resultados en una gráfica y se puede exportar el reporte generado en Excel. Como extra, se permite ajustar la calibración y otros ajustes opcionales de forma manual.

En comparación con QualiApp360, no dispone de métricas de calidad para vídeos 360. Pero incluye una característica interesante con la que seleccionar un lote de vídeos y evaluar todos de una pasada.

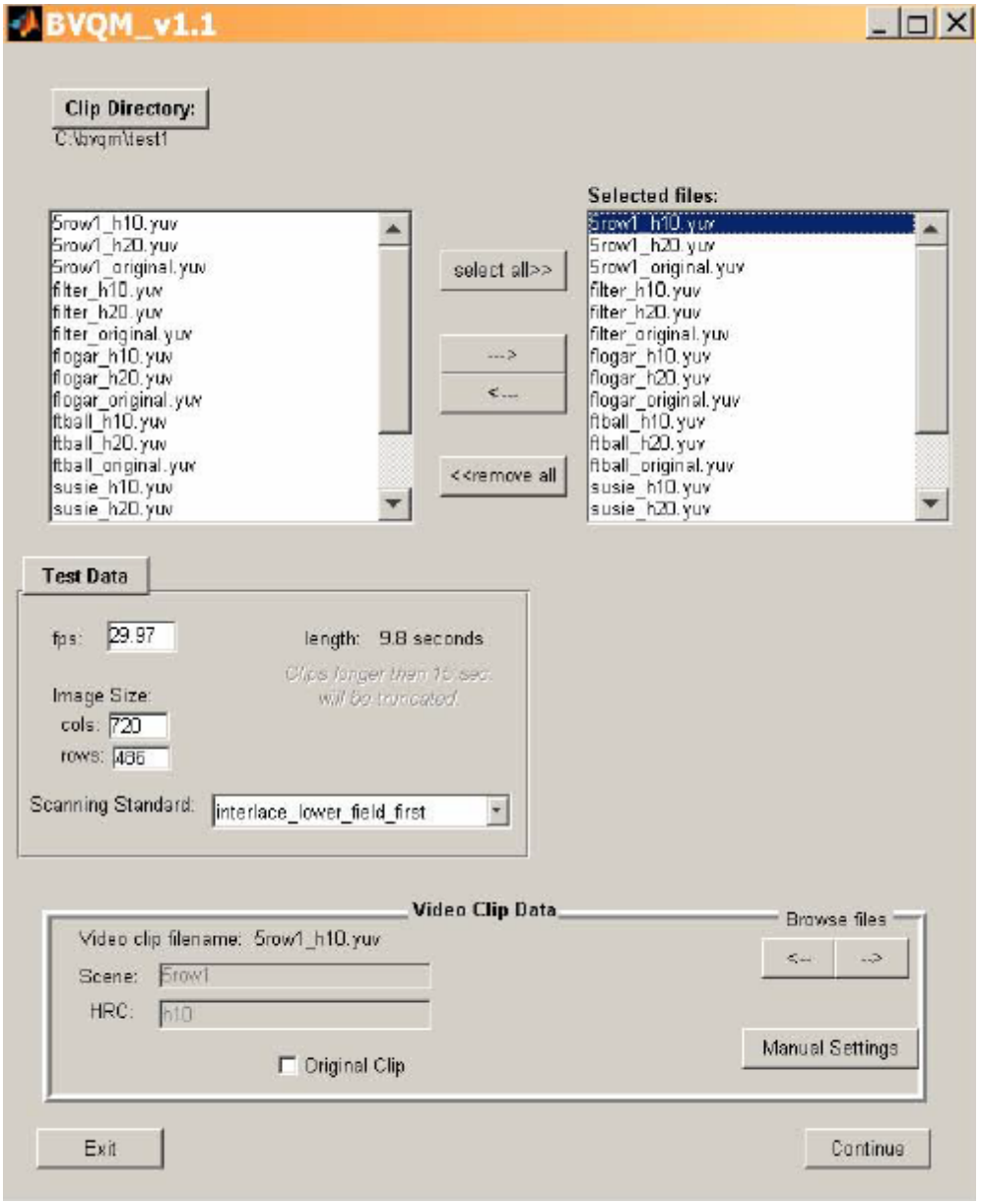

Ilustración 5: Interfaz de BVQM. Fuente: BVQM User's Manual [PDF MANUAL]

2.7.2. Sin interfaz

360tools [9] es un proyecto desarrollado en C++ y enfocado en el procesamiento de vídeos esféricos. Su funcionamiento se divide en dos ámbitos, la conversión de vídeo y la evaluación de vídeos.

El apartado más interesante es la evaluación, ya que se utilizan métricas tales como PSNR, S-PSNR, WS-PSNR, CPP-PSNR para obtener datos de las comparativas entre dos vídeos. Varias de estas métricas se han explicado anteriormente en este documento y se van a desarrollar en la aplicación.

Su uso se realiza a través de una línea de comando en terminal. Esto supone un obstáculo para usuarios no acostumbrados a programas sin interfaz.

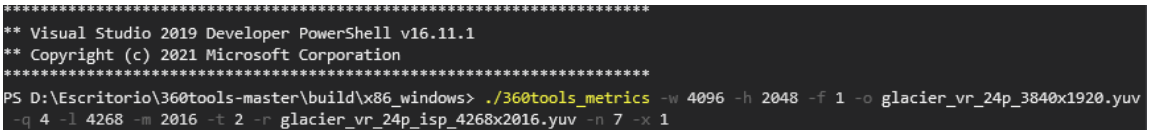

Ilustración 6: Llamada de ejecución de 360tools desde la terminal de Visual Studio

2.8. Conclusiones

Una vez se ha recogido toda la información sobre las métricas y los enfoques a utilizar, se ha elegido utilizar modelos matemáticos como PSNR, SSIM o VMAF para realizar una comparación entre un vídeo normal y el mismo archivo distorsionado.

Inicialmente se aprecia que *BVQM* muestra de forma muy coherente los resultados a través de una gráfica comparativa entre todos los vídeos insertados. Además da la información al usuario y se le permite descargarla para almacenarla y no perder los resultados del proceso.

Como contraste se encuentra *360tools*, que carece de interfaz pero es similar a la aplicación planteada a desarrollar en lo que a métricas se refiere.

Por lo tanto, se ha decidido optar por una apariencia más sencilla y directa para recoger los vídeos y seleccionar las métricas a evaluar.

Al pensar en cómo llevar a cabo la aplicación, se ha elegido Python como el lenguaje de desarrollo por el previo conocimiento adquirido en la carrera y por tener librerías específicas sobre algoritmos para analizar vídeos. Además, se usará la librería OpenCV para generar y analizar vídeos.

Tras analizar el diseño de otros programas se plantean las funcionalidades que se deben desarrollar y la interfaz que se debe diseñar. En los siguientes apartados se explicará el diseño y desarrollo de la aplicación.

3. Planificación

En este capítulo se tratará la fase de análisis donde se mostrará una vista general de la estructura planeada para el proyecto.

El objetivo es crear una aplicación que a través de dos vídeos se evalúen y se obtenga como resultado datos de métricas para comparar los dos archivos. La aplicación se divide en dos bloques marcados: La interfaz gráfica con los menús y funcionalidades diseñadas, y la parte de cálculo generadora de resultados.

El *target* es principalmente usuarios con conocimientos mínimos previos en el ámbito de la informática. Aún así, en el caso de que un usuario con conocimientos más básicos se disponga a utilizar la aplicación no habrá problema ya que se ha diseñado desde un punto de vista intuitivo.

3.1. Análisis de Requisitos

A la hora de decidir las funcionalidades a implementar, se fijaron una serie de características mínimas. En los siguientes puntos se explican los requisitos de cada tipo más detalladamente. Dentro del proyecto se distinguen dos tipos de requisitos: los funcionales y los no funcionales.

3.1.1. Requisitos funcionales

En las tablas 1 y 2 se recogen los requisitos funcionales de iniciar y salir de la aplicación.

RF1	Iniciar la aplicación.
Explicación	Abre el programa y carga la interfaz.
Datos de entrada	Ninguno.
Datos de salida	Interfaz de la aplicación.

Tabla 1: Requisito Funcional 1

RF2	Salir de la aplicación.
Explicación	Cierra el programa.
Datos de entrada	Ninguno.
Datos de salida	Ninguno.

Tabla 2: Requisito Funcional 2

En la tabla 3 se puede observar el requisito 3 que trata de recoger los datos referentes al ancho y alto del vídeo para así obtener la resolución aparte del número de fotogramas.

RF3	Recoger datos sobre la resolución y el número de *frames.*
Explicación	A través de entrada de texto recoge los valores necesarios para poder abrir correctamente los archivos con extensión yuv.
Datos de entrada	Ancho, alto, número de frames.
Datos de salida	Guarda dichos valores para la carga de los vídeos.

Tabla 3: Requisito Funcional 3

Los requisitos funcionales de carga de ficheros, tanto de vídeo (tabla 4) como el fichero de texto con los puntos esféricos (tabla 5) se muestran en sus respectivas tablas.

RF4	Cargar ficheros de vídeo.
Explicación	Permite seleccionar un vídeo de referencia y uno distorsionado con extensión mp4, avi o yuv.
Datos de entrada	Fichero de vídeo.
Datos de salida	Carga los vídeos para su evaluación.

Tabla 4: Requisito Funcional 4

RF5	Cargar fichero txt.
Explicación	Permite seleccionar un archivo con extensión txt. Este fichero debe tener las coordenadas de los puntos P en la esfera. Es obligatorio para iniciar la métrica S-PSNR.
Datos de entrada	Fichero txt.
Datos de salida	Carga los puntos esféricos del archivo.

Tabla 5: Requisito Funcional 5

El requisito funcional de la selección de métricas se contempla en la tabla 6.

RF6	Selección de métricas.
Explicación	Entre las 6 disponibles, permite al usuario elegir las métricas a evaluar.
Datos de entrada	Ninguno.
Datos de salida	Métricas elegidas.

Tabla 6: Requisito Funcional 6

El inicio de la evaluación de calidad de las métricas seleccionadas se presenta en la tabla 7.

RF7	Iniciar la evaluación.
Explicación	Ejecuta las métricas seleccionadas. Su procesamiento se observa a través de una barra de progreso.
Datos de entrada	Métricas a evaluar.
Datos de salida	Resultado mostrado por consola.

Tabla 7: Requisito Funcional 7

La selección de filtros y su correspondiente requisito funcional se muestra en la tabla 8.

RF8	Selección de filtros.
Explicación	Entre los 4 disponibles, permite al usuario elegir las métricas a evaluar.
Datos de entrada	Ninguno.
Datos de salida	Métricas elegidas.

Tabla 8: Requisito Funcional 8

En la tabla 9 se puede ver el requisito funcional de mostrar una gráfica.

RF9	Mostrar una gráfica.
Explicación	Mediante los datos obtenidos, genera y muestra una gráfica.
Datos de entrada	Datos con el resultado de la evaluación y los filtros seleccionados.
Datos de salida	Gráfica mostrada en la interfaz.

Tabla 9: Requisito Funcional 9

El requisito funcional de Guardar un archivo CSV con la puntuación de las métricas se presenta en la tabla 10.

RF10	Guardar CSV.
Explicación	Obtención de un fichero CSV con los datos obtenidos en las métricas seleccionadas.
Datos de entrada	Datos con el resultado de la evaluación.
Datos de salida	Fichero CSV generado.

Tabla 10: Requisito Funcional 10

3.1.2. Requisitos no funcionales

RNF1: Documentación

1. Con la instalación se otorga un manual pdf al usuario para que se pueda guiar a través de la aplicación.
2. El manual debe permitir al usuario entender el funcionamiento total de la aplicación y servir para solucionar dudas cuando surjan.
3. La implementación del código debe ser clara y con comentarios para que sea posible agregar más funcionalidades. Además, al documentar se permite a otros desarrolladores utilizar la aplicación o entenderla rápidamente.

RNF2: Portabilidad

1. Disponibilidad para Windows 10.
2. Una vez obtenidos los archivos necesarios para que VMAF funcione correctamente no es necesario disponer de una conexión a internet. Estos archivos se obtienen la primera vez que se elige y ejecuta la métrica. Si ya se poseen estas carpetas, el programa pasaría a funcionar *offline* correctamente.

RNF3: Interfaz y usabilidad

1. Aplicación en inglés para hacerla entendible a un público general.
2. Mantener una interfaz sencilla e intuitiva.
3. Todos los elementos en la misma ventana.
4. El uso del programa no debe suponer esfuerzo al usuario.
5. Uso guiado gracias a la habilitación y deshabilitación de elementos.
6. Habilitar/Deshabilitar elementos una vez tenga sentido que aparezcan.
7. El usuario no se debe memorizar la memoria para entender el programa, sus movimientos deben ser mecánicos.

3.2. Análisis de casos de uso

Para describir las posibles interacciones del usuario con la aplicación se necesita detallar los casos de uso, a través de Visual Paradigm se han creado los diagramas necesarios para explicar la configuración del programa. Con estos escenarios se indican las respuestas sugeridas tras observar las acciones del usuario.

3.2.1. Diagrama de Casos de Uso

Un diagrama de casos de uso se utiliza para indicar qué realiza el sistema, pero no define cómo lo hace. Explica las interacciones entre el sistema y diferentes elementos externos como puede ser un actor.

Dentro de los diagramas de casos de uso encontramos diferentes elementos importantes que se deben explicar.

Actor
Se define actor como cualquier persona que interactúe o utilice la aplicación, por lo tanto en este proyecto el usuario es el único actor existente. Está representado por un muñeco.

Asociación
Línea que conecta al actor con los procesos en los que puede actuar o interactuar.

Proceso
Acción o ejecución que puede realizar el usuario en este programa. Se representa con óvalo.

Paquete
Agrupa elementos, en este caso procesos, para una mejor visualización y organización del diagrama. Es reconocible por su apariencia de carpeta.

