TRABAJO FIN DE GRADO

Interpolación de Frames de Video

Realizado por

Salvador Castagnino. 60590

Nicolas Birsa. 61482

Valentin Ye Li. 61011

Para la obtención del título de
INGENIERO EN INFORMÁTICA

Dirigido por
Pérez Sammartino, Francisco.

Convocatoria de Junio, curso 2024

Índice general

1. Introducción

1.1. Contexto

En los últimos años, la experiencia de visualización de videos ha experimentado significativos avances para mejorar la calidad de reproducción, tanto en términos de resolución como de la tasa de cuadros por segundo (FPS). Actualmente, el 60 % de los televisores tipo Smart TV vendidos tienen una resolución de 4K[4], y los monitores con tasas de actualización superiores a 60Hz están ganando popularidad.

Sin embargo, la obtención de videos con alta calidad nativa puede ser computacionalmente costosa, sin mencionar los gastos adicionales que implica el hardware necesario para reproducirlos, como las tarjetas gráficas o GPUs[5]. Este problema también se presenta en situaciones donde, aunque no se requiera una alta calidad de video, la obtención está limitada por el hardware utilizado, como ocurre con las cámaras IoT[6].

1.2. Justificación

Una solución a este problema es mejorar la calidad del video después de su captura utilizando técnicas de Inteligencia Artificial. Entre estas técnicas se incluyen la Super Resolución[7], que busca aumentar la definición de un video capturado a baja resolución, y la Interpolación de Fotogramas, cuyo objetivo es incrementar la tasa de cuadros por segundo (FPS) del video.

En particular, este estudio se enfoca en mejorar la tasa de cuadros por segundo de un video, una tarea con diversas aplicaciones como la compresión de video[8], la restauración de videos[9][10] o la creación de videos en cámara lenta[11][12].

1.3. Objetivo

El objetivo principal de este trabajo es diseñar un modelo capaz de realizar interpolación de fotogramas (VFI), generando una imagen intermedia a partir de otras dos tomadas consecutivamente en el tiempo.

Inicialmente, con imágenes en $t = 0$ y $t = 1$, se busca generar una imagen en el tiempo intermedio $t = 0{,}5$, con la posibilidad de expandir este proceso a otros valores temporales si el modelo lo permite.

De este modo, se podría generar una nueva secuencia como la observada en la Figura 1.1, donde se inserta la imagen generada entre las dos utilizadas como base,

efectivamente creando un video con aproximadamente el doble de fotogramas por segundo en comparación con el original.

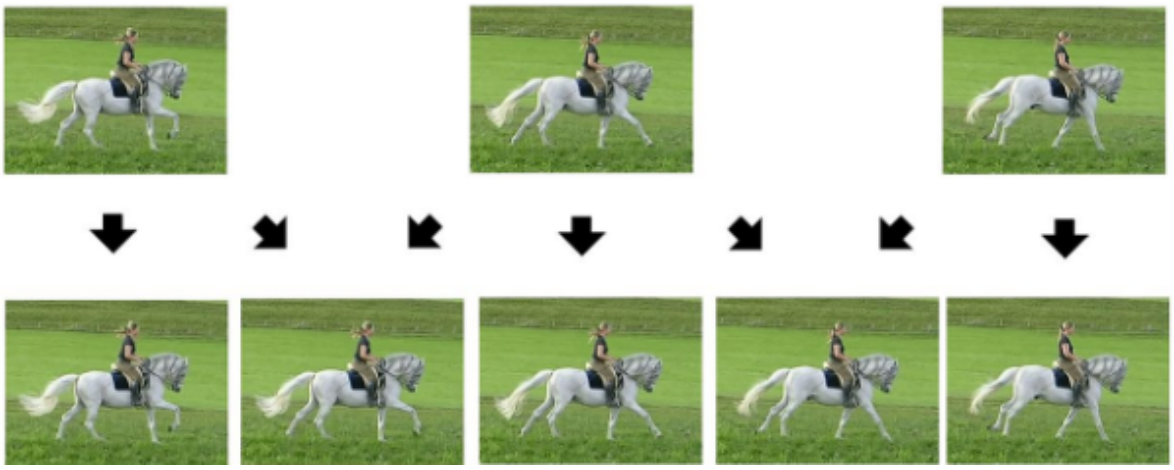

Figura 1.1: Posible secuencia generada a partir de tres imágenes.

