Superresolución de imágenes de satélite con Gaussian Splatting

Adrián García Tapia

12 de abril de 2025

Resumen– Resum del projecte, màxim 10 línies.

Palabras clave– Superresolución, Escala Arbitraria, Gaussian Splatting, Sentinel-2.

Abstract– Versió en anglès del resum.

Keywords– Superresolution, Arbitrary-Scale, Gaussian Splatting, Sentinel-2.

1 Introducción

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2 Estado del arte

2.1. Superresolución

La superresolución (SR) se divide en: superresolución fija y superresolución arbitraria (ASR), y en superresolución con una sola imagen (SISR) y con múltiples imágenes (MISR). Para este trabajo, es de interés la ASR y la SISR.

La ASR en su mayoría se hace con una sola imagen, por lo que también entra en la categoría de SISR. Esta consiste en entrenar un único modelo para que sea capaz de mejorar la resolución de una imagen en cualquier factor, como por ejemplo x2, x3, x4, etc. incluidos factores decimales. Meta-SR [8] es un método basado en operadores de interpolación, y fue el primero en hacer ASR dentro de los distintos métodos de deep learning que hay actualmente. Sin embargo, el estado del arte hasta hace poco han sido los métodos basados en representaciones neuronales implícitas (INR), inspirados en el éxito que han tenido estos en tareas de reconstrucción 3D con los NeRF [6]. LIIF [4] fue pionero en adaptar los métodos INR a ASR y lo siguieron otros métodos como ITSRN [18], SRNO [16] y CiaoSR [2], siendo este último el actual estado del arte de esta familia de modelos.

2.2. Superresolución con Gaussian Splatting

Una vez más inspirados por avances en el mundo del 3D, últimamente se ha empezado a explorar el uso de Gaussian Splatting (GS) [9] en ASR gracias al éxito que han tenido frente a los NeRF.

El primero en llevar esta técnica a ASR es GaussianSR [7], que a pesar de ser superado por CiaoSR en la calidad de las imágenes finales, rivaliza con LIIF e ITSRN superándolos tanto en calidad como en velocidad de inferencia. GSASR [3] es el siguiente avance en esta área, superando a CiaoSR tanto en calidad de la imagen final como en tiempo de inferencia. Una de las diferencias clave respecto a GaussianSR es el uso de diversos mecanismos de atención y un rasterizador 2D implementado en cuda, que tiene su origen en la primera implementación de GS en 3D.

2.3. Superresolución Sentinel-2

En el área de las imágenes satelitales (RS) los métodos de SR suelen variar dependiendo del satélite de origen. En este caso nos centramos es Sentinel-2, una constelación formada por 2 satélites que proporcionan imágenes de 13 canales distintos, que van desde los 443nm hasta los 2190nm del espectro electromagnético [1]. Estos canales se encuentran a distintas resoluciones: 10m/px, 20m/px y 60m/px.

Algunos de los métodos actuales se centran en unir todos los canales a una misma resolución de 10m/px. HFN [17] es un método convolucional que junta información de todos los canales para llevarlos a 10m/px de forma jerárquica. El transformer propuesto en [13] es un método reciente que también logra el mismo objetivo.

Por otro lado, hay métodos que intentan mejorar la resolución más allá de los 10m/px, aunque se suelen centrar más en el RGB

• E-mail de contacto: adrian.garciat@autonoma.cat
• Mención realitzada: Computació
• Treball tutoritzat per: Felipe Lumbreras Ruiz (Ciències de la Computació)
• Curs 2024/25

en lugar de unir la información de todos los canales y suelen dejar fuera los que están originalmente a 60m/px. SENX4 [5] aplica una mejora de x4 a los canales de 10m/px utilizando una red convolucional. RS-ESRGAN [12] se basa en el ESRGAN original [14] para aplicar una mejora de x5 a los canales de 10m/px. Por desgracia estos últimos no suelen utilizar la información de todas las bandas para esta mejora.