Sistema
Agrupa con un cuadrado todo el programa y muestra de la misma forma qué elementos pertenecen al sistema y qué elementos son externos.

En la ilustración 7 se encuentra representado el diagrama de casos de uso de este proyecto.

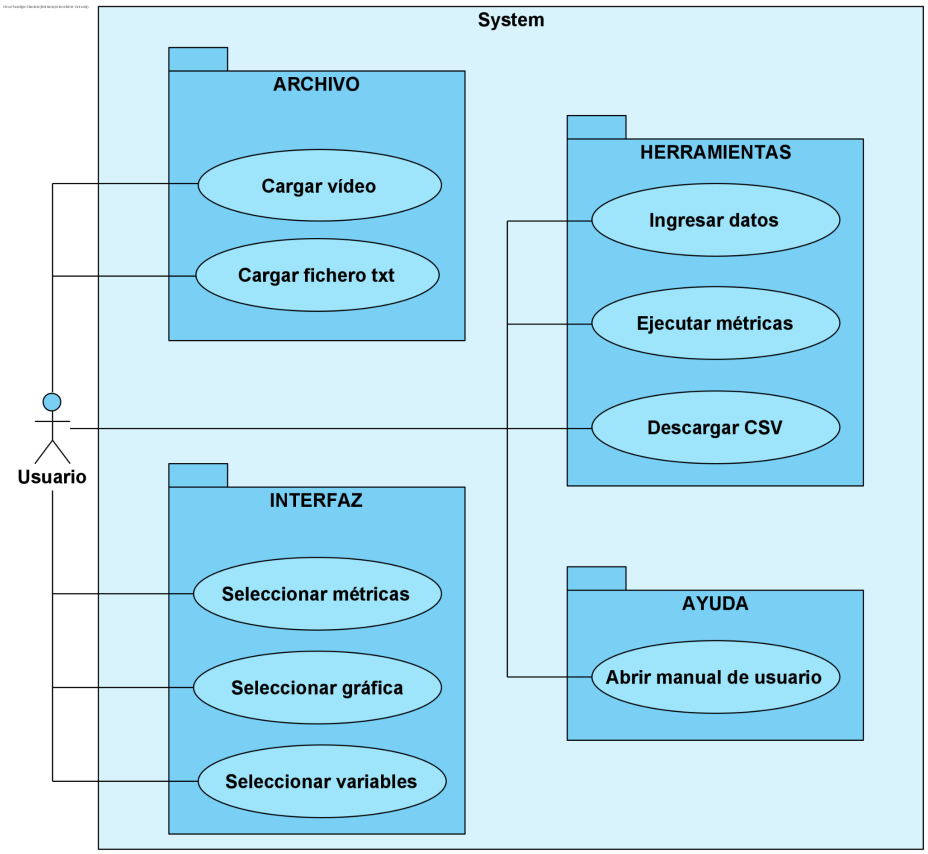

Ilustración 7: Diagrama de casos de uso

3.2.2. Especificación de Casos de Uso

En las tablas mostradas en este apartado se definen y describen los casos de uso con más profundidad. Están ordenadas en el orden en el que el usuario utiliza la aplicación.

Caso de uso	Ingresar datos	
Actor	Usuario	
Descripción	El usuario ingresa los valores para poder cargar vídeos yuv.	
Precondición		
Postcondición	Tras ingresar los valores, se podrá cargar los vídeos yuv (*C.U.Cargar video*)	
Requisitos	RF3	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario introduce los valores dentro de los campos.	2. El sistema almacena dichos valores para abrir ficheros de video yuv.

Tabla 11: Especificación: Ingresar datos

En la tabla 11 se especifica el caso de uso Ingresar datos. Mientras que la especificación de la carga de ficheros se muestra para Cargar Vídeo en la tabla 12 y Cargar fichero de texto en la tabla 13.

Caso de uso	Cargar vídeo	
Actor	Usuario	
Descripción	El usuario selecciona a través de una ventana emergente los archivos de vídeo que quiere evaluar.	
Precondición	Si se quiere evaluar un vídeo con extensión yuv necesitará ingresar los datos en los campos a la izquierda de los botones existentes para cargar vídeos.	
Postcondición	Tras cargar los dos vídeos se podrán seleccionar métricas (*C.U. Seleccionar métricas*)	
Requisitos relacionados	RF4	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón y selecciona un vídeo.	2. El sistema almacena el directorio del fichero escogido y guarda el vídeo en una variable.
Flujo alternativo 1	1. El usuario cierra la ventana emergente.	2. El sistema no guarda ningún archivo.

Tabla 12: Especificación Cargar Vídeo

Caso de uso	Cargar fichero texto	
Actor	Usuario	
Descripción	El usuario selecciona a través de una ventana emergente el fichero txt con los puntos P en una esfera necesarios para evaluar S-PSNR.	
Precondición		
Postcondición	Tras cargar el fichero se podrá seleccionar la métrica S-PSNR (*C.U. Seleccionar métricas*)	
Requisitos relacionados	RF5, RNF3.5	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón y selecciona el fichero de texto.	2. El sistema almacena el directorio del fichero escogido y lee el contenido del archivo de texto.
Flujo alternativo 1	1. El usuario cierra la ventana emergente.	2. El sistema no carga ningún archivo.

Tabla 13: Especificación Cargar fichero texto

La selección de las métricas se puede ver especificado en la tabla 14 junto a la especificación de la ejecución de las métricas de la tabla 15.

Caso de uso	Seleccionar métricas	
Actor	Usuario	
Descripción	El usuario selecciona que métricas de calidad quiere evaluar.	
Precondición	Se tiene que haber cargado correctamente tanto el video de referencia como el video distorsionado.	
Postcondición		
Requisitos relacionados	RF6, RNF3.5, RNF6	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón.	2. En ese momento el sistema muestra la gráfica en una ventana nueva.

Tabla 14: Especificación: Seleccionar métricas

Caso de uso	Ejecutar métricas	
Actor	Usuario	
Descripción	El usuario inicia el proceso de evaluación de calidad mediantes las métricas seleccionadas.	
Precondición	Se tiene que haber elegido una métrica para que el botón que inicia el proceso esté activo. Se tiene que haber pulsado este botón.	
Postcondición		
Requisitos relacionados	RF7	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón.	2. El sistema empieza a evaluar las métricas.

Tabla 15: Especificación: Ejecutar métricas

En el apartado de la gráfica, se observan las especificaciones de Seleccionar variables en la tabla 16 y Seleccionar gráfica en la tabla 17.

Caso de uso	Seleccionar variables	
Actor	Usuario	
Descripción	El usuario selecciona mediante botones qué variables quiere mostrar a la hora de observar la gráfica en una nueva ventana.	
Precondición	Se tiene que haber generado la gráfica, por ende se tiene que haber evaluado correctamente dicha métrica.	
Postcondición		
Requisitos relacionados	RF8, RF9	
Flujo de eventos	Usuario	Sistema
	1. El usuario marca los seleccionables para elegir qué variables mostrar.	2. En ese momento el sistema los añade.

Tabla 16: Especificación: Seleccionar variables

Caso de uso	Seleccionar gráfica	
Actor	Usuario	
Descripción	El usuario selecciona mediante botones qué gráfica quiere abrir en una nueva ventana.	
Precondición	Se tiene que haber generado la gráfica, por ende se tiene que haber evaluado correctamente dicha métrica.	
Postcondición		
Requisitos relacionados	RF9	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón.	2. En ese momento el sistema muestra la gráfica en una ventana nueva.

Tabla 17: Especificación: Seleccionar gráfica

La descarga de un archivo CSV se especifica en la tabla 18.

Caso de uso	Descargar CSV	
Actor	Usuario	
Descripción	Mediante un botón en la interfaz, el actor se podrá descargar los resultados de las métricas en formato CSV.	
Precondición	Se debe haber ejecutado y finalizado correctamente la evaluación de mínimo una métrica para habilitar esta función.	
Postcondición	Se podrán descargar los resultados de las métricas seleccionadas.	
Requisitos relacionados	RF10	
Flujo de eventos	Usuario	Sistema
	1. El caso de uso inicia cuando el usuario da click al botón.	2. En ese momento el sistema muestra la gráfica en una ventana nueva.
Flujo alternativo 2		

Tabla 18: Especificación: Descargar CSV

El caso de uso de Abrir el manual de usuario se puede encontrar definido en la tabla 19.

Caso de uso	Abrir manual de usuario	
Actor	Usuario	
Descripción	El usuario abre el manual de usuario al darle click al botón de ayuda ubicado abajo a la derecha.	
Precondición		
Postcondición	Se abre el manual con el programa por defecto para abrir archivos pdf.	
Requisitos relacionados	RNF1.1, RNF1.2, RNF3.4	
Flujo de eventos	Usuario	Sistema
	1.El usuario da click al icono de ayuda.	2. El sistema abre el manual pdf.
Flujo alternativo 2	1. El usuario da click, pero el fichero no se encuentra en el directorio correspondiente.	2. El sistema no puede abrir el fichero y no ocurre nada.

Tabla 19: Especificación: Abrir manual de usuario

3.2.3. Diagrama de clases

A través de la opción de ingeniería inversa del programa Visual Paradigm se ha obtenido el diagrama de clases importando el código de la aplicación. Como se puede apreciar en la ilustración 8, en este proyecto solo se encuentra la clase principal y las funciones que se llaman para evaluar métricas.

Las funciones más importantes que tratan sobre las métricas se muestran en este documento en las ilustraciones 16 para PSNR, 17 para MSSIM, 18 para SSIM, 19 para VMAF, 20 para S-PSNR y 21 para WS-PSNR.

Ilustración 8: Diagrama de Clases

3.2.4. Diagrama de secuencia

Mediante estos diagramas se muestra la interacción de las clases y objetos entre ellos mismos a través de mensajes. Su utilidad reside en mostrar el proceso de los casos de uso para entender los procesos que se realizan en el proyecto y cómo se relacionan.

En la ilustración 9 se encuentra el diagrama de secuencia que recoge los procesos para abrir todo tipo de ficheros incluyendo los vídeos, el archivo de texto y el manual de usuario.

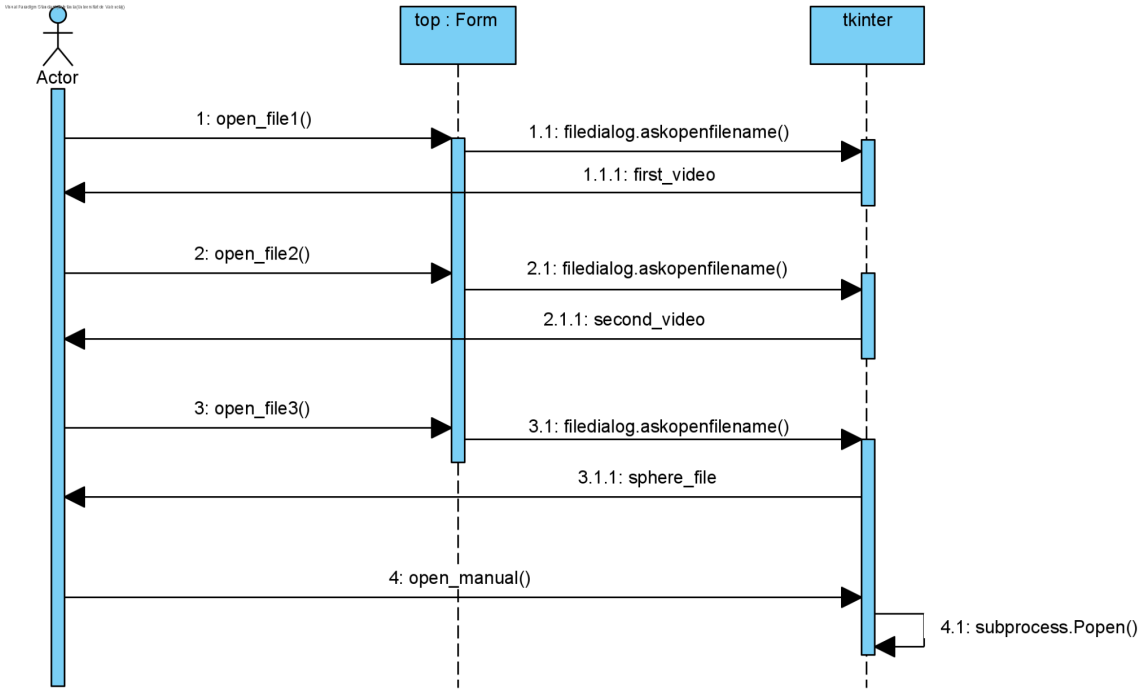

Ilustración 9: Diagrama de secuencia de Cargar video, cargar fichero de texto y Abrir manual de Usuario.

Por otra parte, se ha simplificado el diagrama de secuencia de la evaluación (véase ilustración 10) y se ha indicado el proceso con una plantilla que recoge todas las funciones de las métricas. En este diagrama también se incluye la creación de las gráficas y el guardado del CSV.