2. Estado del Arte

2.1. Preprocesado

2.1.1. Flujo óptico

A la hora de generar una imagen intermedia, es fundamental que un modelo pueda determinar con precisión el movimiento de los objetos para su estimación. Por esta razón, algunas soluciones a este problema, como DLSS para la generación de fotogramas en videojuegos, utilizan información en forma de **Vectores de Movimiento**. Estos vectores indican cómo se mueve cada píxel en una imagen. Por ejemplo, si un píxel p se encuentra en la posición $(10, 20)$ en la imagen I_t, y en la siguiente imagen I_{t+1} está en $(20, 50)$, el vector de movimiento para p sería $(10, 30)$.

Sin embargo, en la mayoría de los casos, estos vectores no están disponibles previamente, por lo que el modelo debe ser capaz de calcularlos basándose en las imágenes proporcionadas. Aquí es donde entra en juego el concepto de **Flujo Óptico**, que representa el movimiento aparente de los patrones de brillo en una imagen, calculado bajo ciertos supuestos:

1. Las intensidades de los píxeles se mantienen constantes a lo largo del tiempo.
2. Existen vecindades de píxeles que se mueven en conjunto.

Dado que el movimiento de un objeto entre imágenes implica un desplazamiento de una vecindad de píxeles con intensidad constante, las condiciones se cumplen.

Por último, es crucial destacar que el Flujo Óptico representa una aproximación a los Vectores de Movimiento. Aunque el movimiento aparente capturado en las imágenes pueda ser similar en ambos métodos, esto no implica necesariamente que sus valores sean idénticos. Un ejemplo ilustrativo de esta diferencia se observa en la Figura 2.1, que describe la ilusión del poste de barbero: mientras el poste parece moverse horizontalmente, los patrones de brillo muestran un movimiento vertical.

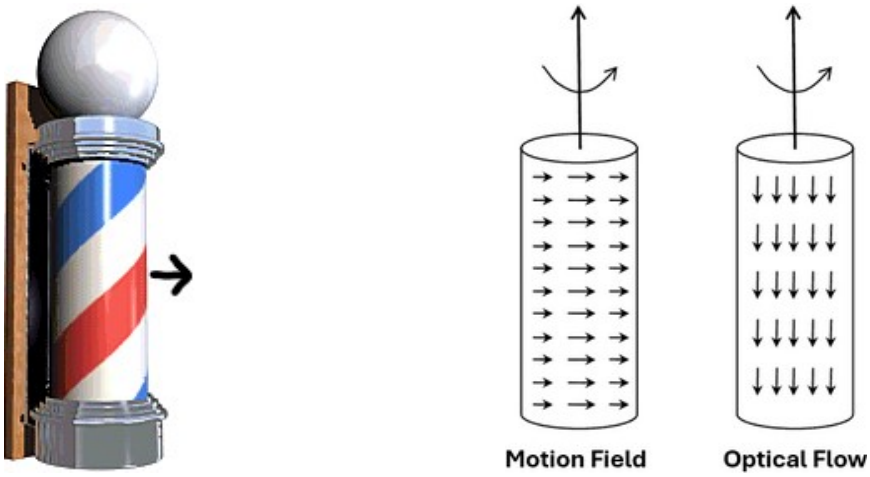

Figura 2.1: Comparación entre los Vectores de Movimiento y el Flujo Óptico en la ilusión del poste del barbero.

2.2. Modelos usados

Entre las soluciones actuales a este tipo de problema predominan dos modelos de deep learning: las **Redes Neuronales Convolucionales** (CNN) y los **Transformers**.