Si saltamos al campo hiperespectral, DMGASR [15] es una técnica de difusión que permite utilizar una gran cantidad de canales C para llevar a cabo la superresolución, cosa que podría aplicarse a las imágenes de Sentinel-2.

3 OBJETIVOS

El objetivo principal de este trabajo es implementar el modelo GSASR, adaptarlo a las imágenes multiespectrales de Sentinel-2 y ver qué mejora hay respecto a la versión RGB. Para comprobar esto, usaré solo los tres canales RGB de Sentinel-2 para la versión RGB y todos los canales para la versión multiespectral.

Aunque el anterior sea el principal objetivo del trabajo, en caso de haber tiempo suficiente también pretendo comparar el modelo respecto a las adaptaciones multiespectrales de los distintos métodos del estado del arte actual. Como propuestas escogeré CiaoSR, el Transformer propuesto en [13] y DMGASR. Además, al final de todo trataría de combinar HFN, que implementé durante mi estancia en las prácticas con éxito, para aplicar GSASR a partir de una resolución común de 10m/px en todos los canales.

En la Tabla 1 hay un desglose de los objetivos en subobjetivos más pequeños.

4 METODOLOGÍA

Como metodología a seguir he decidido optar por Kanban porque no tiene un gran componente de trabajo en equipo, y como solo tengo que organizarme yo es suficiente. Trello será mi tablero Kanban en el que llevar un registro del estado de mis tareas. Las tareas del proyecto están sacadas de la Tabla 1 y se han organizado en el tiempo en un diagrama de Gantt que se encuentra en el dosier de este trabajo.

Durante todo el proceso me centraré en conseguir el objetivo principal mencionado en la Sección 3, sin tener en cuenta tareas de los siguientes objetivos extra para evitar posibles distracciones. Será cuando haya conseguido cumplir con el objetivo principal que empezaré a incluir las tareas adicionales.

5 PLANIFICACIÓN

Para planificar las tareas en el tiempo he hecho un diagrama de Gantt usando MS Project. Las tareas están extraídas directamente de la Tabla 1. El diagrama final se encuentra dividido en orden en las Figuras **??**, **??**, **??**, **??**, **??** y **??**. Las tareas incluidas de la fase de Inicio terminan con la entrega de este informe, cuadrando con la planificación estimada. A continuación empezaré con la fase de implementación del artículo.

Considero que la previsión es bastante optimista por mi parte, pero de esta forma doy pie a la posibilidad de hacer todas las tareas incluyendo las extra.

6 DESARROLLO

En este apartado se detalla el progreso realizado hasta la fecha, las distintas fases del desarrollo y algunos de los problemas encontrados durante la implementación. El núcleo principal del trabajo ha consistido en la reproducción e implementación del modelo descrito en el artículo original, el cual presenta una arquitectura híbrida que combina componentes desarrollados en PyTorch con módulos implementados en CUDA.

Tal como se ilustra en la Figura 1, el modelo está compuesto por tres bloques fundamentales: un *Encoder*, encargado de extraer las características relevantes de la imagen; un *Decoder*, que genera las representaciones gausianas; y un *Rasterizador 2D*, que produce la imagen final a partir de dichas representaciones.

6.1. Encoder

El *Encoder* es el módulo responsable de la extracción de características a partir de la imagen de entrada. Está compuesto por un *backbone* de un modelo de superresolución ya existente, al cual se le añade una capa de convolución adicional que adapta su salida para que sea compatible con la entrada del *Decoder*. En esta implementación se ha utilizado el modelo EDSR[10] como *backbone*, siguiendo una de las configuraciones propuestas en el trabajo original. No obstante, es posible sustituirlo por otros modelos de características.

6.2. Decoder

El *Decoder* se encarga de transformar las características extraídas por el *Encoder* en una representación gausiana que sirva como base para la generación de la imagen final. Este módulo se divide en tres bloques principales: el *Condition Injection Block*, el *Gaussian Interaction Block* y el *Gaussian Primary Head*.