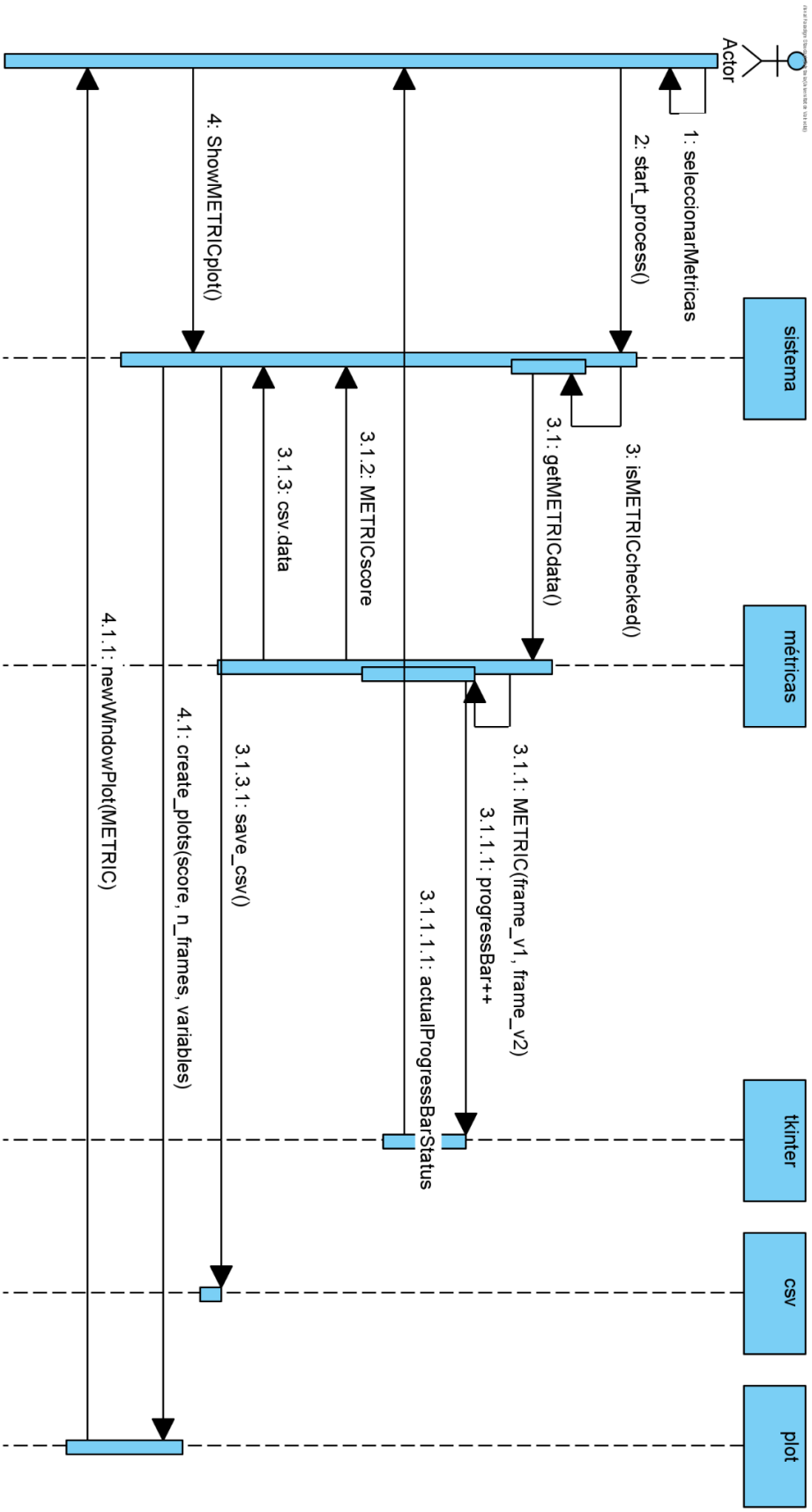

Ilustración 10: Diagrama de secuencia de Seleccionar métricas, Ejecutar métricas, Seleccionar gráfica, Descargar CSV.

3.3. Metodología en cascada

Tras haber especificado los requisitos funcionales y no funcionales junto a los casos de uso, se prosigue con el comentario de qué metodología se ha decidido usar para desarrollar el proyecto.

Para el desarrollo se ha utilizado la metodología en cascada. Su uso implica la división en bloques para dividir la carga de trabajo y reconocer rápidamente el progreso en cualquier momento. Aún así uno de los problemas más graves es el arrastre de errores que puede surgir, por lo tanto es importante realizar comprobaciones y retornar en caso necesario. [10]

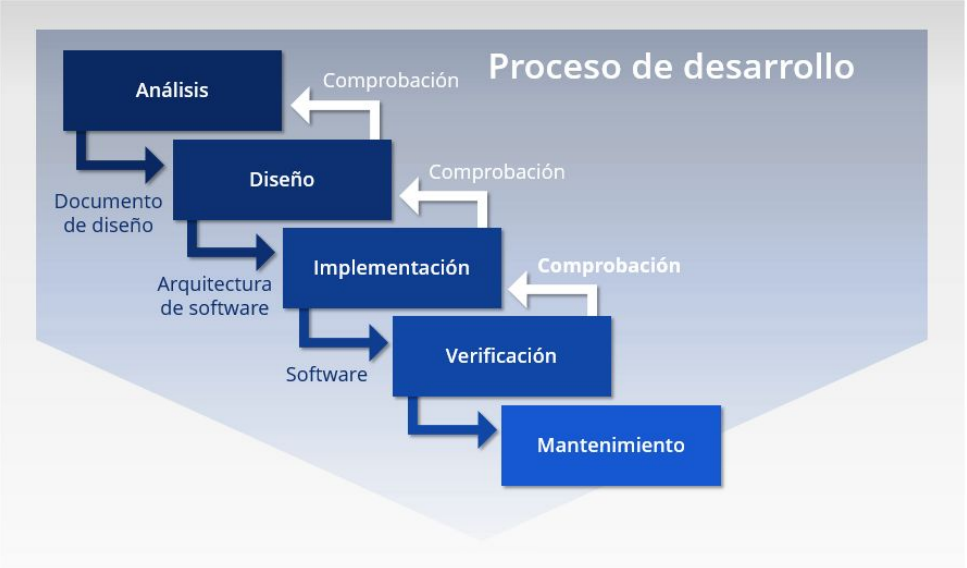

Ilustración 11: Metodología en cascada. Fuente: ionos

Como se muestra en la ilustración 11, las fases se realizan secuencialmente pero se permite regresar en caso de necesitar modificar algún elemento y continuar con las siguientes fases.

Una vez se ha definido el ciclo de vida del proyecto, se planificó cada fase. Se tiene como objetivo la estimación en tiempo de la total realización del proyecto.

3.4. Tareas e hitos

Dividir el proyecto íntegro en tareas es necesario para ver cuánto se lleva completado del proceso. Este proyecto se ha dividido en 6 diferentes bloques, cada uno con su definición y su tiempo estimado.

T1. Realización de un estudio/búsqueda de los algoritmos actuales sobre la evaluación de vídeos 360 - (2 semanas).
T2. Estudio y comprensión de dichos algoritmos para su posterior implementación - (4 semanas).
T3. Diseño de una herramienta para unificar la evaluación de la calidad de los vídeos 360 - (2 semanas).
T4. Desarrollo de la propia aplicación previamente diseñada (4 semanas).
T5. Diseño y elaboración de las pruebas convenientes para comprobar el correcto funcionamiento de la herramienta - (2 semanas).
T6. Análisis de los resultados y obtención de las conclusiones sobre la aplicación desarrollada, finalizando así la memoria del TFG - (2 semanas).

3.5. Estimación temporal

La finalidad es la estimación del tiempo que se necesita para finalizar el proyecto y analizar la variación que ha habido frente al periodo realmente dedicado y la planificación original.

Planificación inicial al inicio de desarrollo en Octubre de 2020 representada en semanas.

	Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	Sem 7	Sem 8	Sem 9	Sem 10	Sem 11	Sem 12	Sem 13
T1													
T2													
T3													
T4													
T5													
T6													

(20)

Tabla 20: Planificación inicial

La valoración inicial plantea una duración de 3 meses y 1 semana haciendo 8 horas diarias, aproximadamente.

A causa del horario irregular estudiantil a jornada completa no se pudo completar rigurosamente el planteamiento inicial y se ha alargado con pausas intermedias por exámenes.

Planificación final en Agosto de 2021 representada en meses.

	Oct	Nov	Dic	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago
T1											
T2											
T3											
T4											
T5											
T6											

(21)

Tabla 21: Planificación final

Desde octubre hasta diciembre se trabajó los fines de semana en el estudio de los algoritmos y su comprensión.

En enero hubo parón debido a los exámenes de la universidad.

Tras los exámenes, en febrero se retomó el estudio de los algoritmos para refrescar conocimientos y se empezó a redactar la memoria.

En marzo y abril se definió el diseño de la aplicación y a finales de abril se empezó a implementar esta solución.

Se volvió a parar el proceso en mayo y junio a causa de entrega de prácticas finales, exámenes y el inicio de las prácticas curriculares.

En julio se retomó el desarrollo de la aplicación.

Desde julio el tiempo invertido y las horas mínimas diarias realizadas aumentaron significativamente. Se realizaron jornadas de 6 a 10 horas diarias para conseguir finalizar el proyecto a tiempo. No se pudo hacer más horas debido a la jornada diurna de las prácticas curriculares.

En agosto se finalizó el desarrollo y en menos tiempo del originalmente dado se realizaron las pruebas y el análisis de los resultados.

Tras esto las horas realizadas por mes han quedado como se indican en la tabla 20.

	Mes	Días trabajados	Total de horas
	Octubre	7	28
	Noviembre	16	64
	Diciembre	6	24
	Febrero	6	24
	Marzo	8	32
	Abril	16	64
	Julio	25	175
	Agosto	28	224
TOTAL		112	635

Tabla 22: Total de horas finales por mes

3.6. Estimación de costes

En esta parte se detallan los costes procedentes de la realización de la anterior planificación teniendo en cuenta el personal y los materiales necesarios para el buen desarrollo del proyecto.

3.6.1. Costes del personal

La alumna se encarga del diseño y desarrollo del proyecto. Por lo tanto todos los costes de personal se centran en una persona.

En la planificación se han obtenido 112 días comenzando en octubre de 2021 y finalizando en agosto del 2021. Observando la jornada de trabajo, se divide el tiempo en dos fases.

Una primera fase que se sitúa desde octubre hasta abril. En esta fase se suman 236 horas en 59 días.Y una segunda fase de julio y agosto que suma 399 horas en 53 días.

Invirtiendo en la primera fase una media de 236 horas y un total de 399 horas en la segunda fase, se obtiene un total de 635 horas de trabajo.

Es preciso señalar que varios días, en especial los coincidentes con exámenes, entrega de prácticas de laboratorio no se ha podido llevar a cabo las horas diarias establecidas. Aún así, se compensaba en días en los que se superaba notablemente dichas horas.

En la tabla 23 se muestra un resumen de los costes totales del personal.

	Coste por horas	Horas	Coste total	Coste SS (20%)	Total + SS
Diseñador	15€	76	1.140€	228€	1.368€
Programador	20€	419	8.380€	1.676€	10.056€
TOTAL		495	9.520€	1.904€	11.424€

Tabla 23: Costes del personal

3.6.2. Costes de software

En la tabla 24 se incluye todo el material software necesario para la completación de la aplicación.

Material	Precio de unidad	Periodo de amortización	Duración del proyecto	Coste total
Microsoft Windows 10 Pro	175,15€	24 meses	11 meses	80,27€
Visual Paradigm	0€	11 meses	11 meses	0€
Visual Studio Code	0€	11 meses	11 meses	0€
Adobe Photoshop	24,19€	1 mes	1 mes	24,19€
COSTE TOTAL				104,46€

Tabla 24: Costes del software

3.6.3. Costes de hardware

El coste total del hardware utilizado sube hasta 320,83€ tal como se calcula en la tabla 25.

Material	Precio de unidad	Periodo de amortización	Duración del proyecto	Coste total
Equipo Intel(R) Core(™) i5-4460 CPU @3.20GHz	700€	24 meses	11 meses	320,83€
Monitor MSI G271 x2	200€	24 meses	11 meses	183,33
COSTE TOTAL				320,83€

Tabla 25: Costes del hardware

3.6.4. Costes totales

En la tabla 26 se recoge el coste total del proyecto incluyendo al personal y el material. Se puede observar que el coste total del proyecto ha sido 9.945,29 euros.

Coste total	
Personal	9.520€
Software	104,46€
Hardware	320,83€
TOTAL	9.945,29€

Tabla 26: Costes totales del proyecto

3.7. Estudio de viabilidad

En este apartado se va a estudiar la viabilidad general del proyecto. Se divide en distintos apartados que se definen brevemente en los siguientes puntos.

- **Viabilidad Técnica**: Este apartado trata de analizar si se tienen los medios técnicos necesarios para la completa realización del proyecto.

- **Viabilidad Estratégica**: Se analiza si es viable la realización del proyecto en este momento.
- **Viabilidad Legal**: Se estudia si se cumple con la normativa legal vigente.
- **Viabilidad Económica**: Se analiza el posible beneficio resultante del desarrollo de la aplicación.

3.7.1. Viabilidad técnica

Tras analizar los requisitos funcionales y las tecnologías a utilizar se puede decir que es posible abordar el proyecto. El lenguaje de programación es de código abierto y las fórmulas utilizadas se encuentran en diversos artículos, investigaciones y páginas web.

3.7.2. Viabilidad estratégica

En la actualidad, no existe ninguna aplicación que trate la evaluación de los vídeos esféricos a través de una aplicación con interfaz. Este proyecto es una nueva aportación al ámbito VR y de medios 360º.

3.7.3. Viabilidad legal

El algoritmo de VMAF se ha tomado de proyectos *open source* ya finalizados. Por ello, tanto la implementación de Netflix [6] como la conversión para vídeos 360º [7] son trabajo ajeno que he importado a mi aplicación.

Los algoritmos de PSNR, SSIM y MSSIM [11], y las fórmulas utilizadas para su entendimiento se obtuvieron del papel [1]. Por otro lado, las fórmulas de S-PSNR y WS-PSNR se obtuvieron de un proyecto Github de vídeos 360 [9].
Los recursos multimedia fueron obtenidos de la base de datos VQA de vídeos omnidireccionales [2]. Aun así se ha utilizado estos datos con el objetivo de fomentar y poner en práctica la investigación sobre la materia.

En el apartado de programas utilizados para el desarrollo, no existe inconveniente legal en el uso de *Python* o el editor Visual Studio Code ya que se han compartido globalmente como *software libre*.