2.2.1. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNN) son ampliamente utilizadas para la clasificación de imágenes debido a su capacidad para capturar y aprender características jerárquicas de las imágenes de manera eficiente. Las CNN emplean capas convolucionales que aplican filtros a las imágenes de entrada, detectando patrones como bordes, texturas y formas en diferentes niveles de abstracción. El resultado de aplicar una capa convolucional a una imagen es conocido como feature map. Este último consiste de un arreglo de matrices en el cual cada matriz contiene información extraída por un filtro particular relacionado a una propiedad específica de la imagen. Esta estructura permite a las CNN reconocer objetos y detalles importantes con alta precisión.

Además, esta capacidad de reconocimiento de objetos permite estimar el flujo óptico al comparar la posición aproximada de un objeto en dos imágenes consecutivas, logrando una gran precisión con algunos ajustes adicionales[13][14].

Sin embargo, las CNN presentan limitaciones en la captura de interacciones de larga distancia debido a la naturaleza local de las operaciones de convolución, lo que les dificulta manejar movimientos rápidos. Este problema también se manifiesta en movimientos discontinuos entre imágenes, como los cambios de escena.

Figura 2.2: Comparación entre la imagen original de un Fórmula 1 en movimiento (izquierda) y una imagen generada por un modelo con CNN (derecha).

2.2.2. Transformers

El transformer [15] nace como una arquitectura originalmente ideada para la traducción de texto. Esta arquitectura presenta una forma revolucionaria de separar un texto en partes o unidades con valor propio y hacer que cada parte tenga una acceso a su contexto independientemente de su ubicación espacial. El proceso de separar el texto es la tokenización, las partes son los tokens y el mecanismo de acceso al contexto es la atención.

Una de las primeras aplicaciones de esta arquitectura es el modelo encoder-only BERT [16]. A través del uso del contexto bidireccional y una técnica de aprendizaje no supervisado llamada enmascarado, el modelo permite realizar tareas como clasificación de los términos del texto o clasificación del texto como un todo.

El transformer comienza a ser utilizado en imágenes a partir de la introducción del Vision Transformer [17]. Este paper presenta una arquitectura análoga a BERT para el procesamiento y clasificación de imágenes. Para esto presenta un mecanismo de tokenización y atención escalable para imágenes. El problema que resuelve el paper es la imposibilidad de tokenizar una imagen en píxeles, gracias a la gran cantidad de dependencias que esto generaría. La solución propuesta entonces es la de dividir a la imagen en ventanas disjuntas los cuales serán tokenizados y entre los cuales se aplicará la atención.

Resulta natural comparar el rendimiento de los Transformers para procesamiento de imágenes contra el de las CNN. Mientras que, a diferencia de las CNN, los transformers permiten acceder al contexto de manera independiente a la ubicación espacial estos exigen una gran cantidad de recursos y pierden propiedades deseables encontradas en las redes convolucionales como el bias inductivo, la capacidad de las CNN de aprovechar invariantes espaciales. Es la primer característica mencionada la que hace a los transformers interesantes para el objetivo planteado, la capacidad de tratar movimientos rápidos y a gran escala dentro de la imagen.