6.2.1. Condition Injection Block

Este bloque recibe como entrada tanto las características generadas por el *Encoder* como un *embedding* entrenable que contiene información inicial sobre las gausianas. Su objetivo principal es fusionar ambos tipos de información mediante un mecanismo de atención cruzada con ventanas, inspirado en la arquitectura Swin Transformer[11], como se detalla en la Ecuación(1). El resultado de este bloque son los *embeddings* gaussianos iniciales.

$$Attention\,(Q,K,V) = SoftMax\left(\frac{QK^T}{\sqrt{d}} + B\right)V \qquad (1)$$

6.2.2. Gaussian Interaction Block

Este bloque consta de dos fases diferenciadas. En la primera, se combinan los *embeddings* gaussianos obtenidos en el bloque anterior con un factor de escalado, utilizando un mecanismo de atención cruzada descrito en la Ecuación (2). En este esquema, las *queries* provienen de los *embeddings*, mientras que las *keys* y *values* derivan del factor de escala.

$$Attention\,(Q,K,V) = SoftMax\left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (2)$$

En la segunda fase, se introduce interacción entre los diferentes *embeddings* gaussianos mediante atención por ventanas desplazadas, siguiendo el enfoque propuesto en Swin Transformer[11]. Este proceso se repite en serie a lo largo de L bloques de tipo *Gaussian Interaction*, permitiendo una integración progresiva de la información. Los *embeddings* resultantes de esta etapa contienen ya todos los parámetros necesarios para definir las gausianas finales.

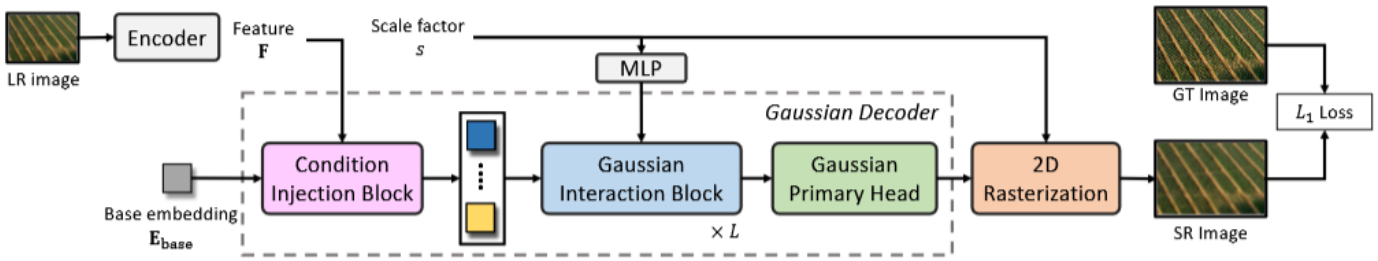

Fig. 1: Arquitectura GSASR.

6.2.3. Gaussian Primary Head

El bloque final del *Decoder* tiene como propósito extraer los parámetros individuales de cada gausiana. Para ello, toma los *embeddings* producidos por el bloque anterior y los procesa mediante cinco redes MLP independientes, las cuales predicen la opacidad, el color, la desviación estándar, el desplazamiento (*offset*) y la correlación de cada gausiana. Finalmente, el *offset* se suma a la posición relativa estimada para determinar la localización exacta de cada gausiana en el espacio de salida.

6.3. Rasterizador 2D

Este componente es responsable de generar la imagen final a partir de las gausianas obtenidas, aunque no incluye parámetros entrenables. Cada gausiana calcula su contribución a los píxeles cercanos empleando la función de densidad gausiana representada en la Ecuación (3), y su efecto acumulativo sobre la imagen final se determina según la Ecuación (4).

$$f(x,y) = \left(2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}\right)^{-1} exp\Bigg[-\frac{1}{2(1-\rho^2)} \times \left(\frac{\Delta x^2}{\sigma_x^2} - \frac{2\rho\Delta x\Delta y}{\sigma_x\sigma_y} + \frac{\Delta y^2}{\sigma_y^2}\right)\Bigg] \tag{3}$$

$$G(x,y) = \alpha c f(x,y) \tag{4}$$

En dichas ecuaciones, α representa la opacidad, c el color, y ρ la correlación, que junto con la desviación estándar σ define la forma y orientación de la gausiana. Los términos Δx y Δy corresponden a la distancia entre el centro de la gausiana y el píxel de destino.