3.7.4. Viabilidad económica

La generación de ingresos no es el principal fin de este proyecto. Se ha enfocado en la meta de generar una solución a la inexistencia práctica de la

actual investigación de aplicaciones de evaluación de vídeo 360º con formato gráfico.

3.8. Análisis de riesgos

Para un proyecto seguro es indispensable valorar la existencia de riesgos que pueden aparecer en el proceso. La generación de errores supone la obtención de datos incorrectos, por lo tanto en este apartado se analizarán los riesgos más notables del proyecto.

Mediante la matriz de probabilidad mostrada en la tabla 27, clasificamos los riesgos relacionando el porcentaje de que suceda con su impacto en el proyecto.

	Impacto		
Probabilidad	Leve	Moderado	Crítico
Alta	Moderado	Importante	Inaceptable
Media	Tolerable	Moderado	Importante
Baja	Aceptable	Tolerable	Moderado

Tabla 27: Matriz de probabilidad

Dependiendo del nivel de riesgo, se debe actuar de una forma diferente:

1. Riesgo **aceptable**: Se puede asumir sin tomar ninguna medida.
2. Riesgo Tolerable: Si el impacto es leve, se puede asumir. Si no, se debe reducir.
3. Riesgo **moderado**: Se debe reducir o transferir su impacto.
4. Riesgo **importante**: Se debe evitar, reducir o transferir el impacto.
5. Riesgo **inaceptable**: Se debe eliminar la acción o el módulo que origina el riesgo. También se puede intentar reducirlo o transferirlo a otras áreas.

Tras esta información, en la tabla 28 se recogen los riesgos más importantes del proyecto. Se puede apreciar como los riesgos inaceptables es necesario aplicar una estrategia de mitigación que eviten el riesgo. Por otra parte, para los tolerables y moderados es necesario aplicar un plan de contingencia que reduzca el impacto.

ID	Descripción	Probabilidad	Impacto	Clasificación
1	La aplicación no se ejecuta	Media	Crítico	Importante
2	Se congela el proceso	Media	Moderado	Moderado
3	No permite leer ni escribir ficheros	Media	Crítico	Importante
4	El usuario no entiende la interfaz	Baja	Leve	Aceptable
5	Pérdida de documentación	Alta	Crítico	Inaceptable
6	Mala estimación temporal del proyecto	Baja	Crítico	Moderado

Tabla 28: Estimación de los riesgos importantes

3.8.1. Estrategias de mitigación

Esta técnica se utiliza a priori para reducir la probabilidad de que se generen riesgos. En la tabla 29 se proponen soluciones para reducir las posibilidades de que ocurran.

ID	Medida de mitigación
1	Si surge un fallo al iniciar la ejecución es posible que se deba a la falta de ficheros o módulos externos. El manual de usuario indicará los módulos necesarios para que el usuario los pueda instalar en caso necesario.
2	La aplicación se diseña para procesar una métrica únicamente en un bloque de dos frames cada vez. De esta forma aunque tarde más, es más seguro.
3	Puede surgir debido a la característica multiplataforma de Python. Se debe indicar al usuario cuando se deba hacer manualmente.
5	Utilizar GitHub para subir la aplicación y utilizar Google Docs para escribir la memoria aumenta la seguridad en caso de apagón o pantallazo azul del SO Windows 10. Hacer copias de seguridad cada día en un disco duro externo permite tener controladas versiones y permite trabajar en otro dispositivo sólo con enchufar el dispositivo.

Tabla 29: Estrategias de mitigación

3.8.2. Planes de contingencia

Por otra parte, una vez ya han ocurrido los impactos se tiene que valorar un plan de contingencia. En la tabla 30 se puede observar el plan planteado.

ID	Medida de contingencia
4	Se dispone de un manual para que el usuario pueda buscar sus dudas o seguirlo a modo de tutorial en caso de confusión. En la interfaz hay un botón que abre el manual de usuario.
6	Reducir las expectativas o los objetivos a cumplir puede ser necesario para poder entregar el proyecto a tiempo. Será necesario acortar la parte de obtención de pruebas en caso necesario.

Tabla 30: Plan de contingencia

4. Desarrollo del proyecto

4.1. Tecnologías utilizadas

En este apartado se comentarán las herramientas y tecnologías utilizadas para el desarrollo del proyecto. Asimismo, se justifica el lenguaje de programación elegido y los módulos importados por orden de importancia.

4.1.1. Visual Studio Code

Visual Studio Code es un editor de código fuente *open source* multiplataforma desarrollado por la empresa Microsoft.

Se diferencia del conocido Microsoft Visual Studio por la rapidez y facilidad que da a los desarrolladores frente al complejo entorno de desarrollo integrado.

Se ha decidido implementar a través de programa ya que incluye herramientas útiles tales como finalización de código, terminal integrada, resaltado de sintaxis, depuración, y enlazamiento con Git. Además, la familiaridad con este editor ha sido un gran factor para su elección.

4.1.2. GitHub

En vista a evitar problemas en el proyecto se ha decidido utilizar GitHub para el control de versiones. De esta forma se consigue trabajar más cómodamente al saber que en caso de existir algún problema se puede volver a una versión anterior funcional.

GitHub es una extendida plataforma de control de versiones para facilitar a los desarrolladores el proceso de diseñar una herramienta ya que permite subir a

la nube un historial de cambios en un proyecto mientras se mantiene en el dispositivo la última versión modificada.

Al ser una plataforma social, también se permite compartir rápidamente código de proyectos para la reutilización por parte de otros usuarios.

El control de versiones supone la integridad y seguridad del proyecto, por lo tanto se ha dado gran valor a su uso en el desarrollo de la aplicación. Durante los cursos académicos, y en la actualidad mientras se desarrollan las prácticas curriculares de forma simultánea a este proyecto, siempre se ha dado importancia a la utilización de repositorios.

4.1.3. Visual Paradigm

Visual Paradigm es una herramienta para crear modelos y diagramas. El principal uso en este proyecto es la creación de diagramas de casos de uso, diagrama de clases y diagrama de secuencia.

Tiene herramientas interesantes como la creación del diagrama de clases. Existe la opción de crearlo a través de la importación del código por lo que se acorta el proceso.

4.1.4. Python

Python es un lenguaje de alto nivel, interpretado y orientado a objetos. A través de su semántica dinámica lo convierte en un lenguaje idóneo para desarrollar aplicaciones rápidas con gran legibilidad.

Al ser un lenguaje interpretado se ahorra el paso de compilación, por lo tanto se disminuye notablemente el tiempo que se utiliza para desarrollar la aplicación. También ayuda a la rapidez el hecho de que no se necesita declarar los tipos de los argumentos de las variables.

Se define también como un lenguaje pegamento que conecta componentes y otros elementos del código entre sí. De modo que si se une con diferentes librerías puede llegar a crear herramientas interesantes.

La extensa cantidad de librerías y módulos con los que se consigue crear aplicaciones de gran variedad no tiene ningún tipo de coste.

En este proyecto se han usado varias librerías disponibles gratuitamente para todo el mundo. Más adelante se explica la utilidad de cada módulo o librería con importancia para la aplicación.

La utilización de este lenguaje de programación se debe a las útiles librerías que maneja en el ámbito de la gestión y manipulación de vídeos además de métricas y otros datos obtenibles.

4.1.5. Librería OpenCV

La librería *Open Computer Vision* es una gran librería *open source* de visión artificial creada por Intel. Se enfoca en los ámbitos de aprendizaje máquina, procesado de imagen y actualmente tiene un papel importante en operaciones en tiempo real.

Principalmente se desarrolló para C++, pero en la actualidad se ha incluido en otros lenguajes como Python, Java, JavaScript o Matlab. Su continua actualización se traduce en numerosos tutoriales y documentación

Debido a sus herramientas, las cuales se basan en operaciones matemáticas, se puede utilizar para procesar imágenes o vídeos para el reconocimiento facial y de objetos, análisis de imágenes médicas, procesamiento de vídeo y otros aspectos.

En este proyecto se emplea para el procesamiento de vídeos y obtención de métricas a partir de la lectura de cada fotograma de los vídeos. Para ello se utilizan diferentes módulos dentro de la librería como *Video I/O* [12] y *Image Processing*.

Video I/O es el módulo utilizado para leer y escribir secuencias de imágenes. El principal uso dado en este proyecto ha sido con la clase *VideoCapture* para capturar vídeos desde un archivo.

En el módulo *Image Processing* se encuentran herramientas como la función *GaussianBlur* para añadir un filtro de desenfoque gaussiano a la imagen en el cálculo de MSSIM, o la conversión del espacio de color con la función *cvtColor*.

4.1.6. Módulo NumPy

El principal beneficio de utilizar NumPy son los objetos ndarray. Estos objetos se pueden crear con N dimensiones, por lo que los elementos *array* creados con este módulo pueden ser de cualquier dimensión.

La generación de matrices y las correspondientes operaciones o modificaciones sobre ellas recaen en este mismo módulo.

4.1.7. Módulo Pyplot

Pyplot es un módulo del paquete matplotlib que permite crear gráficas y figuras de forma automática a través de datos obtenidos de listas o arrays.

La función que ha tenido en este proyecto ha sido la de crear la gráfica con los valores de la métrica elegida y su cambio de valores en cada frame.

4.1.8. Módulo Interfaz: Tkinter

Para implementar la interfaz se ha utilizado el módulo Tkinter. Actualmente es el paquete más popular para diseñar interfaces gráficas en Python, es el estándar *de facto* ya que viene de base con la instalación de Windows.

Sus funciones se enfocan en la implementación de widgets Tk y clases para diseñar la GUI de cualquier aplicación Python con herramientas Tk GUI.

Al tratarse del módulo por defecto para implementar la interfaz se puede conseguir mucha documentación sobre los métodos y diversos ejemplos explicativos.

4.1.9. Módulo os

Módulo empleado para abrir ficheros y manipular directorios. En el proyecto se ha dado uso a través de la creación de directorios y eliminación de ficheros. Además, también se ha usado para ejecutar sentencias *if* tras la comprobación de las extensiones de los ficheros.

4.1.10. Módulo glob

En el momento de comparar la extensión de los ficheros de vídeo leídos se ha utilizado la función *glob* del módulo homónimo. Con esta función se crea una lista de directorios coincidentes con el patrón definido.

En el caso de la implementación de la métrica VMAF, el patrón define la extensión de los vídeos a evaluar.

4.1.11. Módulo re

Mediante el módulo *re* se proporcionan operaciones útiles para comparar expresiones regulares. Por medio de las funciones que incluye se permite verificar si una cadena en particular coincide con una expresión regular dada.

4.2. Diseño

4.2.1. Interfaz gráfica

En este apartado se explicará la interfaz y las correspondientes ventanas que componen esta aplicación.

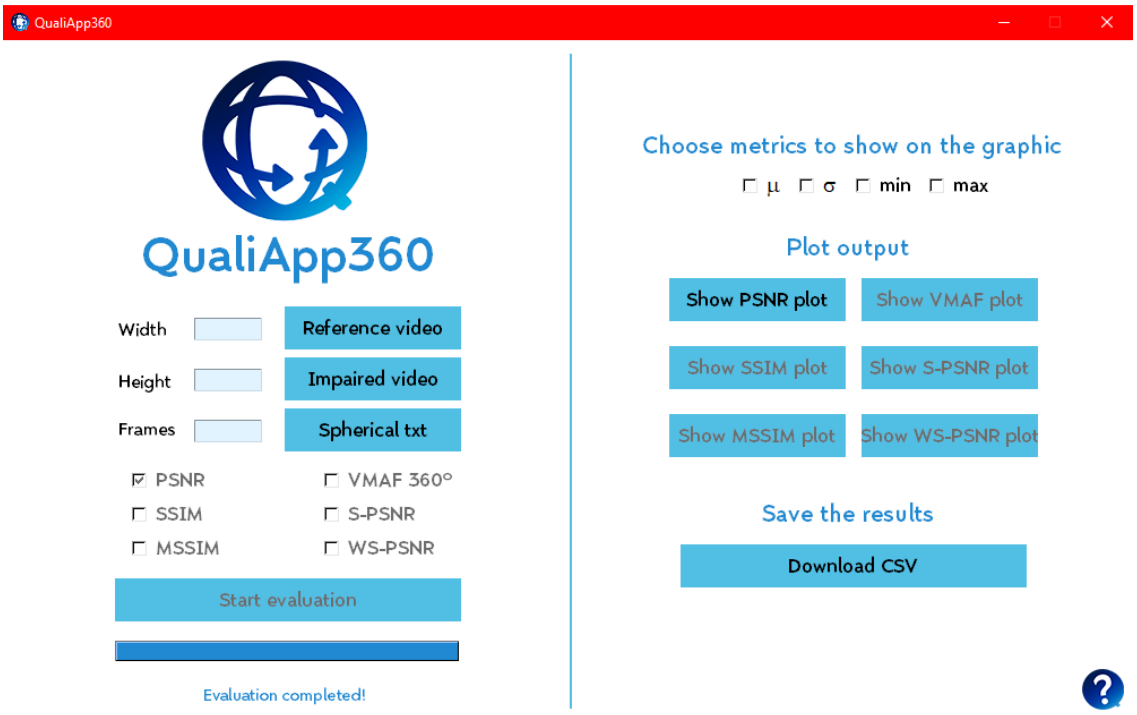

Ilustración 12: Interfaz de la aplicación QualiApp360

Como se puede apreciar en la ilustración 12, se recogen todos los elementos en una sola ventana para así mostrar al usuario todas las funcionalidades, sin que haya nada oculto. Solamente se crean ventanas nuevas para tratar con las gráficas tras el proceso de evaluación.

Aun así, para evitar errores y confusiones por parte del usuario, el programa se ejecuta de forma lineal. Por lo tanto, distintos elementos están desactivados pero visibles hasta que se recojan los datos y pasos necesarios.