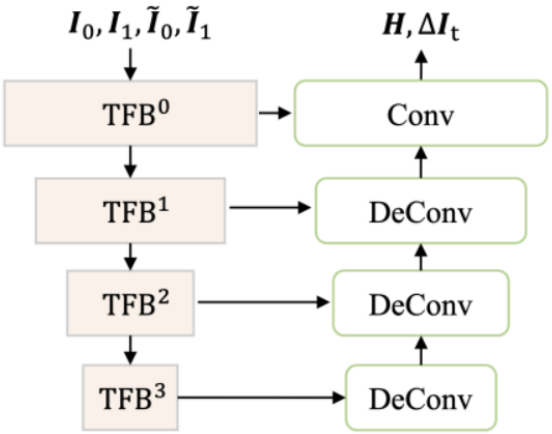

Figura 2.3: Arquitectura en forma de U del VFIformer

Se han considerado dos modelos de VFI desarrollados en los últimos años. En primer lugar encontramos el VFIformer [18]. Este se basa en la arquitectura U-Net [19] siguiendo su estructura encoder/decoder para capas convolucionales y deconvolucionales. A diferencia de esta última, el VFIformer apila capas convolucionales y deconvolucionales con capas de atención propias del modelo llamadas TFB con el fin de aprovechar lo mejor de cada una, como se observa en la Figura 2.3. A diferencia del Vision Transformer clásico, la atención no se aplica sobre ventanas en la imagen sino que sobre ventanas en los feature maps generados por las capas convolucionales. En primer lugar, la atención se calcula dentro de cada ventana entre los múltiples canales encontrados en un feature map. Para no limitarse únicamente al contexto dentro de cada ventana, a esto se le agrega aplicación de atención cruzada sobre ventanas escaladas, es decir, ventanas más abarcativas pero con menor definición que las originales, a este mecanismo lo llamaremos CSWA. Este mecanismo permite que cada ventana de la imagen generada tenga acceso a las diferentes propiedades de las imágenes originales encontradas en esa ventana más una vista periférica menos detallada de las propiedades en los alrededores de la ventana, como se observa en la Figura 2.4.

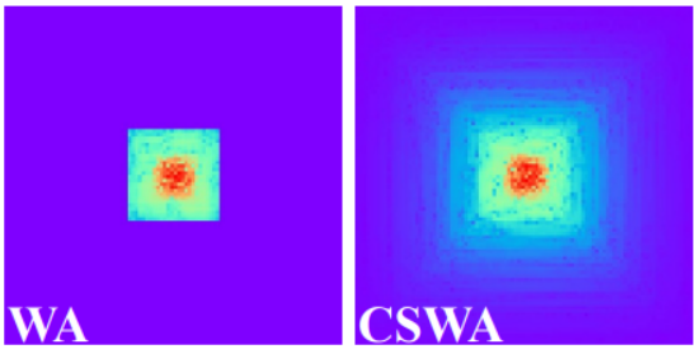

Figura 2.4: Atención sobre ventana con escalado (CSWA) contra atención sobre ventana sin escalado (WA)

En cada paso del proceso, el CSWA se verá aplicado al feature map generado por la capa anterior concatenado a los feature maps de las imágenes interpoladas, con el fin de no perder los frames originales de vista,

$$F_t^i = TFB^i([F_t^{i-1}, F_0^i, F_1^i]) \tag{2.1}$$

Siendo F_t^i el feature map a tiempo t para la i-esima capa. Sumado a eso, el paper utiliza flujo óptico para extraer movimiento de los frames iniciales y de los feature maps. El cálculo del flujo óptico es externo e independiente a la arquitectura. En conclusión, el diferencial del paper es el uso de CSWA, el cual le permite obtener un contexto amplio al momento de analizar el movimiento sin ver una pérdida de performance significativa.

El segundo modelo de VFI que consideraremos es el presentado en [20]. Esta arquitectura, al igual que el VFIformer, hace uso de capas convolucionales con el fin de obtener feature maps sobre los cuales se aplicarán los mecanismos de atención. Las principales diferencias se encuentran en la forma en la que se estudia el movimiento y en la forma en la que se relacionan los frames interpolados para realizar la síntesis. En primer lugar, el paper propone procesar la apariencia y el movimiento de manera separada, combinando estos solamente al momento de realizar la interpolación. Ambos se generan a partir del mecanismo de atención la diferencia es cómo se utiliza este último en su cálculo. Esto presenta varias ventajas como la capacidad de interpolar a un tiempo arbitrario y permite obtener información más detallada propia de cada proceso, estos siendo apariencia y movimiento. En segundo lugar, en lugar de combinar los frames originales concatenandolos, como se presentaba en VFIformer, se aplicará atención cruzada entre los feature maps de ambos frames. Al igual que en el VFIformer, la atención se aplica dentro de ventanas definidas sobre el feature map generado por cada capa convolucional. Sumado a esto el paper utiliza técnicas con el fin de obtener información fina presente en los frames originales la cual podría perderse gracias a la predominancia de los feature maps por sobre los frames originales. Entre estas encontramos el uso de Convoluciones Dilatadas y el uso de una red de refinamiento para los resultados obtenidos [21].