El procedimiento general seguido para llevar a cabo esta etapa se resume en el Algoritmo 1.

Algorithm 1 Algoritmo de rasterización 2D

Input: Two integers a and b
Output: T H E V O I D

Initialize I_{SR} as an (sH, sW, 3) array of zeros.
for each G_i in $\{G_1, G_2, ..., G_N\}$ **do**
  Initialize $\alpha, \mu_x, \mu_y, \sigma_x, \sigma_y, \rho, c$ from G_i.
  for each pixel (x, y) in I_{SR} **do**
    if $|x - \mu_x| < rsH$ and $|y - \mu_y| < rsW$ **then**
      Obtain $f(x/s, y/s)$ using Eq. (3);
      Obtain $G_i(x/s, y/s)$ using Eq. (4);
      $I_{SR}(x, y; s) += G_i(x/s, y/s)$
    end if
  end for
end for

Para garantizar un rendimiento computacional adecuado, esta parte ha sido implementada en CUDA, a diferencia del resto del modelo que se ha desarrollado íntegramente en Python utilizando la biblioteca PyTorch.

7 Progreso

La implementación completa del modelo ha sido finalizada, al igual que la construcción del bucle de entrenamiento. Para evaluar el rendimiento del modelo se ha definido como métrica principal el Peak Signal-to-Noise Ratio (PSNR), mientras que como métricas secundarias se emplean el Structural Similarity Index Measure (SSIM), Learned Perceptual Image Patch Similarity (LPIPS) y Deep Image Structure and Texture Similarity (DISTS).

En la actualidad, el modelo se encuentra en proceso de entrenamiento utilizando una tarjeta gráfica NVIDIA RTX 3090 con 24 GB de memoria VRAM. El tamaño del *batch* se ha fijado en 4 imágenes, extraídas del conjunto de datos DIV2K. Dichas imágenes han sido preprocesadas previamente mediante una segmentación en fragmentos más pequeños, con el objetivo de optimizar el uso de memoria durante el entrenamiento.

El propósito fundamental del entrenamiento es que el modelo aprenda a realizar superresolución de forma arbitraria. Para ello, se entrena con factores de escala ×2, ×3 y ×4, siendo la resolución objetivo para las imágenes generadas de 144×144 píxeles.

Durante el proceso, se identificaron ciertos problemas relacionados con la eficiencia del rasterizador 2D, lo que inicialmente ralentizó el ritmo de entrenamiento más de lo esperado. No obstante, tras algunos ajustes y optimizaciones, dichas dificultades han sido resueltas. En la actualidad, el modelo alcanza una velocidad de aproximadamente dos iteraciones por segundo.

Este contratiempo ha generado un ligero retraso de un par de días respecto a la planificación original. Sin embargo, dicho desfase no supone un obstáculo significativo para el cumplimiento de los plazos establecidos.

8 Conclusiones

Agradecimientos

Gracias a Arnau Marcos por ayudarme a optimizar el código de CUDA. Gracias también a Jordi Ventosa y Paula Font por el apoyo moral (me han obligado a incluirlos).

Referencias

[1] European Space Agency. *Sentinel-2 User Handbook*, 2015. Accessed: 25/02/2025.

[2] Jiezhang Cao, Qin Wang, Yongqin Xian, Yawei Li, Bingbing Ni, Zhiming Pi, Kai Zhang, Yulun Zhang, Radu Timofte, and Luc Van Gool. Ciaosr: Continuous implicit attention-in-attention network for arbitrary-scale image super-resolution, 2023.