Sin embargo, es interesante que el usuario lea y comprenda el manual para evitar problemas. En caso de realizar alguna acción errónea, se indicará a través de una ventana emergente que no se ha podido realizar su petición.

La interfaz se divide en dos zonas distinguidas una de entrada de datos y otra de salida de resultados, ambas se definen en la tabla 31.

Bloque de entrada	Bloque de salida
Icono de la aplicación con el nombre.	Seleccionables para mostrar u ocultar valores en la gráfica.
Introducción de datos necesarios para leer ficheros yuv.	Botones para elegir qué gráfica ver.
Botones para cargar ficheros de tipo yuv, mp4, avi o txt.	Botón para descargar el archivo CSV.
Seleccionables para indicar las métricas a evaluar.	Botón ayuda para abrir el manual.
Botón para iniciar la evaluación.	
Barra de progreso que indica al usuario el porcentaje finalizado.	

Tabla 31: Separación de la interfaz por bloques

Bloque de entrada
Analizando de arriba a abajo la interfaz, primeramente se encuentra el logo con el nombre. De esta forma se sabe en todo momento qué aplicación se está utilizando.

A continuación se pide al usuario la introducción de datos y de ficheros, abriendo una ventana emergente para buscar el directorio como se ve en la ilustración 13. Los formatos de vídeo permitidos son mp4, avi y yuv.

Su utilización depende del tipo de archivo a evaluar. Si se trata de archivos con extensión yuv, es necesario que el usuario añada valores a los campos de la resolución y el número de frames. Esto se debe a que para poder leer un archivo yuv, es necesario saber estos datos de antemano. En cambio si el archivo es un vídeo mp4 o avi, no es necesario introducir estos datos ya que se recogen automáticamente al cargar el vídeo.

El botón de fichero es solamente necesario cuando se quiere evaluar con la métrica S-PSNR ya que es indispensable para su cálculo.

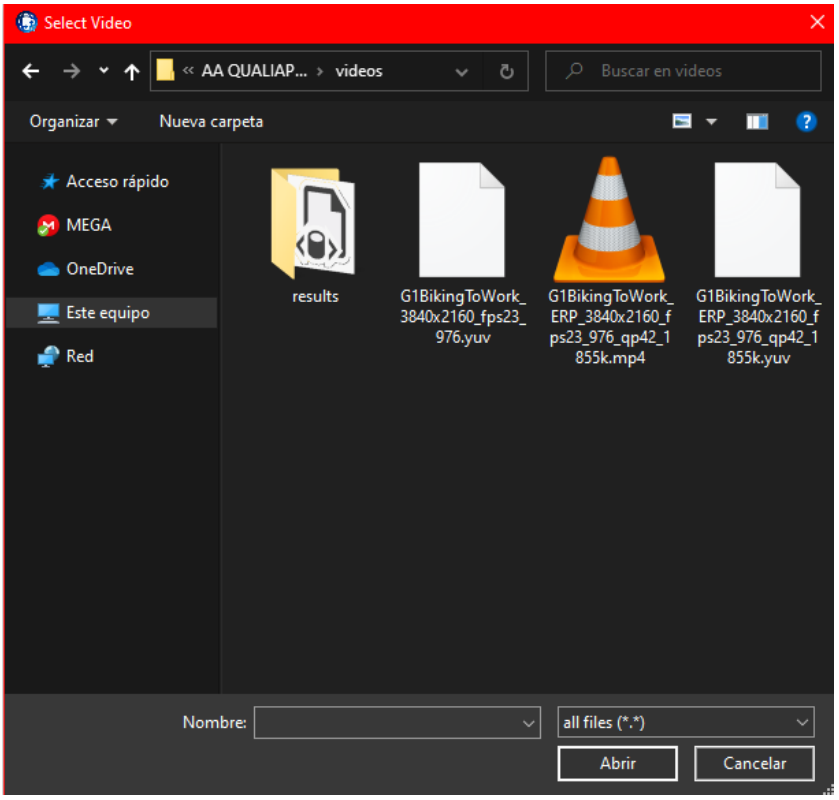

Ilustración 13:Selección de vídeo desde directorio

Por otra parte, una vez se han cargado todos los ficheros necesarios se habilitan las métricas. En este punto, el usuario puede elegir las métricas que quiera teniendo en cuenta que solo se habilitarán las disponibles atendiendo a los ficheros cargados. Es decir, si no se ha cargado el fichero de texto no se habilitará S-PSNR, una vez cargado estará disponible para su elección.

Tras elegir una métrica, al pulsar el botón de iniciar evaluación se ejecutarán las seleccionadas. La barra de progreso indica el porcentaje evaluado. En casos como VMAF parece que no avance ya que tiende a tardar más. Al acabar saldrá un mensaje indicándolo debajo de la barra de progreso.

Bloque de salida

En la parte derecha tenemos varias zonas conectadas entre sí. Arriba tenemos diferentes valores que seleccionar para que se muestren al observar la gráfica. Las gráficas se muestran solamente de las métricas evaluadas, como se aprecia en el caso de la imagen 12. Se observa que el botón *Show PSNR plot* es el único habilitado ya que en este caso PSNR ha sido la única métrica seleccionada.

El usuario puede ver una gráfica con los datos una vez evaluados los vídeos tal como se ve en la ilustración 14. A través de los botones del bloque derecho, se

abre una nueva ventana que muestra una gráfica comparando la puntuación de dicha métrica con el número de frames de los vídeos.

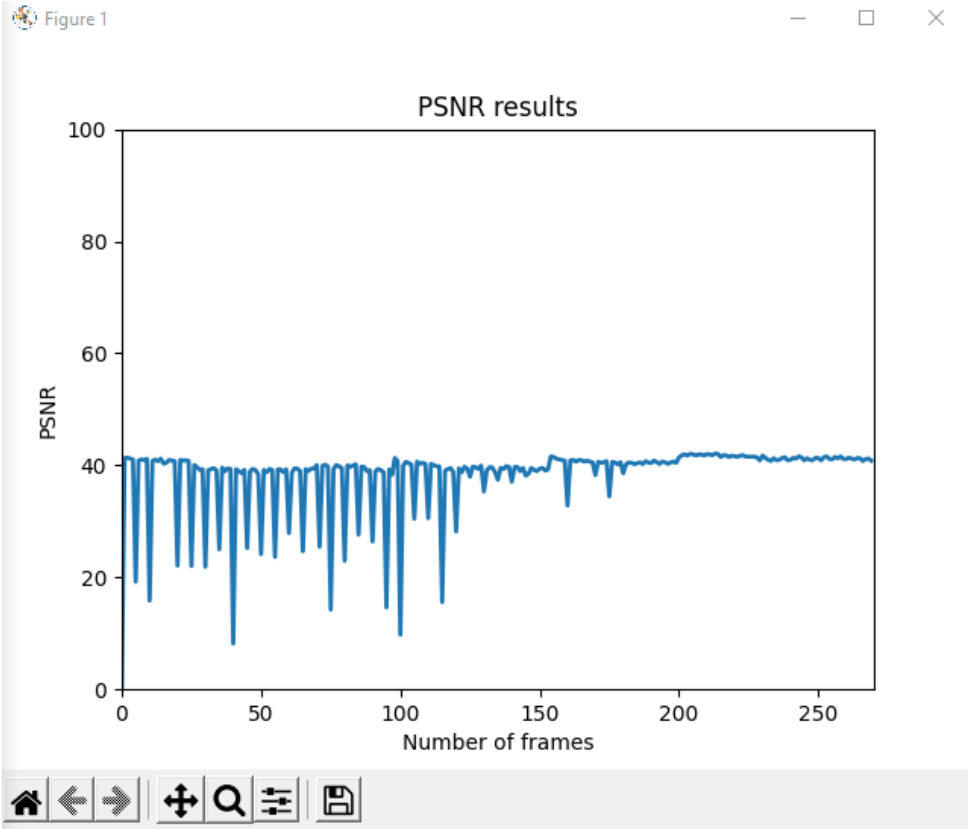

Ilustración 14: Visualización de la gráfica obtenida con la evaluación de calidad

En esta pantalla el usuario puede ampliar y moverse por toda la gráfica a través de los botones del menú inferior. Además, tiene otras funciones como permitir guardar la gráfica o volver a la posición inicial.

También es posible descargar un archivo CSV con los valores resultantes de cada frame. El resultado se puede comprobar en la ilustración 15.

Y por último, abajo a la derecha se encuentra la ayuda para el usuario. Este botón abre el manual de usuario para resolver posibles dudas que hayan surgido.

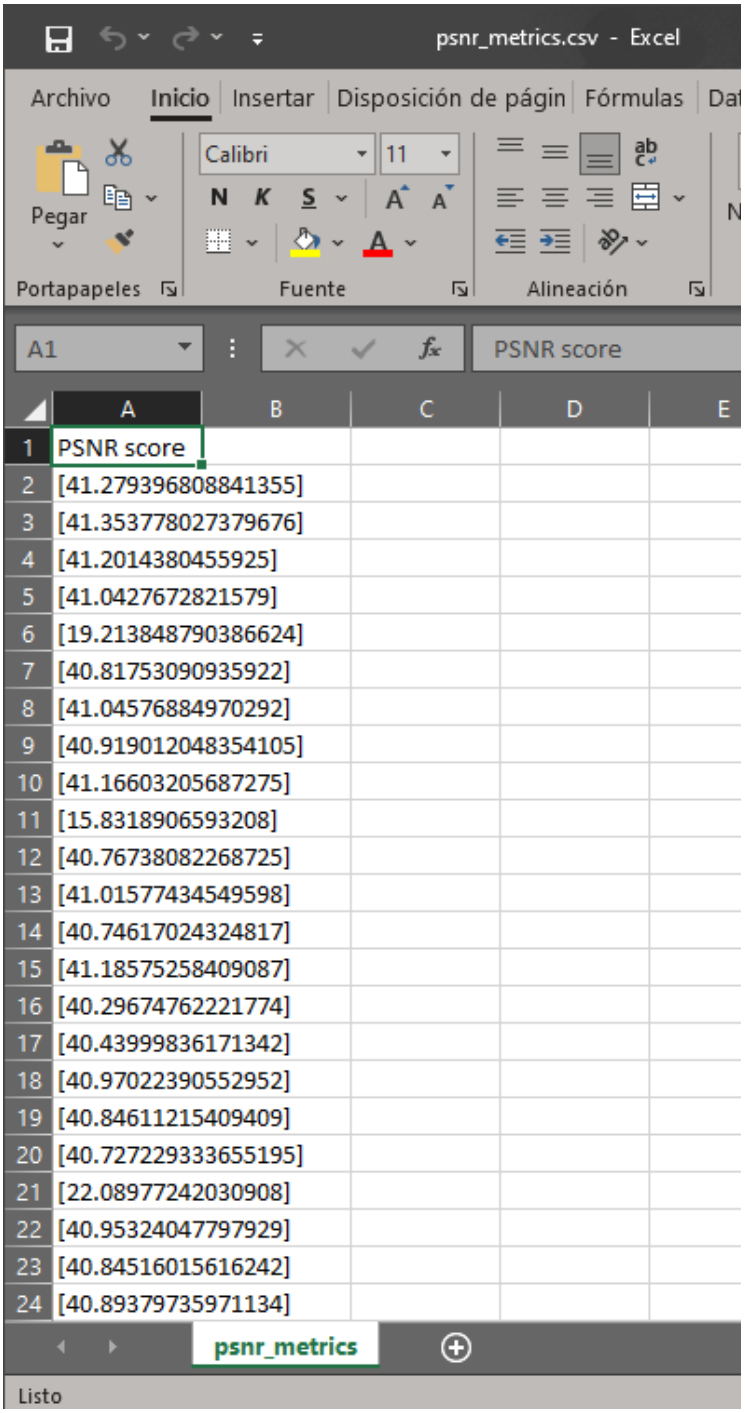

Ilustración 15: Muestra del CSV obtenido con los resultados de la evaluación

4.3. Implementación

Una vez mostrada la interfaz, el siguiente paso es describir las funciones más importantes de la aplicación.

4.3.1. Funciones importantes

En este apartado se definen todas las partes del código importantes que definen el funcionamiento general de la aplicación. Se van a tratar principalmente todas las métricas y su definición.

4.3.1.1. getPSNRdata

A través de ***getPSNRdata(frameVid1, frameVid2)*** se ha implementado la función que calcula el PSNR de cada frame. La función es llamada en la función principal iterativamente hasta que recorre todos los fotogramas del archivo.

```
# PSNR
def getPSNRdata(I1, I2):

    ix_iy = absdiff(I1, I2)                 # absolute difference between the two frames
    ix_iy = float32(ix_iy)                  # cannot make a square on 8 bits
    ix_iy_2 = ix_iy ** 2                    # (Ix - Iy)^2
    summationPSNR = ix_iy_2.sum()           # Sum of (Ix - Iy)^2

    if(summationPSNR <= 1e-10):             # check if the value is too small
        return 0                            # then return zero
    else:
        shape = I1.shape                    # shape of an array is the number of elements
        cij = shape[0] * shape[1] * shape[2] # c * i * j
        mse = summationPSNR / (cij)         # sum of (Ix - Iy)**2 / c * i * j
        maxI = 255                          # 1 byte per pixel per channel = 255
        psnr = 10.0 * log10((maxI ** 2) / mse)
        return psnr
```

Ilustración 16: Implementación de la métrica PSNR

Siguiendo la formulación de la ilustración 16, $I1$ e $I2$ se toman como valores de los fotogramas con valor de dos dimensiones (i, j) que se compone con un número c de canales. En este proceso se calculan los valores del sumatorio a través de la diferencia absoluta de los dos fotogramas a comparar elevado a dos.

$$\mathrm{MSE} = \frac{1}{\mathrm{c} * \mathrm{i} * \mathrm{j}} \sum (\mathrm{I}_1 - \mathrm{I}_2)^2 \qquad (17)$$

Tras calcular el sumatorio se compara con un valor pequeño como puede ser $1e - 10$. Si el resultado obtenido es mayor que este número, se prosigue con el cálculo de $c * i * j$ tomados del fotograma del archivo de referencia.