2.3. Datasets

En la actualidad, los datasets utilizados suelen consistir en secuencias de tres imágenes temporalmente consecutivas, denominadas I_0, I_1, e I_2. El procedimiento general consiste en proporcionar al modelo las imágenes I_0 e I_2 como entrada y prever una imagen cercana a I_1 como salida.

Entre algunos de los datasets comúnmente empleados para entrenar este tipo de modelos se incluyen:

2.3.1. Vimeo90k[1]

Este dataset consta principalmente de 73,171 secuencias de tres imágenes con una resolución fija de 448x256 píxeles. Además, presenta 91,707 secuencias de siete imágenes del mismo tamaño, permitiendo entrenar al modelo con un mayor número de pasos temporales.

2.3.2. UCF101[2]

Compuesto por 13,320 clips de video extraídos de YouTube, con una resolución de 320x240 píxeles y 25 FPS. Están clasificados en 101 categorías, que abarcan movimientos del cuerpo humano, interacciones sociales, interacciones humanas con objetos, interpretación musical y deportes.

2.3.3. X4K1000FPS[3]

Este dataset incluye videoclips con una resolución de 4096x2160 (4k) y una tasa de 1000 FPS. Cada video presenta objetos en movimiento rápido, permitiendo al modelo aprender a interpolar objetos en situaciones de alta velocidad.

2.4. Métricas

2.4.1. Métricas tradicionales

En el campo de la interpolación de fotogramas, la calidad de un video se mide comúnmente basándose en las imágenes que lo componen. Por esta razón, los modelos se suelen comparar evaluando las imágenes generadas en relación con las imágenes de referencia, utilizando las siguientes métricas:

Peak Signal To Noise Ratio (PSNR)

Dadas dos imágenes I_a e I_b, ambas de tamaño HxW, se define como:

$$PSNR(I_a, I_b) = 10 * log_{10}\left(\frac{255^2}{MSE(I_a, I_b)}\right) \tag{2.2}$$

siendo:

$$MSE(I_a, I_b) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(I_{a_{i,j}} - I_{b_{i,j}})^2 \tag{2.3}$$

Al utilizar la inversa del Error Cuadrático Medio, esta métrica compara los valores de los píxeles en ambas imágenes. Así, cuanto mayor sea su valor, más parecidas serán las dos imágenes.

Structural Similarity Index Measure (SSIM)

Dadas dos imágenes I_a e I_b, se define como:

$$SSIM(I_a, I_b) = l(I_a, I_b).c(I_a, I_b).s(I_a, I_b) \tag{2.4}$$

siendo:

$$l(I_a, I_b) = \frac{2\mu_a\mu_b + c_1}{\mu_a^2 + \mu_b^2 + c_1} \tag{2.5}$$

$$c(I_a, I_b) = \frac{2\sigma_a\sigma_b + c_2}{\sigma_a^2 + \sigma_b^2 + c_2} \tag{2.6}$$

$$s(I_a, I_b) = \frac{\sigma_{ab} + c_3}{\sigma_a\sigma_b + c_3} \tag{2.7}$$

con:

μ_a la muestra media de píxeles en I_a.

μ_b la muestra media de píxeles en I_b.

σ_a^2 la varianza de I_a.

σ_b^2 la varianza de I_b.

σ_{ab} la covarianza de I_a e I_b.

$c_1 = (0{,}01L)^2$, $c_2 = (0{,}03L)^2$ y $c3 = C_2/2$ variables para estabilizar la división con denominadores chicos

L el rango dinámico de valores de los píxeles (típicamente $2^{\#bitsporpixel} - 1$)

A diferencia de la métrica anterior, SSIM se enfoca en ponderar la luminancia, contraste y estructura de la imagen a través de las fórmulas respectivas l, c y s. Debido a esto, ambas métricas son complementarias y suelen utilizarse en conjunto para evaluar el rendimiento de un modelo.