[3] Du Chen, Liyi Chen, Zhengqiang Zhang, and Lei Zhang. Generalized and efficient 2d gaussian splatting for arbitrary-scale super-resolution, 2025.

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function, 2021.

[5] Mikel Galar, Rubén Sesma, Christian Ayala, Lourdes Albizua, and Carlos Aranda. Super-resolution of sentinel-2 images using convolutional neural networks and real ground truth data. *Remote Sensing*, 12(18), 2020.

[6] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2023.

[7] Jintong Hu, Bin Xia, Bin Chen, Wenming Yang, and Lei Zhang. Gaussiansr: High fidelity 2d gaussian splatting for arbitrary-scale image super-resolution, 2024.

[8] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution, 2019.

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[10] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[12] Luis Salgueiro Romero, Javier Marcello, and Verónica Vilaplana. Super-resolution of sentinel-2 imagery using generative adversarial networks. *Remote Sensing*, 12(15), 2020.

[13] Alireza Sharifi and Mohammad Mahdi Safari. Enhancing the spatial resolution of sentinel-2 images through super-resolution using transformer-based deep-learning models. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 18:4805–4820, 2025.

[14] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

[15] Zhaoyang Wang, Dongyang Li, Mingyang Zhang, Hao Luo, and Maoguo Gong. Enhancing hyperspectral images via diffusion model and group-autoencoder super-resolution network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(6):5794–5804, March 2024.

[16] Min Wei and Xuesong Zhang. Super-resolution neural operator, 2023.

[17] Jingan Wu, Liupeng Lin, Chi Zhang, Tongwen Li, Xiao Cheng, and Fang Nan. Generating Sentinel-2 all-band 10-m data by sharpening 20/60-m bands: A hierarchical fusion network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:16–31, 2023.

[18] Jingyu Yang, Sheng Shen, Huanjing Yue, and Kun Li. Implicit transformer network for screen content image continuous super-resolution, 2021.

APÉNDICE

O1(T)	O2(F)	O3(P)
Implementar RGB y Multiespectral.	Implementar RGB.	Crear repositorio de GitHub.
		Entender Condition Injection Block.
		Entender Gaussian Interaction Block.
		Entender Gaussian Primary Head.
		Entender 2D Rasterization Block.
		Aprender a combinar PyTorch+cuda.
		Estudiar las implementaciones en cuda de GS.
		Implementar Condition Injection Block.
		Implementar Gaussian Interaction Block.
		Implementar Gaussian Primary Head.
		Implementar 2D Rasterization.
		Preparar EDSR feature extractor.
		Conectar todos los bloques para el modelo final.
		Crear bucle de entrenamiento.
		Preparar Dataset RGB.
		Entrenar modelo.
		Comparar.
	Implementar MS.	Adaptar Condition Injection Block a MS.
		Adaptar Gaussian Interaction Block a MS.
		Adaptar Gaussian Primary Head a MS.
		Adaptar 2D Rasterization a MS.
		Adaptar EDSR feature extractor a MS.
		Preparar Dataset MS para EDSR.
		Preparar Dataset MS para GSASR.
		Crear bucle de entrenamiento para EDSR.
		Entrenar EDSR.
		Entrenar GSASR.
		Comparar.
Comparar con SotA Multiespectral	Comparar CiaoSR.	Preparar CiaoSR para MS.
		Crear bucle de entrenamiento.
		Entrenar CiaoSR.
		Comparar.
	Comparar Transformer SotA.	Preparar Transformer para MS.
		Crear bucle de entrenamiento.
		Entrenar Transformer.
		Comparar.
	Comparar Difusión SotA.	Preparar modelo para MS.
		Crear bucle de entrenamiento.
		Entrenar modelo de Difusión.
		Comparar.
Combinar HFN con GSASR.	HFN + GSASR.	Preparar HFN.
		Crear bucle de entrenamiento.
		Entrenar HFN.
		Comparar.

TABLA 1: TABLA DE OBJETIVOS