Una vez recogidos los valores anteriormente calculados, obtenemos el error cuadrático medio (MSE) . Tras calcular el *MSE* como se muestra en la ecuación 17, se puede obtener el PSNR como se ve en la ecuación 18.

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right) \tag{18}$$

La transformación del valor a una escala logarítmica se realiza ya que los valores de los píxeles tienen un rango dinámico muy amplio.

El valor MAX_I es el máximo valor válido para cada píxel. Al tratar con imágenes simples de 1 byte por píxel por canal, el valor de MAX es 255.

Cuando se comparan dos imágenes idénticas, *MSE* tiene valor de 0. Por lo tanto, al tener que dividir por 0 el resultado es inválido. En este caso, el PSNR es indefinido y se necesitará evaluar de otra forma separada.

4.3.1.2. *getMSSIMdata*

Mediante ***getMSSIMdata(frameVid1, frameVid2)*** se genera un índice de similitud de cada canal de la imagen. El rango de este valor se encuentra entre 0 y 1, siendo 1 el ajuste ideal. Esta función también se ejecuta dentro del bucle iterativo que recorre todos los fotogramas de los vídeos.

```
# MSSIM
def getMSSIMdata(I1, I2):
    K1 = 0.01
    K2 = 0.03
    L = 255
    C1 = (K1*L)**2
    C2 = (K2*L)**2

    i1 = float32(I1)
    i2 = float32(I2)
    i1_2 = i1 ** 2
    i2_2 = i2 ** 2
    i1_i2 = i1 * i2

    mu_x = cv.GaussianBlur(i1, (11, 11), 1.5)
    mu_y = cv.GaussianBlur(i2, (11, 11), 1.5)

    mu_x_2 = mu_x * mu_x
    mu_y_2 = mu_y * mu_y
    mu_x_mu_y = mu_x * mu_y

    sigma_x_2 = cv.GaussianBlur(i1_2, (11, 11), 1.5)
    sigma_y_2 = cv.GaussianBlur(i2_2, (11, 11), 1.5)
    sigma_x_y = cv.GaussianBlur(i1_i2, (11, 11), 1.5)

    sigma_x_2 -= mu_x_2
    sigma_y_2 -= mu_y_2
    sigma_x_y -= mu_x_mu_y

    #SSIM(x,y)
    step1 = 2 * mu_x_mu_y + C1
    step2 = 2 * sigma_x_y + C2
    step3 = step1 * step2                   # t3 = ((2*mu_x_mu_y + C1).*(2*sigma_x2 + C2))
    step4 = mu_x_2 + mu_y_2 + C1
    step5 = sigma_x_2 + sigma_y_2 + C2
    step6 = step4 * step5                   # t1 =((mu_x_2 + mu_y_2 + C1).*(sigma_x_2 + sigma_y_2

    ssim_map = cv.divide(step3, step6)      # ssim_map =  t3/t1;
    mssim = cv.mean(ssim_map)               # mssim = average of ssim map
    return mssim
```

Ilustración 17: Implementación de la métrica SSIM

Como dato a considerar, el desenfoque gaussiano utilizado en la función de SSIM (mostrada en la ilustración 17) es más costoso en tiempo que PSNR por lo tanto tardará mucho más en obtener los mismos resultados.

En la llamada a *GaussianBlur* se encuentra sigma, que es el valor que se encarga del espesor o grosor de la función núcleo. Un valor alto de sigma es indicador de un mayor difuminado en un radio más amplio.

En comparación con PSNR que calcula cada frame, MSSIM solamente se calcula en los fotogramas donde el valor obtenido de PSNR cae por debajo de un valor preestablecido.

4.3.1.3. getSSIMdata

La función ***getSSIMdata(frameVid1, frameVid2)*** recoge de forma iterativa una imagen del vídeo de referencia junto al vídeo distorsionado y devuelve en cada fotograma el SSIM correspondiente. Si el resultado es alto, indica que las imágenes a comparar con más similares.

Debido a la implementación de SSIM en la función *getMSSIMdata* (ilustración 18)*, se acorta código y se agiliza* el proceso llamando a la función SSIM ya existente de la librería OpenCV.

```
# SSIM
def getSSIMdata(I1, I2):

    # Convert the images to grayscale
    gray1 = cv.cvtColor(I1, cv.COLOR_BGR2GRAY)
    gray2 = cv.cvtColor(I2, cv.COLOR_BGR2GRAY)

    ssim_value = ssim(gray1, gray2)

    return ssim_value
```

Ilustración 18: Implementación de la métrica MSSIM

4.3.1.4. getVMAFdata

El código de esta métrica fue obtenido desde el algoritmo de Netlix [6] y se ha modificado de forma leve para ser añadido a la aplicación. Se van a comentar primordialmente las modificaciones realizadas en el código.

Esta función se ejecuta fuera del bucle que recorre los fotogramas ya que el propio proceso contiene bucles para recorrer los vídeos.

Inicialmente tal como se ve en la ilustración 19 se definen los argumentos de entrada desde la terminal. Sin embargo anteriormente su funcionamiento era a

través de la terminal, por lo que actualmente los argumentos son variables que se recogen desde la clase principal.

```
def getVMAFdata(self, num_frames):
    parser = argparse.ArgumentParser()
    parser = argparse.ArgumentParser(description='VMAF ODV')

    parser.add_argument('--w', default=self.resolucion[0], help="resolution width of a given videos")
    parser.add_argument('--h', default=self.resolucion[1], help="resolution height of a given videos")
    parser.add_argument('--f', default=num_frames, help="number of frame")
    parser.add_argument('--r', default=self.dir1,  help="reference video")
    parser.add_argument('--c', nargs='?', type=int, default=15, action="store", help="cell number")

    user_input = parser.parse_args()

    user_input.r = os.path.basename(self.dir1)[:-4]
   # download the zip file and extract it
    try:
        if(os.path.isfile(file_name)!=True):
            wget.download('http://v-sense.scss.tcd.ie/Datasets/' + file_name, file_name)
            extract_process(file_name)
        if(os.path.isfile(thirdParty_zip)!=True):
            wget.download('http://v-sense.scss.tcd.ie/Datasets/' + thirdParty_zip, thirdParty_zip)
            extract_process(thirdParty_zip)
    except:
        print("No file to be downloaded!")

    # generate patches for each video file
    try:
        types = (video_folder + '*.mp4', video_folder + '*.yuv')
        files =[]
        [files.extend(glob.glob(_type)) for _type in types]
        for video in files:
            # create a folder for each video
            create_dir(video_folder + 'results/' + os.path.basename(video)[:-4] + '/')
            # generate patches
            generate_patches(video)
    except:
        print()

    # compute vmaf scores
```

Ilustración 19: Implementación de la métrica VMAF 360.

Se han eliminado las funciones de conversión de vídeo que se encontraban en la aplicación original. Esto se debe a que este proyecto prioriza la evaluación de métricas y se originan problemas al recoger los vídeos desde directorio en lugar de su lectura por línea de comando.

El resto del código está sin modificar por lo que si se está interesado en leer más información y ver su versión completa.

4.3.1.5. getSPSNRdata

Para el cálculo de la puntuación de S-PSNR (véase ilustración 20), es necesario que los vídeos se complementen con un archivo de texto con puntos esféricos. El archivo utilizado para ello es *sphere_655362.txt* y se puede encontrar en el github del proyecto vídeos 360 de Samsung [9].

Tras la lectura del fichero de texto, se recogen el número de esferas y los puntos de cada una de ellas. La función recorre el número de esferas y convierte los puntos de dicha esfera en un rectángulo. Luego se pasa a una función llamada *filter*, donde se interpola.

Una vez obtenidos los datos interpolados, se calcula la diferencia absoluta y se añade al sumatorio. Tras analizar toda la esfera se divide entre el número de esferas y se realiza la ecuación PSNR. En el caso de obtener un valor pequeño, se añade una condición para que el resultado no dé cero.

```
# S-PSNR
def getSPSNRdata(self, sph_points, n_sphere, src, dst):
    sum = 0
    i = 0
    cart = []
    w = self.global_width
    h = self.global_height

    while(i < n_sphere):

        self.frame_actual += 1
        self.progressBar['value'] = self.frame_actual
        self.root.update_idletasks()

        cart = convSphToCart(sph_points[i], cart)
        x, y = convCartToRect(w, h, cart)

        x -= 0.5
        y -= 0.5

        val1 = filter(src, x, y, w, h, w)
        val2 = filter(dst, x, y, w, h, w)
        diff = val1 - val2
        diff = absdiff(diff)
        sum += diff * diff

        i += 1

    sum /= n_sphere

    if sum == 0:
        sum = 100
    else:
        sum = 10*log10(255*255 / sum)

    return sum
```

Ilustración 20: Implementación de la métrica S-PSNR

4.3.1.6. getWSPSNRdata

La función getWSPSNRdata se encuentra fuera del bucle principal ya que ejecuta un bucle dentro de su propia llamada.

Se inicia en la clase principal (ilustración 21) creando una matriz rellenada con unos. Tras su creación, dentro de un bucle que recorre el alto de los vídeos a comparar se va introduciendo la variable de peso en la misma matriz.

Una vez se ha recorrido toda la matriz se hace llamada a la función getWSPSNRdata pasándole la matriz anteriormente creada y modificada además de los archivos de vídeo y el número de fotogramas.

```
if(self.isWSPSNRchecked() == True):
    self.frame_actual = 0
    j = 0

    weightMap = np.ones((self.global_width, self.global_height), dtype=np.float64)

    while(j < self.global_height):
        weight = cos((j - (self.global_height/2 - 0.5)) * math.pi / self.global_height)
        weightMap[j] = weightMap[j] * weight
        j+=1

    totalWS, durationWS = getWSPSNRdata(self, captRefrnc, captUndTst, num_frames, weightMap)

    #WS-PSNR data print
    global_ws_psnr = totalWS / num_frames
    avg_time_ws_psnr = durationWS / num_frames

    print("Global WS-PSNR: {}".format(round(global_ws_psnr, 4)), end=" ")
    print()
    print("Average Time: {}".format(round(avg_time_ws_psnr, 4)), end=" ")
```

Ilustración 21: Inicialización de la métrica WS-PSNR

Al entrar en la función, como se aprecia en 21, se crean nuevas matrices de unos con dimensión igual a los vídeos a evaluar. Se crean variables que toman el valor del frame de los vídeos y se les hace la conversión de color mediante la línea resaltada de la ilustración 22 en la función ReadWS.

```
def ReadWS(self, frame):
    if frame is not None:
        aux_frame = cv.cvtColor(frame, cv.COLOR_BGRA2YUV_I420)
        cv.rectangle(aux_frame, (0, 0), (self.global_width, self.global_height), (255,0,0))
    return frame
```

Ilustración 22: WS-PSNR, lectura de frames

El código de color COLOR_BGRA2YUV_I420 indica que hace conversión de BGR (Blue, Green, Red) a YUV (Intensity, Hue, Value) con formato 4:2:0.

Al entrar en el bucle, la barra de progreso se activa y se llama otra vez a la función de lectura ReadWS para actualizar los valores.

La iniciación del tiempo recoge la duración de procesado del cálculo de la métrica y en cada iteración va introduciendo su valor en una variable que más tarde se devolverá a la clase principal.

Se calcula la diferencia entre los dos frames a comparar y weightedDiffAux, la matriz resultante, se eleva a dos. Luego se multiplica esta misma matriz con weightMap, la matriz originada en la clase principal, introduciendo el resultado en weightedDiffAux.

En weightedDiff introducimos el valor obtenido de multiplicar la matriz weightedDiffAux por el escalar 100000. Estos cálculos a continuación se reúnen para calcular WMSE y WSPSNR.

WMSE se obtiene con la división de los sumatorios de las matrices weightedDiff y weightMap. Tras eso se divide el resultado por el escalar 100000.

Para WSPSNR se utiliza una función similar a PSNR, en este caso se utiliza WMSE en lugar de MSE.

Una vez calculado todo, se suman los valores a las variables generales. Se incrementa el número de frames y se vuelve a ejecutar el bucle. Al finalizar devuelve la suma de los WS-PSNR de todos los frames mediante la variable *total* y la duración total de procesado por medio de la variable *duration.*

Tras su ejecución se calcula el WS-PSNR global y el tiempo medio por frame tal como se observa en la ilustración 23.

```
def getWSPSNRdata(self, I1, I2, i, weightMap):

    total_frames = 0
    total = 0
    duration = 0

    weightedDiff = np.ones((self.global_width, self.global_height), dtype=np.float64)
    weightedDiffAux = np.ones((self.global_width, self.global_height), dtype=np.float64)
    diffMap = np.ones((self.global_width, self.global_height), dtype=np.float64)
    srcImg = np.ones((self.global_width, self.global_height), dtype=np.float64)
    dstImg = np.ones((self.global_width, self.global_height), dtype=np.float64)

    srcImg = ReadWS(self, I1)
    dstImg = ReadWS(self, I2)

    while(total_frames < i):
        self.progressBar['value'] = total_frames
        self.root.update_idletasks()

        srcImg = ReadWS(self, srcImg)
        dstImg = ReadWS(self, dstImg)

        start = time.time()

        cv.absdiff(srcImg, dstImg , diffMap)
        cv.pow(diffMap, 2, diffMap)

        weightedDiffAux = np.multiply(diffMap, weightMap)
        weightedDiff = np.multiply(weightedDiffAux, 100000)

        WMSE = sum(weightedDiff[i]) / sum(weightMap[i]) / 100000
        WSPSNR = 10 * log10(255 * 255 / (WMSE + sys.float_info.epsilon))

        end = time.time()
        t = end - start

        print("Frame: {} WS-PSNR score: {} Time: {}".format(total_frames, WSPSNR, t, end=" "))
        print()

        total += WSPSNR
        duration += t
        total_frames += 1

    return total, duration

def ReadWS(self, frame):
    if frame is not None:
        aux_frame = cv.cvtColor(frame, cv.COLOR_BGRA2YUV_I420)
        cv.rectangle(aux_frame, (0, 0), (self.global_width, self.global_height), (255,0,0))
    return frame

def VideoCaptureYUV(self, filename, width, height):
    self.height = height
    self.width = width
    self.frame_len = self.width * self.height * 3 / 2
    # Open '*.yuv' as a binary file.
    self.f = open(filename, 'rb')
    self.shape = (int(self.height*1.5), self.width)
```

Ilustración 23: Implementación de la métrica WS-PSNR

4.3.2. Obstáculos en el desarrollo

La decisión de utilizar un repositorio fue tardía y conlleva a la pérdida de tiempo al no ser posible volver a un punto anterior donde la aplicación funcionase correctamente. Esto ha llevado a desaprovechar los días que se enfocaron a programar y alargó el proceso de implementación.