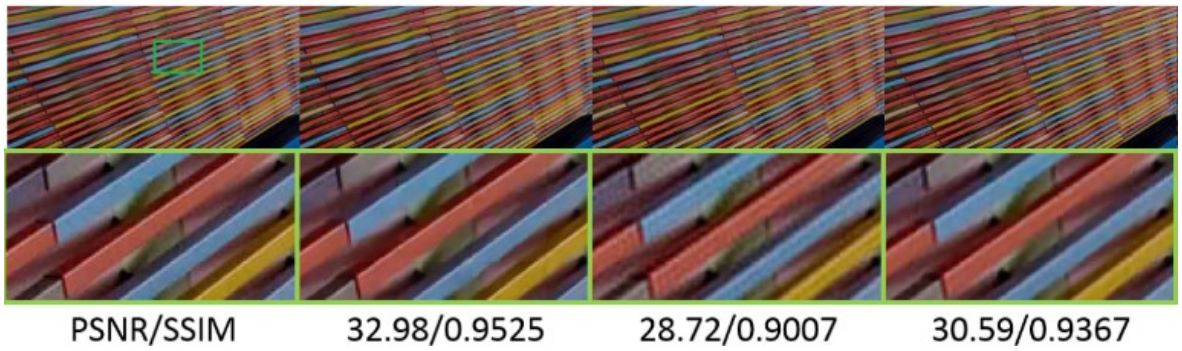

Figura 2.5: Comparación entre imágenes generadas y sus valores PSNR/SSIM. Extraído de [22].

Method	Vimeo90K	UCF101	Middlebury	SNU-FILM			
				Easy	Medium	Hard	Extreme
ToFlow [51]	33.73/0.9682	34.58/0.9667	2.15	39.08/0.9890	34.39/0.9740	28.44/0.9180	23.39/0.8310
SepConv [35]	33.79/0.9702	34.78/0.9669	2.27	39.41/0.9900	34.97/0.9762	29.36/0.9253	24.31/0.8448
CyclicGen [25]	32.09/0.9490	35.11/0.9684	-	37.72/0.9840	32.47/0.9554	26.95/0.8871	22.70/0.8083
DAIN [2]	34.71/0.9756	34.99/0.9683	2.04	39.73/0.9902	35.46/0.9780	30.17/0.9335	25.09/0.8584
CAIN [9]	34.65/0.9730	34.91/0.9690	2.28	39.89/0.9900	35.61/0.9776	29.90/0.9292	24.78/0.8507
AdaCoF [22]	34.47/0.9730	34.90/0.9680	2.24	39.80/0.9900	35.05/0.9754	29.46/0.9244	24.31/0.8439
BMBC [36]	35.01/0.9764	35.15/0.9689	2.04	39.90/0.9902	35.31/0.9774	29.33/0.9270	23.92/0.8432
RIFE-Large [17]	36.10/0.9801	35.29/0.9693	1.94	40.02/0.9906	35.92/0.9791	30.49/0.9364	25.24/0.8621
ABME [37]	36.18/0.9805	35.38/0.9698	2.01	39.59/0.9901	35.77/0.9789	30.58/0.9364	25.42/0.8639
Ours	36.50/0.9816	35.43/0.9700	1.82	40.13/0.9907	36.09/0.9799	30.67/0.9378	25.43/0.8643

Figura 2.6: Comparación entre modelos usando como métrica PSNR/SSIM. Extraído de [18].

2.4.2. Métricas actuales

A pesar de la alta utilidad del par de métricas PSNR/SSIM presentado anteriormente, estudios recientes[23] han demostrado que un valor más alto en estas métricas no siempre se traduce en una mayor calidad de video percibida por las personas.

Por lo tanto, se han comenzado a adoptar otras métricas como LPIPS[23] y DISTS[24] para evaluar la calidad de imagen, así como VFIPS[25] y FloLPIPS[26] para medir la calidad de video. Estas métricas muestran una mayor correlación con la percepción humana de la calidad de interpolación de fotogramas.