Por otra parte, el solapamiento del bloque de implementación y pruebas con el progreso de las prácticas curriculares supuso extender el tiempo inicial definido para la finalización del desarrollo.

Otro problema externo al proyecto y a la alumna fue la lluvia. A causa de los altos niveles de precipitaciones en diferentes días durante el proceso se sufrieron apagones en la vivienda de la alumna. Gracias a la elección de Google Docs como editor de texto, en estos casos se pudo proseguir con el desarrollo de la memoria. En cambio, al no disponer de la última versión del código en un dispositivo portátil se ralentizó el progreso hasta que se calmó la lluvia.

5. Pruebas y resultados

5.1. Comparativa entre métricas

5.1.1. Puntuaciones obtenidas

A continuación se muestran los resultados de las métricas obtenidas, en varios casos como el resultado se da por fotograma se ha recortado y se indican los primeros y últimos valores. Se han hecho pruebas de las métricas PSNR, SSIM, MSSIM y VMAF 360.

En PSNR los resultados obtenidos y mostrados en el CSV se muestran a través de decibelios. Debido a que la escala de puntuación va de 0 a 100, se aprecia que el vídeo a comparar tiene una mala calidad.

PSNR score

[41.279396808841355]

[41.353778027379676]

[41.2014380455925]

[41.0427672821579]

[19.213848790386624]

[40.81753090935922]

[41.04576884970292]

[40.919012048354105]

[41.16603205687275]

…

[41.034124977543414]

[41.274817794487774]

[41.252770759221505]

[40.77099793193773]

[41.093125281737464]

[41.21362507743672]

[40.83214201714209]

En SSIM los valores obtenidos son muy buenos. Cabe recordar que el rango de puntuación de esta métrica es de 0 a 1.

SSIM: 1.0
SSIM: 0.9839886978934393
SSIM: 0.9844614292828849
SSIM: 0.9838381202624427
SSIM: 0.9822220276959499
SSIM: 0.9558991700118331
SSIM: 0.9817723802231727
SSIM: 0.9824288939338717
...
SSIM: 0.9808833001485944
SSIM: 0.9800167266026544
SSIM: 0.9779714139628409
SSIM: 0.9802757618302247
SSIM: 0.9800387386941052
SSIM: 0.9772706454828439
SSIM: 0.9800965513478103
SSIM: 0.9803762281130028
SSIM: 0.9782338552386881

En MSSIM los resultados se dividen por los tres canales de RBG: rojo, azul, y verde. Para observar su puntuación sigue el mismo rango de 0 a 1 que SSIM.

(1.0, 1.0, 1.0, 0.0)
MSSIM: R 100.0% G 100.0% B 100.0%

(0.9352865459337856, 0.9656528720533811, 0.9782366591512762, 0.0)
MSSIM: R 97.82% G 96.57% B 93.53%

(0.9344269520180086, 0.9669472776272614, 0.9792716326913248, 0.0)
MSSIM: R 97.93% G 96.69% B 93.44%

(0.932900249064933, 0.9656245066444092, 0.9788454596032404, 0.0)
MSSIM: R 97.88% G 96.56% B 93.29%

(0.9317532405479375, 0.9636035320768839, 0.9763913737541781, 0.0)
MSSIM: R 97.64% G 96.36% B 93.18%

(0.9005599912318532, 0.9338855171830942, 0.9468855014028542, 0.0)
MSSIM: R 94.69% G 93.39% B 90.06%
…

(0.9437805652070825, 0.9808600358767686, 0.9778666576929548, 0.0)
MSSIM: R 97.79% G 98.09% B 94.38%

(0.9443886155553824, 0.97926499664553, 0.9760307612628222, 0.0)
MSSIM: R 97.6% G 97.93% B 94.44%

(0.9404964281902422, 0.9783414907659395, 0.9754787784742399, 0.0)
MSSIM: R 97.55% G 97.83% B 94.05%

(0.9381457991097117, 0.9763884534073919, 0.9736834930430583, 0.0)
MSSIM: R 97.37% G 97.64% B 93.81%

(0.9429660973968518, 0.9752416812588214, 0.9746203444189446, 0.0)
MSSIM: R 97.46% G 97.52% B 94.3%

(0.9380359731646339, 0.9761837113989276, 0.9744547281463659, 0.0)
MSSIM: R 97.45% G 97.62% B 93.8%

En VMAF los resultados proporcionados se muestran por modelo utilizado y finalmente se calcula el VI-VMAF omnidireccional. A modo de recordatorio, el rango de calificaciones de esta métrica es de 0 a 100. En vistas de que el resultado dado es 70.9078, se puede ver que la calidad del vídeo es buena.

VMAF score, model 0001 = 69.264802
VMAF score, model 0002 = 67.288284
VMAF score, model 0003 = 67.369409

VMAF score, model 0004 = 67.183672
VMAF score, model 0005 = 68.215428
VMAF score, model 0006 = 71.631540
VMAF score, model 0007 = 69.855490
VMAF score, model 0008 = 70.943504
VMAF score, model 0009 = 70.499033
VMAF score, model 0010 = 69.983872
VMAF score, model 0011 = 68.375021
VMAF score, model 0012 = 69.893091
VMAF score, model 0013 = 68.215926
VMAF score, model 0014 = 67.788119
VMAF score, model 0015 = 71.287008
VMAF score, model 0016 = 63.883339
VMAF score, model 0017 = 69.569446
VMAF score, model 0018 = 71.299237
VMAF score, model 0019 = 71.028022
VMAF score, model 0020 = 70.602252

::.. 360 VI-VMAF score for G1BikingToWork_3840x2160_fps23_976 = 70.9078

WS-PSNR también muestra un resultado por cada frame indicando tanto su puntuación como el tiempo requerido para el cálculo en ese frame.

Frame: 0 WS-PSNR score: 48.1308036086791 Time: 0.08234739303588867
Frame: 1 WS-PSNR score: 48.1308036086791 Time: 0.08534002304077148
Frame: 2 WS-PSNR score: 48.1308036086791 Time: 0.0818483829498291
Frame: 3 WS-PSNR score: 48.1308036086791 Time: 0.08533978462219238
Frame: 4 WS-PSNR score: 48.1308036086791 Time: 0.07585859298706055
Frame: 5 WS-PSNR score: 48.1308036086791 Time: 0.09282517433166504
Frame: 6 WS-PSNR score: 48.1308036086791 Time: 0.08833599090576172
Frame: 7 WS-PSNR score: 48.1308036086791 Time: 0.09981417655944824
Frame: 8 WS-PSNR score: 48.1308036086791 Time: 0.10330772399902344
Frame: 9 WS-PSNR score: 48.1308036086791 Time: 0.10779953002929688
Frame: 10 WS-PSNR score: 48.1308036086791 Time: 0.09582042694091797
...
Frame: 390 WS-PSNR score: 48.1308036086791 Time: 0.0918271541595459
Frame: 391 WS-PSNR score: 48.1308036086791 Time: 0.09232759475708008
Frame: 392 WS-PSNR score: 48.1308036086791 Time: 0.09083008766174316
Frame: 393 WS-PSNR score: 48.1308036086791 Time: 0.12476754188537598
Frame: 394 WS-PSNR score: 48.1308036086791 Time: 0.13924074172973633
Frame: 395 WS-PSNR score: 48.1308036086791 Time: 0.1686849594116211
Frame: 396 WS-PSNR score: 48.1308036086791 Time: 0.1292581558227539
Frame: 397 WS-PSNR score: 48.1308036086791 Time: 0.10630154609680176
Frame: 398 WS-PSNR score: 48.1308036086791 Time: 0.10879755020141602
Frame: 399 WS-PSNR score: 48.1308036086791 Time: 0.10630083084106445

5.1.1.1. Análisis de dichas métricas

PSNR no es una buena métrica para obtener datos, ya que frente a las demás ha dado un resultado notablemente inferior. Su puntuación media en PSNR es de 40dB, en otras métricas como SSIM daba una puntuación casi perfecta de 0,9 aproximadamente.

VMAF 360 es la mejor para analizar ya que aunque utilice más tiempo de procesado, da puntuaciones más específicas basándose en modelos y en una puntuación media general de la comparación entre los dos ficheros.

5.1.2. Pruebas de rendimiento

Hay una notable diferencia entre las métricas PSNR, MSSIM y SSIM frente a las métricas VMAF 360, S-PSNR y WS-PSNR. Tanto por tiempo de ejecución como por porcentaje de CPU utilizado. El tiempo estimado para cada métrica se muestra en la tabla 32.

Nombre	Tipo de vídeo	Número de frames	Tiempo total
PSNR	Bidimensional	300	6 segundos
MSSIM	Bidimensional	300	33 segundos
SSIM	Bidimensional	300	40 segundos
VMAF 360	360º	400	30 minutos
S-PSNR	360º	400	2 minutos
WS-PSNR	360º	400	47 segundos

Tabla 32: Tiempo estimado de cada métrica

5.1.3. Uso de CPU y de memoria

Como se aprecia en la ilustración 24, el porcentaje de uso de CPU y memoria en reposo está en unos valores bajos.

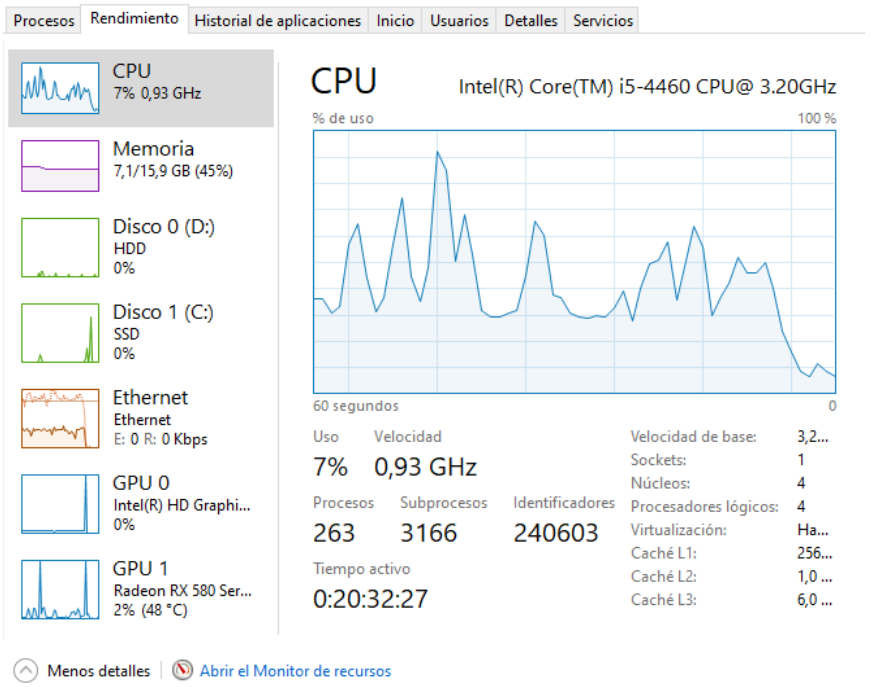

Ilustración 24: Uso de CPU en reposo

Al ejecutar las métricas, en el caso de la ilustración 25 se estaba ejecutando VMAF 360, el rendimiento baja por el alto porcentaje de uso de CPU llegando a ralentizar el dispositivo y bloqueando la aplicación hasta la finalización de la evaluación.

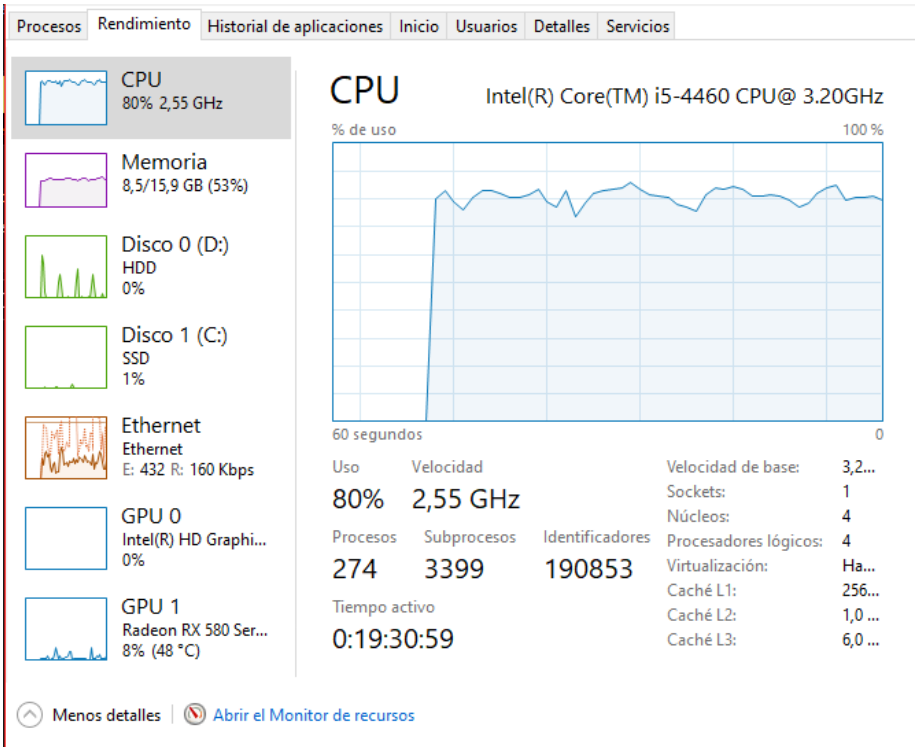

Ilustración 25: Uso de CPU ejecutando VMAF 360

5.2. Pruebas de usabilidad

En este apartado se va a analizar la usabilidad, es decir, la capacidad de la aplicación a ser aprendida y utilizada por el usuario. En el momento de realizar las pruebas, se han pedido a usuarios con distinto conocimiento informático para tener un abanico de respuestas realistas. Se han obtenido datos directos, al preguntarle a la persona, e indirectos, al ver gestos durante el uso de la aplicación.