Por consiguiente, una estrategia viable para evaluar un modelo podría ser basarse en los resultados obtenidos por estas métricas, mientras se mantiene el par PSNR/SSIM por razones de completitud en comparación con modelos anteriores.

Bibliografía

[1] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. *Video enhancement with task oriented flow.* International Journal of Computer Vision, 2019.

[2] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *Ucf101: A dataset of 101 human actions classes from videos in the wild*. arXiv preprint arXiv:1212.0402, 2012.

[3] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. *Xvfi: Extreme video frame interpolation*. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021.

[4] Rohan Jambhale. Smart tv statistics 2024 by market size, share and trends, 2024. URL https://www.coolest-gadgets.com/smart-tv-statistics.

[5] Monica J. White. No surprise — graphics cards have gotten twice as expensive since 2020, 2023. URL https://www.digitaltrends.com/computing/gpu-prices-2020-vs-2023-compared/.

[6] Bandhav Veluri, Collin Pernu, Ali Saffari, Joshua Smith, Michael Taylor, and Shyamnath Gollakota. Neuricam: Key-frame video super-resolution and colorization for iot cameras. *Association for Computing Machinery*, 2023.

[7] Hongying Liu, Zhubo Ruan, Peng Zhao, Chao Dong, Fanhua Shang, Yuanyuan Liu, Linlin Yang, and Radu Timofte. *Video super-resolution based on deep learning: a comprehensive survey*. Artificial Intelligence Review, 2022.

[8] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. *Video compression through image interpolation*. Proceedings of the European conference on computer vision (ECCV), 2018.

[9] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. *Space-time-aware multi-resolution video enhancement*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.

[10] Soo Ye Kim, Jihyong Oh, and Munchurl Kim. *Fisr: deepjoint frame interpolation and super-resolution with a multiscale temporal loss*. Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

[11] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. *Super slomo: High quality estimation of multiple intermediate frames for video interpolation*. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018.

[12] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. *Depth-aware video frame interpolation*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[13] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. *Flownet 2.0: Evolution of optical flow estimation with deep networks*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[14] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. *Rife: Real-time intermediate flow estimation for video frame interpolation*. arXiv preprint arXiv:2011.06294, 2020.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[16] Kenton Lee AJacob Devlin, Ming-Wei Chang and Kristina Toutanova. An image is worth 16x16 words: Transformers for image recognition at scale. 2019.

[17] Alexander Kolesnikov Dirk Weissenborn Xiaohua Zhai Thomas Unterthiner Mostafa Dehghani Matthias Minderer Georg Heigold Sylvain Gelly Jakob Uszkoreit Neil Houlsby Alexey Dosovitskiy, Lucas Beyer. Bert: Pre-training of deep bidirectional transformers for language understanding. 2022.

[18] Liying Lu, Ruizheng Wu, Huaijia Lin, Jiangbo Lu, and Jiaya Jia. Video frame interpolation with transformer. 2022.

[19] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation, 2015.

[20] Guozhen Zhang, Yuhan Zhu, Haonan Wang, Youxin Chen, Gangshan Wu, and Limin Wang. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. 2023.

[21] Chunhua Shen Ian Reid Guosheng Lin, Anton Milan. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, 2016.

[22] Jian Zhang, Chen Zhao, and Wen Gao. *Optimization-Inspired Compact Deep Compressive Sensing*. IEEE Journal of selected topics in signal processing, 2020.

[23] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. *The unreasonable effectiveness of deep features as a perceptual metric*. CVPR, 2018.

[24] Keyan Ding, Kede Ma, Shiqi Wang, , and Eero P. Simoncelli. *Image quality assessment: Unifying structure and texture similarity*. arXiv preprint arXiv:2004.07728, 2020.

[25] Qiqi Hou, Abhijay Ghildyal, and Feng Liu. *A perceptual quality metric for video frame interpolation*. ECCV, 2022.

[26] Duolikun Danier, Fan Zhang, and David R. Bul. *Flolpips: A bespoke video quality metric for frame interpolation*. PCS, 2022.