5.2.1. Test SUS

Para evaluar la usabilidad se ha realizado el Test SUS (System Usability Scale). Este método se define en 10 afirmaciones que se deben puntuar del 1 al 5 por los usuarios a evaluar. Las cuestiones [13] se encuentran definidas de forma simple y sin ambigüedades.

1. Pienso que daría un uso frecuente a este sistema.
2. Encuentro la aplicación innecesariamente compleja.
3. Siento que el sistema es fácil de utilizar.
4. Creo que necesitaría ayuda de una persona que sepa usar este programa.
5. Siento que varias funciones están bien integradas.
6. Creo que hay demasiadas inconsistencias en este sistema.
7. Imagino que este sistema sería fácil de aprender para muchas personas.
8. Siento que el programa es incómodo de usar.
9. Siento que el programa es cómodo de usar.
10. Siento que necesito aprender muchas cosas antes de poder utilizarlo.

El método de puntuación se basa en sumar cinco menos la puntuación en los pares y en sumar la puntuación de la afirmación menos uno en los impares. Dicho esto, la puntuación final varía en un rango de 0 a 40 puntos.

$$puntuación = \sum 5 - pares + \sum impares - 1 \quad (19)$$

5.2.2. Resultados

Se han encuestado a seis personas, de las cuales la mitad no tienen conocimientos avanzados en informática y las otras tres personas son estudiantes de ingeniería.

Para obtener el resultado final, se deben sumar todas las respuestas y multiplicarlo por 2’5. Esto convierte el rango de valores de 0-40 a 0-100.

$$Test\ SUS\ = 2,5\ (\ \sum\ impares\ -\ 1\ +\ \ \sum 5\ -\ pares) =$$

$$2,5\ *\ (1'3\ +\ 3'1\ +\ 3'6\ +\ 2'5\ +\ 3'6)\ +\ (3'4\ +\ 3'7\ +\ 3'8\ +\ = 2,5\ *\ 29'2\ =\ 73\% \quad (20)$$

Analizando el resultado frente a la puntuación SUS [14], la aplicación no supera el porcentaje de la media definida en 76,2%. El resultado se ha visto notablemente afectado por la última cuestión que trata de si el usuario necesitaría conocimientos antes de utilizar la aplicación. Se debe en parte al ser un programa enfocado a una tecnología tan novedosa y todavía en investigación.

6. Conclusiones

6.1. Conclusiones

La evolución continua que sufren los intereses de la sociedad provoca una demanda de resultados rápidos y avances a la tecnología. Especialmente si se considera el ámbito multimedia, donde los usuarios piden más calidad y una mayor interactividad con el medio.

La fluidez de uso de las tecnologías que mejora con cada generación implica que en un futuro lo que actualmente está en desarrollo puede llegar a ámbitos educativos. Hoy en día el estudio de métricas de calidad para vídeos 360 se encuentra solamente en la esfera de investigación. Más adelante es posible que el uso de programas como el desarrollado se extienda a centros educativos para enseñar de forma más detallada sobre la realidad virtual, los vídeos omnidireccionales y la evaluación de calidad de dichos vídeos.

Sobre el desarrollo técnico del proyecto, las fases de entendimiento de los algoritmos para su implementación, diseño e incorporación del código fueron especialmente densas pero por eso también fueron las fases más importantes.

El requisito principal era ofrecer una aplicación que recogiese diferentes métricas para la evaluación de vídeos omnidireccionales. Para ello se ha desarrollado con Python en Visual Studio Code.

Una vez planteada la aplicación de forma general, se han definido los requisitos siguiendo las necesidades presentadas por el tutor. El objetivo era incluir las métricas definidas PSNR, SSIM, MSSIM, S-PSNR, WS-PSNR, CPP-PSNR y OV-PSNR. Las últimas dos a causa del tiempo ajustado no pudieron ser desarrolladas, por lo tanto el objetivo no fue completado totalmente.

Se ha cumplido con el diseño original y parcialmente con los objetivos de la aplicación. La aplicación final ofrece la obtención de métricas comparativas y la extracción de esas métricas en un archivo externo.

En aspecto personal, este proyecto ha servido para aprender a desarrollar un proyecto en todas sus fases, no quedándome solamente con los apartados de diseño e implementación. Gracias a los conocimientos impartidos en la carrera sobre la gestión de proyectos he podido aplicarlos para completar esta aplicación.

6.2. Trabajo futuro

Teniendo en cuenta que no se ha podido realizar todos los objetivos que se tenían planteados, a continuación se definen varios elementos que hubiera sido interesante añadir en un futuro.

Ampliaciones de métricas
Las métricas CPP-PSNR y OV-PSNR no fueron implementadas. Se podrían añadir en un futuro con más tiempo.

Con el avance de las investigaciones en el ámbito de vídeos esféricos y por consiguiente una mejora en los algoritmos actuales, se deberá modificar las funciones respectivas a la codificación para aplicar estas mejoras. Y, en el caso de la creación de un paquete nuevo que permita aplicar una métrica con una simple llamada a función, modificar el código para actualizar la aplicación.

Añadir diferentes idiomas.
Sería interesante añadir el idioma Español por ser una aplicación desarrollada en una universidad española. Asimismo, se ha obtenido gran parte de la información y conocimiento de este ámbito gracias a equipos de investigación de China, por lo cual sería interesante traducirlo a su idioma.

Se podría cambiar el contenido del texto de todos los elementos a través de un botón en la interfaz, pero se necesitaría contratar a una persona con conocimientos de Chino para traducir de forma correcta.

7. Anexos

7.1. Anexo I: Código fuente y ejecutable

El código de la aplicación así como su manual de usuario, y ejecutable se puede encontrar en GitHub en el siguiente enlace:

https://github.com/faniaesc/qualiapp360

Para utilizar el programa se debe descargar el contenido al completo, ya que sin los ficheros y las carpetas predefinidas el funcionamiento no sería el deseado.

7.2. Anexo II: Icono de la aplicación

Para dotar de icono a la aplicación se ha creado un logo que combina lo más importante de la aplicación: la calidad y los vídeos 360º. Por ello, el logo se diseña a partir de la Q y el 360 del nombre de la aplicación. El logo está compuesto por la letra Q, y dos flechas ↺ definiendo cada una 360º.

Para definir el color se tomó en cuenta la integridad y accesibilidad de la aplicación. Evitando colores fáciles de confundir por personas con alteraciones genéticas se optó por el color azul por ser un color poco confundible, informativo y serio. Las personas que sufren Tritanopia, las cuales tienen ceguera al azul, visualizan el logo de un tono verdoso. Con el añadido del degradado y los bordes redondeados se refleja una apariencia más profesional.

En la ilustración 26 se muestra el diseño de icono finalizado realizado en Adobe Photoshop.

Ilustración 26: Diseño del logo de QualiApp360

7.3. Anexo III: Índice de tablas

- Tabla 1: Requisito Funcional 1 P.29
- Tabla 2: Requisito Funcional 2 P.30
- Tabla 3: Requisito Funcional 3 P.30
- Tabla 4: Requisito Funcional 4 P.30
- Tabla 5: Requisito Funcional 5 P.31
- Tabla 6: Requisito Funcional 6 P.31
- Tabla 7: Requisito Funcional 7 P.31
- Tabla 8: Requisito Funcional 8 P.32
- Tabla 9: Requisito Funcional 9 P.32
- Tabla 10: Requisito Funcional 10 P.32
- Tabla 11: Especificación: Ingresar datos P.35
- Tabla 12: Especificación Cargar Vídeo P.36
- Tabla 13: Especificación Cargar fichero texto P.36
- Tabla 14: Especificación: Seleccionar métricas P.37
- Tabla 15: Especificación: Ejecutar métricas P.37
- Tabla 16: Especificación: Seleccionar variables P.38
- Tabla 17: Especificación: Seleccionar gráfica P.38
- Tabla 18: Especificación: Descargar CSV P.39
- Tabla 19: Especificación: Abrir manual de usuario P.39
- Tabla 20: Planificación inicial P.45
- Tabla 21: Planificación final P.46
- Tabla 22: Total de horas finales por mes P.47
- Tabla 23: Costes del personal P.48
- Tabla 24: Costes del software P.48
- Tabla 25: Costes del hardware P.49
- Tabla 26: Costes totales del proyecto P.49
- Tabla 27: Matriz de probabilidad P.51
- Tabla 28: Estimación de los riesgos importantes P.52
- Tabla 29: Estrategias de mitigación P.52
- Tabla 30: Plan de contingencia P.53
- Tabla 31: Separación de la interfaz por bloques P.59
- Tabla 32: Tiempo estimado de cada métrica P.76

7.4. Anexo IV: Índice de ilustraciones

- Ilustración 1: Primera generación de HMD, Oculus Rift DK1. Fuente: xinreality P.11
- Ilustración 2: Comparativa de calidad entre dos vídeos 360. Fuente: medium P.11
- Ilustración 3: Transformación de resolución del vídeo al HMD. Fuente: 360rize P.12
- Ilustración 4: Flujo y organización de SSIM. Fuente: medium P.19
- Ilustración 5: Interfaz de BVQM. Fuente: BVQM User's Manual [PDF MANUAL] P.27
- Ilustración 6: Llamada de ejecución de 360tools desde la terminal de Visual Studio P.28
- Ilustración 7: Diagrama de casos de uso P.35
- Ilustración 8: Diagrama de Clases P.41
- Ilustración 9: Diagrama de secuencia de Cargar video, cargar fichero de texto y Abrir manual de Usuario. P.42
- Ilustración 10: Diagrama de secuencia de Seleccionar métricas, Ejecutar métricas, Seleccionar gráfica, Descargar CSV. P.43
- Ilustración 11: Metodología en cascada. P.44
- Ilustración 12: Interfaz de la aplicación QualiApp360 P.58
- Ilustración 13:Selección de vídeo desde directorio P.60
- Ilustración 14: Visualización de la gráfica obtenida con la evaluación de calidad P.61
- Ilustración 15: Muestra del CSV obtenido con los resultados de la evaluación P.62
- Ilustración 16: Implementación de la métrica PSNR P.63
- Ilustración 17: Implementación de la métrica SSIM P.65
- Ilustración 18: Implementación de la métrica MSSIM P.66
- Ilustración 19: Implementación de la métrica VMAF 360. P.67
- Ilustración 20: Implementación de la métrica S-PSNR P.68
- Ilustración 21: Inicialización de la métrica WS-PSNR P.69
- Ilustración 22: WS-PSNR, lectura de frames P.69
- Ilustración 23: Implementación de la métrica WS-PSNR P.71
- Ilustración 24: Uso de CPU en reposo P.77
- Ilustración 25: Uso de CPU ejecutando VMAF 360 P.77
- Ilustración 26: Diseño del logo de QualiApp360 P.82

7.5. Anexo V: Índice de ecuaciones

- Ecuación 1 - P.18
- Ecuación 2 - P.18
- Ecuación 3 - P.19
- Ecuación 4 - P.19
- Ecuación 5 - P.19
- Ecuación 6 - P.19
- Ecuación 7 - P.20
- Ecuación 8 - P.20
- Ecuación 9 - P.20
- Ecuación 10 - P.20
- Ecuación 11 - P.20
- Ecuación 12 - P.20
- Ecuación 13 - P.21
- Ecuación 14 - P.21
- Ecuación 15 - P.21
- Ecuación 16 - P.23
- Ecuación 17 - P.63
- Ecuación 18 - P.64
- Ecuación 19 - P.78

8. Bibliografía

[1] State-of-the-Art in 360° Video/Image Processing:Perception, Assessment and Compression. Mai Xu, ChenLi, Shanyi Zhang,and Patrick Le Callet. 2020

[2] https://github.com/Archer-Tatsu/VQA-ODV

[3] https://www.geeksforgeeks.org/python-peak-signal-to-noise-ratio-psnr/

[4] Image Quality Assessment: From Error Visibility to Structural Similarity
Zhou Wang, Alan Conrad Bovik, Hamid Rahim, Eero P. Simoncelli, 2004
https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

[5] SPHERICAL STRUCTURAL SIMILARITY INDEX FOR OBJECTIVE OMNIDIRECTIONAL VIDEO QUALITY ASSESSMENT
Sijia Chen, Yingxue Zhang, Yiming Li, Zhenzhong Chen and Zhou Wang
https://ece.uwaterloo.ca/~z70wang/publications/icme18.pdf

[6] https://github.com/Netflix/vmaf/blob/master/python/vmaf/script/run_vmaf.py

[7] https://github.com/cozcinar/VI_VMAF_4_360

[8] https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/PB2007108503.xhtml

[9] https://github.com/Samsung/360tools/

[10] https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

[11] https://github.com/I2-Multimedia-Lab/360-video-experimental-platform

[12] https://docs.opencv.org/4.5.2/dd/de7/group__videoio.html#details

[13] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[14]https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/